

Issued on: April 20, 2014

Due by Wednesday 11:25PM, April 30, 2014

Problem 1. An illustration of Bayes' Theorem. Imagine that you have two baskets with apples. The first basket has 3 red apples and 7 yellow apples. The second basket has 4 red apples and 2 yellow apples. If you have picked a red apple, what is the probability that you picked it up from the first basket. NOTE: Solve this problem using pen and paper and then provide a brief MS Word explanation about how you did it. You are not asked to write any computer programs.

Problem 2. Please reproduce results for Mahout Recommender as described in lecture. Once you extract recommendations for several users, please go into the ratings.csv file and lower ratings for one of the movies recommended to one of the users. You can use Vi to reduce several scores for a selected movie from 5 to 4 or 3. Examine whether change of scores affects the results of the recommender.

Problem 3. The purpose of the following example is to demonstrate effectiveness of Mahout Naïve Bayes classifier. Make sure that the Linux user running the following command has HADOOP_HOME and MAHOUT_HOME environmental variables defined. This example was verified on CentOS6.5 MRv1 VM with Mahout 0.9.

1. Download Spam Assassin corpus from <http://spamassassin.apache.org/publiccorpus> . We will use that corpus both to train the classifier and to test how well the classifier performs at detecting spam. Create a new corpus directory and within it a spam-assassin directory.
2. Once you expand the downloaded archive with spam and ham emails, place expanded files in corpus/spam-assassin/spam and corpus/spam-assassin/easy-ham files respectively. Count the number of spam and the number of ham files. Examine a few files in each directory. Those, as you can see are real emails.
3. Next we want to create a training set of files containing both spam and ham emails. You are welcome to work with the entire set of emails. In order to reduce the execution time of the initial test runs, we recommend that you place only some 400 ham emails and some 100 spam emails in the training set. Those two sets of files should be placed in the directories:
 - a. corpus/spam-assassin/train/spam
 - b. corpus/spam-assassin/train/easy_ham
4. Subsequently, in HDFS, create directories:
 - a. train/spam
 - b. train/easy_ham

Transfer all files from the previous two Linux directories (starting with corpus/spam-assassin) into respective directories in HDFS.

5. Next we need to transform all of those HDFS files into Hadoop sequence files. Hadoop sequence file is a flat file consisting of binary key/value pairs. It is extensively used in [MapReduce](#) as input/output format. It is also worth noting that, internally, the temporary outputs of maps are stored using `SequenceFile` objects. We will examine the file(s) once generated and see (their) internal structure. Transformation of files in HDFS directory `train` (please note that we include both subdirectories `spam` and `easy_ham`) is accomplished using the following command:

```
$ mahout seqdirectory -i train -o train_mahout -c UTF-8
```

Option `-i` points to the input directory; option `-o` to the output directory and `-c` to the character set of the resulting files. Please issue command:

```
$ mahout seqdirectory --help
```

and record the output. As you will see, Mahout's `seqdirectory` program has a bunch of sensible and useful options.

6. We are ready to examine the content of the output directory `train_mahout`. Please report on how many files are there in that directory? Why is the number of those files whatever it is?
7. Transfer the file(s) to Linux directory and examine them/it with `vi` or some other text processing tool. Can you tell us what are the `keys` and what the `values` we spoke above in the context of Hadoop sequence files.
8. Please, also examine the sequence file(s) with Hadoop's `fs -text` tool by issuing command:

```
$ hadoop fs -text | head -500
```

The output will be very similar, yet somewhat different from the one you saw when examining the local copy with `vi`. Please explain how observed or utilized key-value pairs preserve the information content of the original `train` directory.

9. Mahout classifier needs to transform our textual files (emails) into sparse vectors in the multi-dimensional vector space of all words (terms). In class, we spoke on a couple of occasions about how in natural language processing we need to remove all stop words, stem different word forms and then assign to every text document weight, i.e. coordinate in terms of so called ***Tf-Idf*** coefficients. Mahout classifier programs will do all of that. Mahout also allows you to select which type of distance in the above vector space to use when assessing similarity or proximity of text documents (vector points). To create all of those linguistic artifacts we run Mahout program `seq2sparse`, all on one line:

```
$ mahout seq2sparse -i train_mahout -o train_vectors -lnorm
-nv -wt tfidf
```

In the above statement we used the following options:

- a. The `-lnorm` parameter which instructs the classifier to use the `L_2` norm (a fancy Mathematical name for Euclidian distance)
- b. The `-nv` parameter is an optional parameter that instructs the classifier to output vectors as `NamedVectors`
- c. The `-wt` parameter instructs which weight function needs to be used. We chose previously mentioned ***Tfidf*** weighing of documents. When we are at it could you tell us what is the typical formula for *Tfidf* coefficients.
- d. Options `-i` and `-o` respectively specify input and output HDFS directories.

10. Please, examine resulting HDFS directory `train_vectors`. Tell us what do you see? Please, examine the contents of individual directories and files. Perhaps you could tell us what is the meaning of individual files and what type of information they contain.

11. Now that we have generated the weight vectors, we need to pass them to the training algorithm. However, if we train the classifier against the whole set of data, we will not be able to test the accuracy of the classifier. To avoid this, we need to divide the vector files into two sets called the 80-20 split. This is a good data-mining approach to divide the whole bunch of data into two sets: one for training and one for testing the algorithm. A good dividing percentage is shown to be 80 percent and 20 percent, meaning that the training data should be 80% of the total while the testing data should be the remaining 20%. To split data, we use the following command, all on one line:

```
$ mahout split -i train_vectors/tfidf-vectors
--trainingOutput train-vector-set
--testOutput test-vector-set --randomSelectionPct 40
--overwrite --sequenceFiles -xm sequential
```

Examine and report on the resulting HDFS directory and files.

12. As result of `split` command, we will have two new folders: `train-vector-set` and `test-vector-set` containing the training and testing vectors. Please invoke program `split` with `--help` option and tell us what is the meaning of options we are using.

13. Now, it is time to train Mahout Naïves Bayes algorithm on the training set of vectors, and for that purpose we use Mahout program `trainnb` (all on one line) :

```
$ mahout trainnb -i train-vector-set -el -o model -li
labelindex -ow
```

Please, examine resulting folders and files. Not all of them are human readable. Please, tell us as much as you can about their contents and purpose.

14. Once program `trainnb` is finished, we are ready to test it against the remaining 20% of the initial input vectors. We perform the test using Mahout program `testnb`, invoking the following command, all on one line:

```
$ mahout testnb -i test-vector-set -m model -l labelindex\
-ow -o testing
```

Please, examine the result and please tell us what is the success rate with which Mahout Naïve Bayes classifier identifies spam emails? What is the name of the final output of `testnb` program?

NOTE: This rightly appears too contrived and too involved. However, once you get used to all of those commands, you can place them in a single script and run them on a single line. If you have experience with Mahout, you certainly know that all of this could be done programmatically from a Java program, using Mahout Java class libraries. If you find it easier to use Java libraries, you are welcome to do so.

Problem 4. Please, look into your Inbox and find a few, 4 or 5, junk emails or emails that are or look like spam. Please use Mahout Naïve Bayes classifier to tell you whether those emails are or are not spam.

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of your command. We cannot retype text that is in JPG images. Please, always submit a copy of original, working scripts and class files you used as separate files. Sometimes we need to run your code and retyping is to costly. Please, do not provide complete outputs of processes such as Map Reduce, unless you believe they contain essential information supporting your assertions. Please, submit to the class drop box. For issues and comments visit the class Discussion Board or send emails to cscie63@fas.harvard.edu, please.