Email Spam Filter in Spark

COEN 242 Project Final Report- Team 1

Ming Ming, Stefan Zecevic, Vincent Tai, Wenqi Zhou.

Santa Clara University, Computer Engineering.

Audience

This is a paper report for the SCU COEN242 final project. We will demonstrate what the goal of project is and what techniques we used to achieve that goal. The readers do not necessarily need to have specific knowledge on Big Data to read the paper, but knowing the general concept of distributed computing or knowing Spark-like system will help for understanding some of the details in this paper.

Table of Contents

Introduction

Spam is an ever-increasing problem; since e-mail is very easy and cheap to send, it has not only become the method used by friends and colleagues to exchange messages, but also a medium for conducting electronic commerce. One study shows that in the total count of all emails on the network, over 50% of the emails are spam [1]. How to battle spam email is an essential issue for individuals and companies. In the Design section, we will demonstrate how do we chose the dataset, and how we prepare the dataset for the implementation of our project. In the implementation section, we will show the flow chart of our program and explain the infrastructure and algorithm in details. The evaluation section will contains the latest running data we have collected from our program in terms of performance and accuracy.

Design

Enron Email Data set

Searching for an appropriate email data set was not easy. Due to privacy concerns, most available email sets are simulated or the only consist of spam emails. In order to build an effective filter, we needed the data set that has a real world mixture of spam emails and legitimate emails. The Enron forensic data set fits this criteria. [2]

The data set consists of the Enron email collected in the discovery process of the Enron legal proceedings. This means that each email has been preserved. Although attachments have been removed to save space, the contents of the email body have been perfectly preserved. Because this is a forensics data set, legitimate emails coexist with spam emails. Some emails have been removed on users requests for privacy reasons, but this data set is by far the most representative of a real data set, particularly in a corporate setting where spam filtering is considered more valuable.

The data set itself is organized very simply. Each user has its own folder containing all of that uses emails in their system at the time of legal discovery. Inside each user's folder is there folder hierarchy. While the standard default folders such as inbox, outbox, and trash are there, what's more interesting is that each user has their own personal folders. Each email is stored as a plain text file. Each file itself contains raw source, including any headers and metadata.

While some spam filters may use email headers in order to analyze how many copies of an email was sent or the time an email was received, or email filter was clearly not going to be that sophisticated. Most examples that we have seen in class regarding the data analysis take advantage of raw data that has been preformatted for the purpose. In real life, particularly in a real-life data set such as the Enron email dump, data needs to be manipulated and converted explicitly for the purpose of analysis.

Naive Bayes

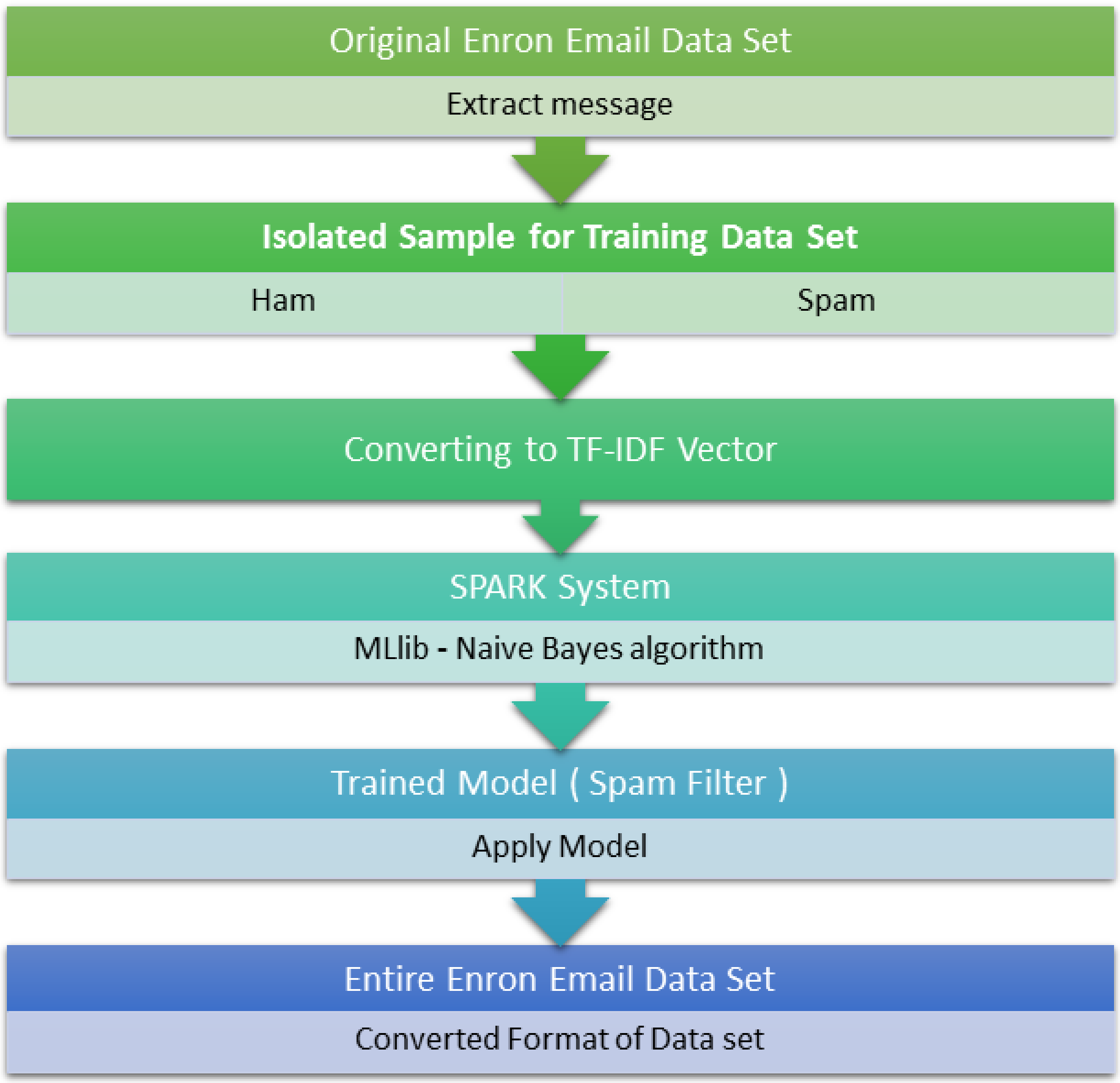
Naive Bayes classifiers work by correlating the use of tokens (typically words of the messages), with spam and non-spam e-mails and then using Bayesian inference to calculate a probability that an email is or is not spam. Naïve Bayes based filter are simple way to clarify text and are easy to train efficiently and parallelize. It’s a high intelligent linear clarifier which evolve with Spam.

Prepare training email data set

In order to create the training email data set, we randomly selected 1024 e-mails from the Enron data set, and sorted them into two categories, “ham” and “spam”. The ham dataset has all the legitimate e-mails, and the spam set all the spam e-mails.

To create a spam filter for a financial company like Enron, a list of key words like “sales”, ”offer” and “mortgage” will be less efficient for identifying the spam, because those words are involved in Enron’s legitimate emails in the daily base. Secondly, there are many casual chat mails in the data set which contains lots of words that would consider to be spam key words. Also, there are emails which is sent by Blind carbon copy to the Enron’s employee. All of specific factors complicate the process of creating training data set. [3]

Implementation


Original Enron Email Data Set
Extract message
Isolated Sample for Training Data Set
Ham
Spam
Converting to TF-IDF Vector
SPARK System
MLlib - Naive Bayes algorithm
Trained Model (Spam Filter)
Apply Model
Entire Enron Email Data Set
Converted Format of Data set

Implementation

Our project is conducting by several phased which shows in following figure. The original data set is simply organized, we extract the message and build an isolated sample training data set. The training data set will convert to TF-IDF Vector, and get tokenized. The MLlib Naive Bayes Algorithm will take the training data set and generated the model which will then apply to the whole data set.

The implementation was done in Scala, and is based on samples found in “Machine Learning with Spark”. The initial RDD is created by using the “wholeTextFiles” method, which creates an RDD out of all the files in a given directory. The RDD then undergoes transformations to produce TF-IDF vectors, which get processed using Spark’s Naive Bayes implementation to produce the final result.

Naive Bayes Algorithm

Naive Bayes is a conditional probability model that was formulated by Thomas Bayes, it is quite simple yet powerful; it can be written down in simple words as follows:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

The statistical filtering sketch is: start with one corpus of spam and one of non-spam mail, each one has more than 1000 messages. Scan the corpuses and consider alphanumeric characters, dashes, apostrophes, and dollar signs to be tokens, establish 2 hash table, each maps the tokens to their occurrence. Then create the third hash table which map each token to the probability that an email containing it is a spam. When new mails arrives, they are scanned into tokens and the combined probability are calculated based on tokens.

The advantage of the Bayesian method is that it considers the most interesting words (as defined by their deviation from the mean) and comes up with a probability that a message is spam. In this way it take the whole message into account. It won't simply kick the email which contains "cash" and "free" into junk mailbox, but will calculate probability of "cash" and "free" as well as other words in this mail. In other words, Bayesian filtering is a much more intelligent approach because it examines all aspects of a message, as opposed to keyword checking that classifies a mail as spam on the basis of a single word.

The problem of other content based anti-spam filters is, spams senders learn how to avoid words which tend to be blocked during time. Spam are like pest who evolve to resist pesticide. Whereas filter based on Naïve Bayes evolves with spam. If spam start to use "c1ick" instead of "click", Naïve Bayesian filter can add that word into list automatically by training its model.

Spark has a Naïve Bayes clarification API which make implementation of spam filter much easier. MLlib supports multinomial naive Bayes, which is typically used for document classification. Within that context, each observation is a document and each feature represents a term whose value is the frequency of the term. [4]

Results: Evaluation

Two experiments were run: one on accuracy and one on performance.

To test accuracy, we used another e-mail set, TREC. This set of e-mails is pre-sorted into ham and spam, thus allowing accuracy to be determined; the Enron mail was not pre-sorted, so we could not determine accuracy of our results.

1000 e-mails from TREC were selected: 500 spam and 500 ham. This smaller set was then divided into two parts: training and test data. Running the Scala program on this set resulted in a 13.8% false negative rate (spam marked as ham) and 1.6% false positive rate (ham marked as spam).

To test performance, we used our manually-sorted 1024 e-mail set as training. We could not use the entire data set, due to limitations of the SCU Hadoop cluster, which has a 50000 file limit. Instead, we randomly selected 48000 files from the set and added them to the HDFS.

The experiment was done once on the SCU Hadoop cluster and once on a laptop computer.

The final step of the computation took 233 seconds on the laptop and 109 seconds on the SCU cluster. It would be expected that the cluster computation would be faster than that, instead of only by a factor of 2. It is possible that the bottleneck was having many file I/O operations of HDFS.

References

[1] GFI, "Why Bayesian filtering is the most effective anti-spam technology," GFI.

[2] W. W. Cohen, "Enron Email Dataset," MIT, SRI, and CMU, 2015. [Online].

[3] P. Graham, "Better Bayesian Filtering," January 2003. [Online]. Available: http://www.paulgraham.com/better.html.

[4] "MLlib - Naive Bayes," 2015. [Online]. Available: https://spark.apache.org/docs/latest/mllib-naive-bayes.html.

[5] "Spark Documentation," 2015. [Online]. Available: https://spark.apache.org/docs/latest/.

[6] "Spark Programming Guide," 2015. [Online]. Available: https://spark.apache.org/docs/latest/programming-guide.html.

[7] B. Satterfield, "Ten Spam-Filtering Methods Explained," 30 November 2006. [Online]. Available: http://www.techsoupcanada.ca/learning_center/10_sfm_explained.

[8] K. Rubin, "The Ultimate List of Email SPAM Trigger Words," 11 January 2012 . [Online]. Available: https://blog.hubspot.com/blog/tabid/6307/bid/30684/The-Ultimate-List-of-Email-SPAM-Trigger-Words.aspx.

[9] P. Seibel, "Practical: A Spam Filter," 2005. [Online]. Available: http://www.gigamonkeys.com/book/practical-a-spam-filter.html.

References

[10] N. Pentreath, Machine Learning with Spark, Birmingham, UK: Packt Publishing Ltd., 2015.

[11] M. Sahami, S. Dumais, D. Heckerman and E. Horivitz, "A Bayesian approach to Filtering Junk E-Mail," 1999.