ISSN No. 0976-5697

Volume 8, No. 1, Jan-Feb 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Spam: A Big Data Challenge

Liny Varghese
Cochin University of Science and Technology
Kochi, India

Supriya M.H
Cochin University of Science and Technology
Kochi, India

K Poulose Jacob
Cochin University of Science and Technology
Kochi, India

Abstract: Spam consists of varieties of contents like text, image, embedded HTML, MIME attachments and also the volume of spam mails sent per day is massive. To handle this high volume, high velocity and large varieties of spam, a scalable spam filtering solution is required. Scalable solutions available for machine learning and statistical studies can be used to implement a scalable solution for spam filtering also. From Big data Analytics domain, Mahout is an open source library from Apache for building scalable solutions in machine learning. This paper uses mahout framework to analyse the time and accuracy efficiencies of the results of two Naïve Bayes classification algorithms.

Keywords: Apache Mahout, big data, scalable algorithms, Naïve Bayes algorithms

I. APACHE MAHOUT

Apache –Mahout is a set of scalable algorithms to carry out the clustering and classification in big data arena problem free[1][2]. Mahout is used as a machine learning tool when the collection of data to be processed is very large, or too large for a single machine[3]. Mahout algorithms are written in Java, and some portions are built upon Apache's Hadoop distributed computation project[4]. It doesn't provide a user interface; but a framework of tools intended to be used and adapted by developers[5].

II. MAHOUT IN CLASSIFICATION

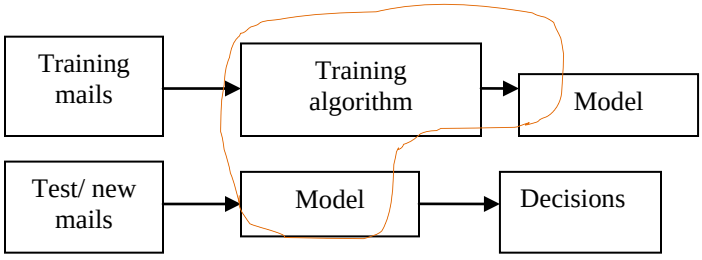

Figure 1: Classification systems

In the book [6] , the authors explain how mahout can be used to build and personalize effective classifiers. Different data mining and machine learning models are explained with examples. The book discusses classification and its applications and what algorithms and classifier evaluation techniques are supported by Mahout. The paper[7] compares k-means and fuzzy c-means for clustering a noisy realistic and big dataset. They made the comparison using a free cloud computing solution Apache Mahout/ Hadoop and Wikipedia's latest articles. And the authors claim that in a noisy dataset, fuzzy c-means can lead to worse cluster quality than k-means. They concluded that Mahout is a promise clustering technology but is premature. The study[8] uses Apache Mahout for Collaborative Filtering and conclude that it is a mature framework for building recommenders, still a lot of room for improvements and extensions. An ideal situation to evaluate an e-commerce recommender systems, the study[9] suggests to find an open-source platform with many active contributors that provides a rich and varied set of recommender system functions that meets all or most of the baseline development requirements

III. METHODOLOGY

This study discusses on how to choose and extract features effectively to build a Mahout classifier, how these extracted features are used for creating a model to test the new incoming mails. The steps in methodology are explained below.

a. Extracting features to build a Mahout classifier

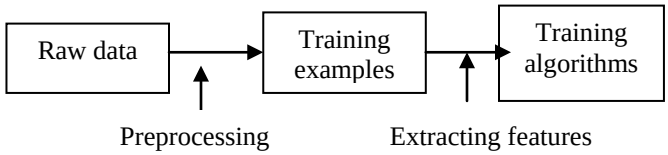

Figure 2: extracting features

Getting data into a form usable by a classifier is a complex and often time-consuming step[10]. Preparing data for the training algorithm consists of two main steps:

1. Preprocessing raw data: Raw data is rearranged into records with identical fields. In spam filtering context, the data are words.
2. Converting data to vectors: Classifiable data is parsed and vectorized using custom code or tools such as Lucene analyzers and Mahout Vector encoders. Some Mahout classifiers also include vectorization code.

b. Preprocessing raw data into classifiable data :

The first phase of feature extraction involves rethinking the data and identifying features in mails to use as predictor variables. Here the header and body parts of the mails are used to extract the features in preprocessing task.

c. Transforming raw data

Once the features are identified, they must be converted into a format that's classifiable. This involves rearranging the data into a single location and transforming it into an appropriate and consistent form. Each record contains the fully de-normalized description of one training example.

d. Classifying Spam mails

Classification models for the spam using the learning algorithms Naïve bias and complement Naïve bias are built based on the spam data set. These models are applied to new set of test data and the efficiency is computed and compared.

a) Data set pre-processing

The first step in preparing a data set is to examine the data and decide which features might be useful in classifying spam. To begin, download Enron data set from this URL: http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html

Dataset	Number of mails ineach dataset	
	spam	ham
Enron 1	1500	3672
Enron 2	1496	4361
Enron 3	1500	4012
Enron 4	4500	1500
Enron 5	3675	1500
Enron 6	4500	1500
Total	33716	

Table 1: Enron Dataset –Distribution of spam and ham mails

The Enron data set consists of one mail per file. Each file begins with header lines that specify things such as: who sent the message, how long it is, what kind of software was used, and the subject. The predictor features in this kind of data are either in the headers or in the message body. A natural step when first examining this kind of data is to count the number of times different header fields are used across all emails. This helps determine which ones are most common and thus are likely to affect our classification of emails.

b) Choosing an algorithm to train the classifier

The main advantage of Mahout is its robust handling of extremely large and growing data sets. The algorithms in Mahout all share scalability, but they differ from each other in other characteristics. This study uses Naive Bayes and complement Naive Bayes as the classifiers. The Naive Bayes and complementary Naive Bayes algorithms in Mahout are parallelized algorithms that can be applied to larger data sets because they can work effectively on multiple machines at once. The Mahout implementation of naive Bayes, however, is restricted to classification based on a single text-like variable which is apt for spam problem since spam contains only words or text.

B. Classifying Enron spam data with naive Bayes

The data extraction step is applied to get the data ready for the training and then the model is got trained. Once that's done, the process of evaluating initial model is started to determine whether it is performing well or changes need to be made.

1) Data extraction for naive Bayes

First get the spam and legitimate mails into a classifiable form and convert it to a file format for use with the Naïve Bayes algorithm. The Naïve Bayes classifier's parser creates a file with each line contains the value of the target variable followed by space-delimited features, where a 1 indicates the presence of the feature name and 0 indicates absence. Each directory is scanned and each file is transformed into a single line of text that starts with the directory name and then contains all the words in the email.

2) Training the naive Bayes classifier

In this step, the Naïve Bayes classification model is trained with the training and test data converted in right format. The resulted model is stored in a directory and the model consisted of several files that contain the components of the model. These files were in binary format and used to classify the test data.

3) Testing with naive Bayes model

To evaluate the performance of newly trained model, naive Bayes model is run on the test data. The test program produced the following output in Table 2. The summary has raw counts of how many emails were classified correctly or incorrectly.

Summary		
Correctly Classified Instances	1425	99.234%
Incorrectly Classified Instances	11	0.766%
Total Classified Instances		1436

Table 2: Classifying the Enron 2 dataset with 25% split using Naïve Bayes Model

In this testing, the naive Bayes model is performing well, with a score of nearly 93% correct. The program also produced the following confusion matrix.

Confusion Matrix

```
----------------
a       b       <--Classified as
1040    6       | 1046   a    = ham
5       35      | 390    b    = spam
```

Statistics

```
-----------------
Kappa           0.9739
Accuracy        99.234%
```

Reliability 66.041%
Reliability 0.572

Figure 3: Confusion Matrix and Statistics using Naïve Bayes model

C. *Classifying with complement naïve bayes classifier*

In this step, the Complement Naïve Bayes classification model is trained with the training dataset as in the case of Naïve Bayes. A new model with complement Naïve Bayes algorithm is generated and this model is used to classify the test data.

1) Testing with complement naïve bayes classifier

To classify the mails in test dataset, the newly trained is model is run with the test data. The test program produced the following output as shown in Table 3. The summary has raw counts of how many emails were classified correctly or incorrectly.

Summary		
Correctly Classified Instances	2394	93.0431%
Incorrectly Classified Instances	179	6.9569%
Total Classified Instances		2573

Table 3: Classifying the dataset Enron 1 with 50% split using Complement Naïve Bayes

In this testing, the Complimentary Naive Bayes model is performing well, with a score of nearly 93% correct. The program also produced the following confusion matrix.

Confusion Matrix

a b <--Classified as
132 2 | 134 a = ham
177 562 | 739 b = spam

Statistics

Kappa 0.143
Accuracy 93.0431%
Reliability 5.6466%
Reliability (standard deviation) 0.5217

Figure 7: Confusion Matrix and Statistics using Complement Naïve Bayes

D. *Performance Evaluation*

1) Time complexity

The time taken for testing of different partitions of training set and test set are given in the Table -4. The algorithms took almost same amount of time even in different sample sizes of training and test data set.

Dataset	Time taken for testing at different Splits of dataset (time in milli seconds)			
	25%	50%	75%	99%
Enron 1	2014	2083	3049	3057
Enron 2	2020	3086	3128	3140
Enron 3	2059	3091	3087	3143
Enron 4	2019	3091	3116	3184
Enron 5	2042	2068	3120	3151
Enron 6	1995	3050	3135	3129

Table 4: Time taken for Complimentary Naïve Bayes algorithm

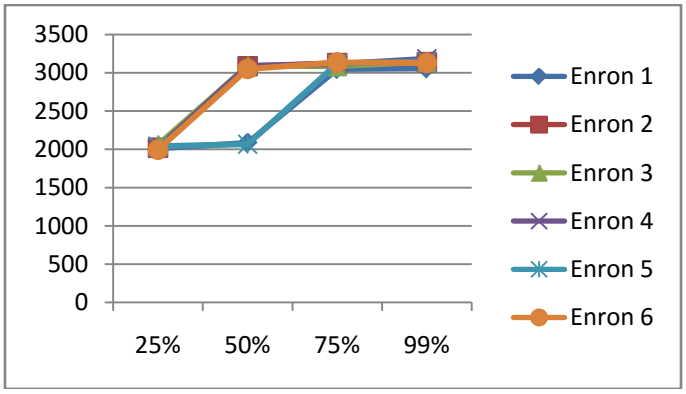

Figure 4: Time taken for Complimentary Naïve Bayes algorithm

Classifying mails with Mahout shows that increasing the number of mails in training set and test set do not increase the time complexity even linearly as shown in the table. The results of Naïve Bayes and complementary Naïve Bayes prove this statement.

Dataset	Time taken for testing at different Splits of dataset (time in milli seconds)			
	25%	50%	75%	99%
Enron 1	92255	92255	93210	93350
Enron 2	92194	93219	93284	93235
Enron 3	92191	93181	93297	94338
Enron 4	92198	93248	93237	93302
Enron 5	92161	92406	93291	93194
Enron 6	92142	93365	93298	93289

Table 5: Time taken for Naïve Bayes algorithm

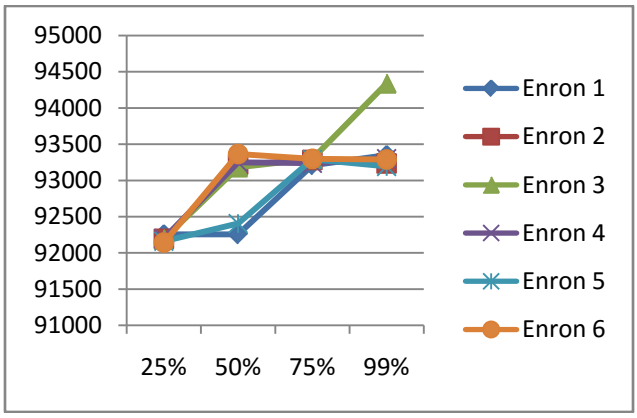

Figure 5: Time taken for Naïve Bayes algorithm

2) Accuracy

As presented in Table 6 the accuracy of classification is almost same and high for all the split-ups. This shows that the algorithms in Mahout are designed to work robustly and reliable in any size of datasets.

Dataset	Accuracy for testing at different Splits of dataset			
	25%	50%	75%	99%
Enron 1	95.02%	93.04%	95.76%	87.5%
Enron 2	99.24%	98.9%	98.47%	82.49%
Enron 3	98.41%	98.73%	96.55%	77.96%
Enron 4	81.50%	89.22%	95.71%	86.2%
Enron 5	96.59%	97.55%	98.76%	92.06%
Enron 6	81.39%	87.03%	93.44%	80.29%

Table 6: Accuracy of testing using Complimentary Naïve Bayes algorithm

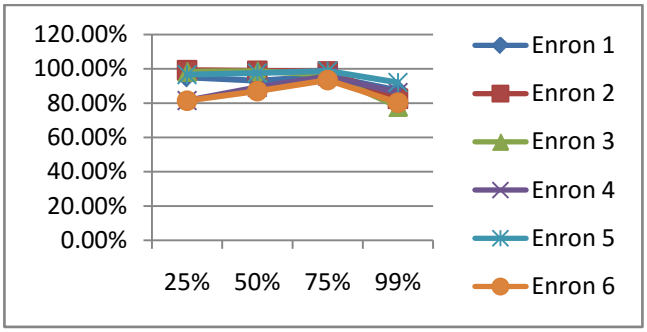

Figure 6: Accuracy of testing using Complimentary Naïve Bayes algorithm

Dataset	Accuracy for testing at different Splits of dataset			
	25%	50%	75%	99%
Enron 1	98.1395	97.920	97.0226	88.2099
Enron 2	99.234	99.1456	98.6725	72.913
Enron 3	99.0566	99.0345	98.4672	74.642
Enron 4	98.6622	98.0451	96.0071	27.729
Enron 5	99.3039	99.243	98.7394	62.139
Enron 6	98.2456	97.1765	92.7236	46.54

Table 7: Accuracy of testing using Naïve Bayes algorithm

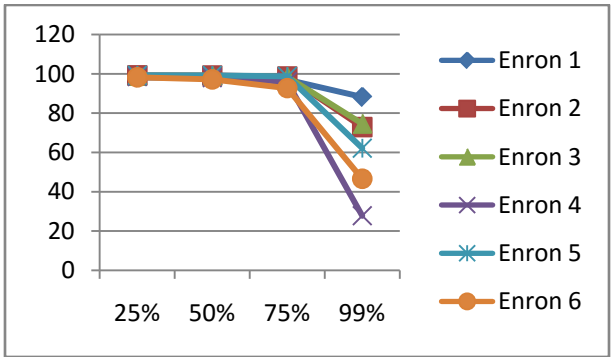

Figure 7: Accuracy of testing using Naïve Bayes algorithm

E. ***Summary***

The reason Mahout has an advantage with larger data sets is that as input data increases; the time or memory requirements for training may not increase linearly as in a non-scalable system. The classification algorithms in Mahout require resources that increase not faster than the number of training or test examples, and in most cases the computing resources required can be parallelized. This allows to trade off the number of computers used against the time the problem takes to solve.

If the training samples are more than ten million and the predictor variable is a single, text-like value, naive Bayes or complement naive Bayes may be the best choice of algorithm. Naive Bayes algorithms are best choice for data with more than 100,000 training examples. The amount of time taken for classification does not linearly depend on the number of training data.

F. ***References***

[1] G. Ingersoll, "Introducing apache mahout," *IBM developerWorks Technical Library*. 2009.

[2] P. Giacomelli, *Apache mahout cookbook*. Packt Publishing Ltd, 2013.

[3] G. Ingersoll, "Introducing Apache Mahout Scalable, commercial-friendly machine learning for building intelligent applications," *IBM Corporation*. 2009.

[4] A. Mahout, "Scalable machine learning and data mining," *2013-4-24. http://mahout. apache, org*. 2012.

[5] A. Mahout, "Scalable machine-learning and data-mining library," *available at mahout. apache. org*. 2008.

[6] A. Gupta, *Learning Apache Mahout Classification*. Packt Publishing Ltd, 2015.

[7] C. Rong and others, "Using mahout for clustering wikipedia's latest articles: A comparison between k-means and fuzzy c-means in the cloud," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 565–569.

[8] S. Schelter and S. Owen, "Collaborative filtering with Apache Mahout," *Proc. ACM RecSys Chall.*, vol. i, no. September 2012, pp. 1–13, 2012.

[9] S. G. Walunj and K. Sadafale, "An online recommendation system for e-commerce based on apache mahout framework," in *Proceedings of the 2013 annual conference on Computers and people research*, 2013, pp. 153–158.

[10] S. OWEN, R. ANIL, T. DUNNING, and E. FRIEDMAN, *Mahout in Action*. Manning Publications Co. Greenwich, CT, USA, 2011.