A Comparison of Machine Learning Approaches for Spam E-mail Classification

Filipe Gama Batista
Leiden University
Leiden, Netherlands
s1780514@leidenuniv.nl

ABSTRACT

The purpose of this project was to compare the performance of three different machine learning approaches - naive bayes, support vector machines, and k-nearest neighbors - when used to solve the problem of automatically labeling e-mails as spam or ham[1]. In this paper, brief explanations of the algorithms are presented, as well as experimental results of the performance of these methods when applied on 3 different real-world datasets.

CCS Concepts

•Information systems → Information retrieval;
•Computing methodologies → Machine learning;

Keywords

Spam; E-mail; Classification; Machine learning; Text-mining

1. INTRODUCTION

E-mail spam (common name for junk/unsolicited email) has been a problem ever since the e-mail became mainstream. Spam emails are not only an annoyance for most users, cluttering their inboxes, or making them miss important information lost in the middle of spam emails, but also for internet providers that face huge costs and problems with traffic between servers. The European Union's Internal Market Commission estimated that spam cost EUR 10 billion per year worldwide in 2001[7].

In 2015, according to statistics from Kaspersky Lab, the proportion of spam in email flows was 55.28%[1], and the total number of worldwide email accounts is expected to increase from 3.3 billion accounts in 2012 to over 4.3 billion accounts by year-end 2016. [4]

[1]ham will be the word used in this paper to refer legit emails (i.e. not spam)

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

We can see that spam is still a problem nowadays, that is expected to affect more and more people in the future. However, fighting spam is a not an easy task. *Spammers* make use of simple tricks to overcome the filtering, such as using random sender addresses and append random characters to the beginning or the end of the message subject line.

Using knowledge engineering approaches for spam filtering is not practical anymore, since it implies defining rules, updating and maintaining them periodically by some company or entity, which costs time and money.[2].

Machine learning, on the other hand, offers a way to make spam filters able to learn automatically what is spam or ham, by using a set of training samples with pre-classified spam/ham e-mails. This approach does not require any rules to be specified manually[3].

This problem can be seen as a simple Text Categorization task where the classes to be predicted are spam and ham.[10] There are multiple machine learning algorithms that could be used for this task, and in this paper the goal was to find which one would perform better, as well as if there were any significant differences between using different datasets found online and try to understand those differences. For that purpose, three datasets were used, and three different machine learning methods were chosen (out of many others that could be used), those being: naive bayes, support vector machine (linear) and k-Nearest neighbors.

In the following section I'll discuss some related work and previous results, then in section 3 I explain how the 3 machine learning methods can be applied to this problem, section 4 describes the experimental setup, section 5 the results and finally the conclusions, in section 6.

2. RELATED WORK

Comparisons between different machine learning algorithms have been done before. W.A. Awad and S.M Elseuofi performed a comparison between 6 different machine learning approaches in [2], those being: Naive bayes, SVM, kNN, Neural Networks, Artificial Immune System and Rough sets. To analyze the performance of the mentioned algorithms they used a publicly available dataset SpamAssassin. One difference between their approach and the one presented in this paper is that only the top 100 features (most relevant words) were used, while in this paper all the features were used without selection. Another main difference was the use of k-fold cross validation (with k=10) for all the algorithms and datasets in my experiments, while in [2] the dataset was split into training and testing sets. I also compiled different datasets from SpamAssassin (different years) and used "easy ham" as well as "hard ham" (described in the website[2] as a more spam-like ham, i.e. ham easily confused with spam). There is no information in [2] whether this was included in their tested dataset or not. Besides using SpamAssassin, I also applied the algorithms in other 2 datasets: Enron5 and Enron6. The conclusion of their work was that the bayesian filter performed better than any other algorithm, and that SVM performed better than kNN (among other conclusions that are not relevant for this paper).

In another work [8], similar comparisons were performed. In this work, it was concluded that there were "two more-or-less working spam-filters", SVM and naive bayesian filter.

In further and more recent works such as [6] show results in which SVM performed better than the Bayesian approach, even though in this work SMS were used instead of e-mails.

3. ALGORITHMS

3.1 SVM (linear)

The idea of support vector machines is to find a linear separation boundary

$$w^T x + b = 0$$

(also called an hyperplane) and maximize the margin between categories (in Figure 1, minimize W). There are infinite hyperplanes that could classify the data, but only the one that represents the largest separation, or margin, between the two classes, matters in this case.

$\alpha_{=}(\alpha_1, \alpha_2, ..., \alpha_n)$ so that $\alpha_i >= 0$ for all i and $\sum_{i=1}^{n} \alpha_i c_i = 0$

The the solution alpha that maximizes the problem has to be found, and then the parameters of the optimal hyperplane can be calculated as follows[8]:

$$w_o = \sum_{i=1}^{n} \alpha_i c_i x_i$$

$$b_o = \frac{1}{c_k} - w_o^T x_k$$

where k is an arbitrary index for which $\alpha_k \neq 0$

Once the parameters of this "maximal margin separating hyperplane" are calculated, it is then possible to classify new e-mails by verifying them using the hyperplane equation.

3.2 Naive Bayes

A naive classifier is a classifier that applies the bayes' theorem assuming naively that the value of a particular feature is independent of the value of any other feature, given the class variable. This type of classifier was introduced in early 1960's[5] in text mining fields and it's still widely used for text classification tasks one of them being the one addressed in this paper: spam-filtering.

In this context, words have probabilities of occurring in spam and ham emails. The filter doesn't "know" this probabilities, so they have to be "trained", by using a training sample. After training, the word probabilities of all the words present in the email are used to calculate the probability of an email being ham or spam.

[2]http://csmining.org/index.php/spam-assassin-datasets.html

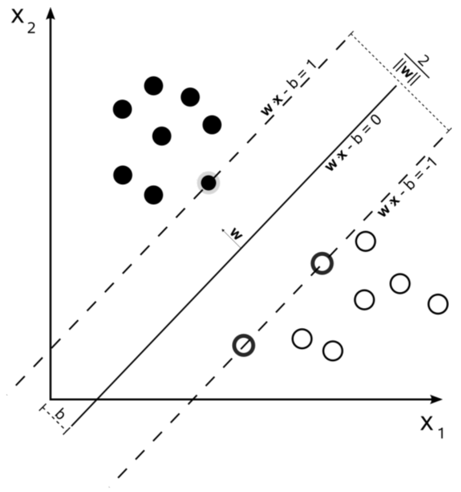

Figure 1: Svm classification example

Each word in the email contributes to the email's spam probability, and the probability that a message containing a given word is spam is done using the Bayes' rule as follows[9]:

$$Pr(S|W) = \frac{Pr(W|S) * Pr(S)}{Pr(W|S) * Pr(S) + Pr(W|H) * Pr(H)}$$

where:

- Pr(S|W) is the probability that a message is a spam, knowing that the some word W is in it;
- Pr(S) is the overall probability that any given message is spam;
- Pr(W|S) is the probability that the word W appears in spam messages;
- Pr(H) is the overall probability that any given message is not spam (is "ham");
- Pr(W|H) is the probability that the word W appears in ham messages.

3.3 k-Nearest Neighbors

In this approach, an email is compared to all the other emails in the training set, being for that reason considered an example-based classifier. When an e-mail has to be classified, the k most similar emails are found (using the distance between the vectors, usually euclidean distance), and if among those k emails there are more spam e-mails than ham e-mails, then the e-mail being classified will be labeled as spam. In Figure 2, if k=3, the green circle would be classified as a red triangle. If k = 5, then the green circle would be classified as a blue square.

One of the problems of this algorithm is that it is what can be considered a memory-based classifier, since all of the training examples are stored in memory[2].

4. EXPERIMENTAL SETUP

To compare the performances between the different algorithms, an integrated environment for machine learning and data mining was used - RapidMiner (https://rapidminer.com/).

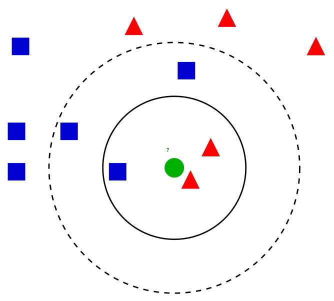

Figure 2: kNN classification example

All the tests and experiments present in this paper, including training, testing and evaluating performance, as well as most of the data pre-processing steps were done using the ”out of the box” features provided by this platform.

4.1 Datasets

As mentioned previously, 3 different datasets were used. The first one from SpamAssassin, was a merge of datasets from 2002, 2003 and 2005. This emails contain ”headers”, as opposed to enron5 and enron6, which have just the subject and content of the emails. Also, the merged SpamAssassin dataset used contains ”hard ham” - ham that includes the use of HTML, unusual HTML markup, coloured text, ”spammish-sounding” phrases etc, and therefore could easily be mistaken as spam. More information about these datasets is shown in Table 1.

4.2 Data pre-processing and Feature vector

Before training and testing each of the classifiers, the data went through some pre-processing steps using several *RapidMiner* tools, sequentially. Those steps were the following:

- Transform Cases - this function simply turns all upper case characters to lower case, since they'd probably have the same meaning and should therefore be considered the same.
- Tokenize - This is a crucial pre-processing step: It splits the emails into tokens (or words) making it possible to use the bag of words method.
- Filter tokens (by length) - Only tokens more than 4 characters long were considered
- Stem (snowball) - This operator changes a word into its ”base word”
- Stopwords - This operator removes common words that have no meaning (only english stopwords were considered)

More detailed information about this operators can be found on RapidMiner documentation[3].

The feature vector was created using TF-IDF (term frequency inverse document frequency). Term Frequency measures how many times a word appears in a document, while IDF measures how important that term is, by considering words that appear more often to be less important than rare words that could more clearly classify an email.

[3]http://docs.rapidminer.com/studio/operators/

4.3 Performance evaluation

In order to evaluate the algorithms used, 3 different measures were used. The first one, Spam/Ham Recall, the second one Spam/ham Precision and finally accuracy.

Spam recall corresponds to the percentage of all spam emails that were classified as spam. As we can see, this measure alone wouldn't be enough, since we could have a 100% spam recall rate if all the emails were classified as spam, even though that wouldn't make our classifier good.

Spam/ham recall is calculated as follows:

$$SR = \frac{\#\ correctly\ classified\ spam/ham}{\#\ of\ all\ spam/ham\ emails}$$

Spam precision indicates the percentage of correctly classified spam of all the e-mails predicted as spam. A low spam precision means that more legit e-mails were misclassified as spam, something that should not happen for a really good classifier in the context of spam filtering. (even though this could be arguable due to its subjectivity, it is more or less consensual that classifying a legit email as spam should be more penalized than classifying a spam one as legit)

Spam/ham precision is calculated as follows:

$$SP = \frac{\#\ correctly\ classified\ spam/ham}{\#\ of\ all\ emails\ classified\ as\ spam/ham}$$

Finally, accuracy is the percentage of the e-mails correctly classified:

$$A = \frac{\#\ of\ correctly\ classsified\ emails}{\#\ of\ all\ emails}$$

5. EXPERIMENTAL RESULTS

Before presenting the results, it is worth noting that it was not possible to execute the k-NN algorithm with the original datasets due to insufficient memory (consequence of the nature of the algorithm, as explained section 3.3). For that reason, randomly selected samples of the original SpamAssassin dataset were created, originating new datasets containing 100/200/500 spam emails and 100/200/500 ham emails. Between these 3 different datasets, k-NN performed better with the largest one (total of 1000 emails).

Furthermore, k-NN was also tested for values of k of 1, 3 and 5. For all dataset sizes, k=1 was steadily the option that achieved better results.

For these reasons, the kNN performance results that were used for the Spam Assassin comparisons (Table 1) were obtained with k=1 and a sample of 1000 emails of the original dataset.

As we can see by looking at Table 2, the best accuracy was obtained using the Naive Bayes approach, followed by kNN and then SVM. However, the SVM approach had a slightly better spam precision that naive bayes and kNN, which means that less legit e-mails were misclassified as spam. This could be a very important outcome in some situations, as explained before. On the other hand, despite the slightly lower spam precision, Naive Bayes achieved a much better Spam Recall (almost 14% higher) - this means that more spam e-mails were actually caught by our filter, which is also a desirable feature of our classifier. Whether one is superior to the other is subjective and would depend on the context itself.

Table 1: Datasets spam and ham messages

Dataset Name	# of spam emails	# of ham emails	# Total emails	Presence of headers
SpamAssassin	2399	6952	9351	Yes
Enron5	3675	1500	5171	No
Enron6	4500	1500	6000	No

Table 2: Results with Spam Assassin

Method	SP	SR	HP	HR	A
Naive Bayes	99.52	95.21	98.37	99.84	98.65
SVM	99.64	81.41	93.97	99.90	95.16
kNN	98	98.2	98.2	98	98.1

Table 3: Results with Enron5

Method	SP	SR	HP	HR	A
Naive Bayes	98.36	94.67	88.03	96.13	95.09
SVM	95.67	99.70	99.18	88.93	96.58
kNN	97.67	84	85.96	98	91

Looking at Table 3, the results were slightly different. The naive bayes approach performed worse with this dataset in every measure, as well as the kNN algorithm. The SVM approach, behaved differently. It achieved a slightly better accuracy, dropped heavily on ”ham recall”, with only 88.93% against the 99.90% achieved with spam assassin, but improved spam recall by almost 20%. This means than more than 1 in every 10 legit emails would be discarded as spam by our filter. At the same time, spam emails would be more frequently detected. This indicates that this filter, which in SpamAssassin was the most strict filter (strict in the sense that it would classify an email as spam only when almost sure it was actually spam), turned out to be the less strict using a different dataset.

Finally, analyzing Table 4, it is possible to see that the results were similar to the ones presented in Table 3. Accuracy dropped even more in general, as well as the all the performance measures, with the exception of the SVM approach. Again, this approach followed the tendency explained before, and it’s spam recall improved even more, being now close to 100% (99.96%). At the same time, Ham Recall dropped even more, along with its spam precision.

Probably the most interesting fact was that both Spam Precision and Ham Recall dropped from SpamAssassin to Enron5, and then from Enron5 to Enronr6. This could be due to the fact that this dataset didn’t contain headers which could maybe tell relevant information for the classification of the emails. It could also be related to the fact that ”hard ham” was used in the former case, training the classifier better for difficult cases of ham that look like spam. The reason for Enron6 having performed worse than Enron5 (given that neither of these datasets contained ”headers” or ”hard ham”) could possibly be that Enron6 dataset was smaller.

Table 4: Results with Enron6

Method	SP	SR	HP	HR	A
Naive Bayes	98.13	93.49	82.90	94.67	93.78
SVM	93.32	99.96	99.83	78.53	94.60
kNN	96.47	82	84.35	97	89.5

6. CONCLUSIONS

In this paper, 3 machine learning algorithms were compared in 3 different datasets.

In terms of datasets, it was possible to conclude that the dataset played an important role in the performance of the classifiers.

According to these experiments, the use of ”hard ham” improved the performance of our filters, as well as considering the headers of the emails in the training and classification phases.

The size of the datasets also seemed to have some impact on the performance of the filters, being that for larger datasets the accuracy was better.

As for the algorithms, the results were rather inconclusive. SVM was the one to obtain the maximum spam precision and Ham Recall for the SpamAssassin dataset, but the worst for the same measures for the other 2 datasets (Enron5 and Enron6). Bayesian approach proved to be a stable option, and achieved the best accuracy of all the algorithms when using the Spam Assasssin dataset. kNN performed better in Spam Assassin, but lost considerable performance with the other 2 datasets.

7. REFERENCES

[1] Kaspersky security bulletin. spam and phishing in 2015.

[2] W. Awad and S. ELseuofi. Machine learning methods for spam e-mail classification. *International Journal of Computer Science information Technology*, 3(1), February 2011.

[3] T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[4] S. Radicati and Q. Hoang. Email statistics report, 2012-2016. the radicati group. *Inc., London*, 2012.

[5] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003.

[6] H. Shirani-Mehr. Sms spam detection using machine learning approach, 2013.

[7] The European Commission. *Data protection: ”Junk” email costs internet users 10 billion a year worldwide Commission study.* The European Commission, 2001.

[8] K. Tretyakov. Machine learning techniques in spam filtering. In *Data Mining Problem-oriented Seminar, MTAT*, volume 3, pages 60–79. Citeseer, 2004.

[9] Wikipedia. Naive Bayes spam filtering — Wikipedia, the free encyclopedia. https: //en.wikipedia.org/wiki/Naive_Bayes_spam_filtering, 2016. [Online; accessed 16-May-2016].

[10] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.