Using the Mahout Naive Bayes Classifier to automatically classify Twitter messages

2013/03/13 118 COMMENTS (HTTPS://CHIMPLER.WORDPRESS.COM/2013/03/13/USING-THE-MAHOUT-NAIVE-BAYES-CLASSIFIER-TO-AUTOMATICALLY-CLASSIFY-TWITTER-MESSAGES/#COMMENTS)

Classification algorithms can be used to automatically classify documents, images, implement spam filters and in many other domains. In this tutorial we are going to use Mahout (http://mahout.apache.org/) to classify tweets using the Naive Bayes Classifier (http://en.wikipedia.org/wiki/Naive_Bayes_classifier). The algorithm works by using a training set which is a set of documents already associated to a category. Using this set, the classifier determines for each word, the probability that it makes a document belong to each of the considered categories. To compute the probability that a document belongs to a category, it multiplies together the individual probability of each of its word in this category. The category with the highest probability is the one the document is most likely to belong to.

To get more details on how the Naive Bayes Classifier is implemented, you can look at the mahout wiki page (https://cwiki.apache.org/MAHOUT/bayesian.html).

This tutorial will give you a step-by-step description on how to create a training set, train the Naive Bayes classifier and then use it to classify new tweets.

Requirement

For this tutorial, you would need:

- jdk >= 1.6
- maven
- hadoop (preferably 1.1.1)
- mahout >= 0.7

To install hadoop and mahout, you can follow the steps described on a previous post (https://chimpler.wordpress.com/2013/02/20/playing-with-the-mahout-recommendation-engine-on-a-hadoop-cluster/) that shows how to use the mahout recommender.

When you are done installing hadoop and mahout, make sure you set them in your PATH so you can easily call them:

```
export PATH=$PATH:[HADOOP_DIR]/bin:$PATH:[MAHOUT_DIR]/bin
```

In our tutorial, we will limit the tweets to deals by getting the tweets containing the hashtags #deal, #deals and #discount. We will classify them in the following categories:

- apparel (clothes, shoes, watches, …)
- art (Book, DVD, Music, …)
- camera
- event (travel, concert, …)
- health (beauty, spa, …)
- home (kitchen, furniture, garden, …)
- tech (computer, laptop, tablet, …)

You can get the scripts and java programs used in this tutorial from our git repository on github:

```
$ git clone https://github.com/fredang/mahout-naive-bayes-exampl
```

You can compile the java programs by typing:

```
$ mvn clean package assembly:single
```

Preparing the training set

UPDATE(2013/06/23): this section was updated to support twitter 1.1 api (1.0 was just shutdown).

As preparing the training set is very time consuming, we have provided in the source repository a training set so that you don’t need to build it. The file is data/tweets-train.tsv. If you choose to use it, you can directly jump to the next section.

To prepare a training set, we fetched the tweets with the following hashtags: #deals, #deal or #discount by using the script twitter_fetcher.py. It is using the python-tweepy 2.1 library (make sure to install the latest version as we have to use the twitter 1.1 api now). You can install it by typing:

```
git clone https://github.com/tweepy/tweepy.git
cd tweepy
sudo python setup.py install
```

You need to have consumer keys/secrets and access token key/secrets to use the api. If you don't have them, simply login on the twitter website then go to: https://dev.twitter.com/apps (https://dev.twitter.com/apps). Then create a new application.
When you are done, you should see in the section 'OAuth settings', the Consumer Key and secret, and in the section 'Your access token', the Access Token and the Access Token secret.

Edit the file script/twitter_fetcher.py and change the following lines to use your twitter keys and secrets:

```
CONSUMER_KEY='REPLACE_CONSUMER_KEY'
CONSUMER_SECRET='REPLACE_CONSUMER_SECRET'
ACCESS_TOKEN_KEY='REPLACE_ACCESS_TOKEN_KEY'
ACCESS_TOKEN_SECRET='REPLACE_ACCESS_TOKEN_SECRET'
```

You can now run the script:

```
$ python scripts/twitter_fetcher.py 5 > tweets-train.tsv
```

Code to fetch tweets:

+ expand source

The file tweets-train.tsv contains a list of tweets in a tab separated value format. The first number is the tweet id followed by the tweet message:

```
308215054011194110      Limited 3-Box $20 BOGO, Supreme $9 BOGO,
308215054011194118      Purchase The Jeopardy! Book by Alex Treb
308215054011194146      #Shopping #Bargain #Deals Designer KATHY
```

To transform this into a training set, you can use your favorite editor and add the category of the tweet at the beginning of the line followed by a tab character:

```
tech    308215054011194110      Limited 3-Box $20 BOGO, Supreme
art     308215054011194118      Purchase The Jeopardy! Book by A
apparel 308215054011194146      #Shopping #Bargain #Deals Design
```

Make sure to use tab between the category and the tweet id and between the tweet id and the tweet message.

For the classifier to work properly, this set must have at least 50 tweets messages in each category.

Training the model with Mahout

First we need to convert the training set to the hadoop sequence file format:

```
$ java -cp target/twitter-naive-bayes-example-1.0-jar-with-depen
```

The sequence file has as key: /[category]/ and as value: .

Code to convert tweet tsv to sequence file

```
+ expand source
```

Then we upload this file to HDFS:

```
$ hadoop fs -put tweets-seq tweets-seq
```

We can run mahout to transform the training sets into vectors using tfidf weights (http://en.wikipedia.org/wiki/Tf%E2%80%93idf)(term frequency x document frequency):

```
$ mahout seq2sparse -i tweets-seq -o tweets-vectors
```

It will generate the following files in HDFS in the directory tweets-vectors:

- df-count: sequence file with association word id => number of document containing this word
- dictionary.file-0: sequence file with association word => word id
- frequency.file-0: sequence file with association word id => word count
- tf-vectors: sequence file with the term frequency for each document
- tfidf-vectors: sequence file with association document id => tfidf weight for each word in the document
- tokenized-documents: sequence file with association document id => list of words
- wordcount: sequence file with association word => word count

In order to do the training and check that the classification works fine, Mahout splits the set into two sets: a training set and a testing set:

```
$ mahout split -i tweets-vectors/tfidf-vectors --trainingOutput
```

We use the training set to train the classifier:

```
$ mahout trainnb -i train-vectors -el -li labelindex -o model -o
```

It creates the model(matrix word id x label id) and a label index(association label and label id).

To test that the classifier is working properly on the training set:

```
$ mahout testnb -i train-vectors -m model -l labelindex -ow -o t
[...]
Summary
-------------------------------------------------------
Correctly Classified Instances          :        314       97.21
Incorrectly Classified Instances        :          9        2.78
Total Classified Instances              :        323

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       <--Class
45      0       0       0       0       0       1        |  46
0       35      0       0       0       0       0        |  35
0       0       34      0       0       0       0        |  34
0       0       0       39      0       0       0        |  39
0       0       0       0       23      0       0        |  23
1       1       0       0       1       48      2        |  53
0       0       1       0       1       1       90       |  93
```

And on the testing set:

```
$ mahout testnb -i test-vectors -m model -l labelindex -ow -o tw
[...]
Summary
-------------------------------------------------------
Correctly Classified Instances          :        121       78.06
Incorrectly Classified Instances        :         34       21.93
Total Classified Instances              :        155


=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       <--Class
27      1       1       1       2       2       2        |  36
1       22      0       2       1       0       0        |  26
0       1       27      1       0       0       1        |  30
0       1       0       23      4       0       0        |  28
0       1       0       2       9       2       0        |  14
0       1       1       1       2       13      1        |  19
0       0       2       0       0       0       0        |  2
```

If the percentage of correctly classified instance is too low, you might need to improve your training set by adding more tweets or by changing your categories to not have too many similar categories or by removing categories that are used very rarely. After you are done with your changes, you would need to restart the training process.

To use the classifier to classify new documents, we would need to copy several files from HDFS:

- model (matrix word id x label id)
- labelindex (mapping between a label and its id)
- dictionary.file-0 (mapping between a word and its id)
- df-count (document frequency: number of documents each word is appearing in)

```
$ hadoop fs -get labelindex labelindex
$ hadoop fs -get model model
$ hadoop fs -get tweets-vectors/dictionary.file-0 dictionary.fil
$ hadoop fs -getmerge tweets-vectors/df-count df-count
```

To get some new tweets to classify, you can run the twitter fetcher again(or use the one provided in data/tweets-to-classify-tsv):

```
$ python scripts/twitter_fetcher.py 1 > tweets-to-classify.tsv
```

Now we can run the classifier on this file:

```
$ java -cp target/twitter-naive-bayes-example-1.0-jar-with-depen
Number of labels: 7
Number of documents: 486
Tweet: 309836558624768000       eBay - Porter Cable 18V Ni CAD 2
  apparel: -252.96630831136127  art: -246.9351025603821  camera:
41377148 => tech
Tweet: 309836557379043329       Newegg - BenQ GW2750HM 27" Wides
  apparel: -287.5588179141781  art: -284.27401807389435  camera:
804349476 => tech
Tweet: 309836556355657728       J and R - Roku 3 Streaming Playe
  apparel: -192.44260718853357  art: -187.6881145121525  camera:
485514894 => camera
Tweet: 309836555248361472       eBay - Adidas Adicross 2011 Men'
  apparel: -133.86214565455646  art: -174.44106424825426  camera
2248114303 => apparel
Tweet: 309836554187202560       Buydig - Tamron 18-270mm Di Lens
  apparel: -218.82418584296866  art: -228.25052760371423  camera
6823539707 => camera
```

Code to classify the tweets using the model and the dictionary file:

```
+ expand source
```

Most of the tweets are classified properly but some are not. For example, the tweet “J and R – Roku 3 Streaming Player 4200R $89.99” is incorrectly classified as camera. To fix that, we can add this tweet to the training set and classify it as tech. You can do the same for the other tweets which are incorrectly classified. When you are done, you can repeat the training process and check the results again.

Conclusion

In this tutorial we have seen how to build a training set, then how to use it with Mahout to train the Naive Bayes model. We showed how to test the classifier and how to improve the training set to get a better classification. Finally we use it to build an application to automatically assign a category to a tweet. In this post, we only study one Mahout classifier among many others: SGD, SVM, Neural Network, Random Forests, …. We will see in future posts how to use them.

Misc

View content of sequence files

To show the content of a file in HDFS, you can use the command

```
$ hadoop fs -text [FILE_NAME]
```

However, there might be some sequence file which are encoded using mahout classes. You can tell hadoop where to find those classes by editing the file [HADOOP_DIR]conf/hadoop-env.sh and add the following line:

```
export HADOOP_CLASSPATH=[MAHOUT_DIR]/mahout-math-0.7.jar:[MAHOUT
```

and restart hadoop.

You can use the command mahout seqdumper:

```
$ mahout seqdumper -i [FILE_NAME]
```

View words which are the most representative of each categories

You can use the class TopCategoryWords that shows the top 10 words of each category.

+ expand source

```
$ java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar
com.chimpler.example.bayes.TopCategoryWords model labelindex dictionary.file-0 df-count
Top 10 words for label camera
– digital: 70.05728101730347
– camera: 63.875202655792236
– canon: 53.79892921447754
– mp: 49.64586567878723
– nikon: 47.830992698669434
– slr: 45.931694984436035
– sony: 44.55785942077637
– lt: 37.998433113098145
– http: 29.718397855758667
– t.co: 29.65730857849121
Top 10 words for label event
– http: 33.16791915893555
– t.co: 33.09973907470703
– deals: 26.246684789657593
– days: 25.533835887908936
– hotel: 22.658542156219482
– discount: 19.89004611968994
– amp: 19.645113945007324
– spend: 18.805208206176758
– suite: 17.21832275390625
– deal: 16.84959626197815
[…]
```

Running the training without splitting the data into testing and training set

You can run the training just after having executed the mahout seq2sparse command:

```
$ mahout trainnb -i tweets-vectors/tfidf-vectors -el -li labelin
```

Using your own testing set with mahout

Previously, we showed how to generate a testing set from the training set using the mahout split command.

In this section, we are going to describe how to use our own testing set and run mahout to check the accuracy of the testing set.

We have a small testing set in data/tweets-test-set.tsv (https://github.com/fredang/mahout-naive-bayes-example/blob/master/data/tweets-test-set.tsv) that we are transforming into a tfidf vector sequence file:

the tweet words are converted into word id using the dictionary file and are associated to their tf x idf value:

+ expand source

To run the program:

```
$ java -cp target/twitter-naive-bayes-example-1.0-jar-with-depen
```

To copy the generated seq file to hdfs:

```
 $ hadoop fs -put tweets-test-set.seq tweets-test-set.seq
```

To run the mahout testnb on this sequence file:

```
  $ mahout testnb -i tweets-test-set.seq -m model -l labelindex
Summary
-------------------------------------------------------
Correctly Classified Instances          :          5       18.51
Incorrectly Classified Instances        :         22       81.48
Total Classified Instances              :         27

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       <--Class
2       1       0       1       3       0       1        |  8
0       0       0       2       0       0       1        |  3
0       0       0       0       1       0       0        |  1
0       0       0       2       1       0       0        |  3
0       0       0       1       1       0       0        |  2
0       0       0       2       0       0       1        |  3
0       3       0       1       0       3       0        |  7
```

Cleanup files

In case you want to rerun the classification, an easy way to delete all the files in your home in HDFS is by typing:

```
$ hadoop fs -rmr \*
```

Errors

When running the script to convert the tweet TSV message, I got the following errors:

```
  Skip line: tech 309167277155168257      Easy web hosting. $4.95
  Skip line: art  309167270989541376      Beautiful Jan Royce Cona
```

Make sure that the category and the tweet id are followed by a tab character and not spaces.

To run the classifier on the hadoop cluster, you can read the post part 2: distribute classification with hadoop (https://chimpler.wordpress.com/2013/06/24/using-the-mahout-naive-bayes-classifier-to-automatically-classify-twitter-messages-part-2-distribute-classification-with-hadoop/). (https://chimpler.wordpress.com/2013/06/24/using-the-mahout-naive-bayes-classifier-to-automatically-classify-twitter-messages-part-2-distribute-classification-with-hadoop/)

FILED UNDER DATAGRID, MACHINE LEARNING, MAHOUT   TAGGED WITH CLASSIFIER, MACHINE LEARNING, MAHOUT, NAIVE BAYES

About chimpler
http://www.chimpler.com

118 Responses to *Using the Mahout Naive Bayes Classifier to automatically classify Twitter messages*

kalyankumar says:
2014/05/21 at 7:52 am
Hello,
Its very urgent. Kindly help me.
In public class Classifier {,

String modelPath = args[0];
String labelIndexPath = args[1];
String dictionaryPath = args[2];
String documentFrequencyPath = args[3];
String tweetsPath = args[4];
I am running Classifier class in eclipse , I am not getting which path I need to provide for above variables. Plz help me. What paths I need to provide?

And Before running Classifier class I have run TweetTSVToSeq class by giving tweets-train.tsv as i/p. I have got chunk-0 seq file as output.

Do I need to run other class’s to generate model,lebelIndex,dictionary, documentFrequency and tweets?

And i am also getting Following errors in Classifier class,

1. The method materialize(Path, Configuration) is undefined for the type
NaiveBayesModel

2.The method all() is undefined for the type
Vector

Save me plzzzzzzz…..

Reply
Daniel says:
2014/06/09 at 8:20 pm
Hello, My Friend

i am trying to use you example inside a Jetty Servlet to classify some spam like the guy at http://emmaespina.wordpress.com/2011/04/26/ham-spam-and-elephants-or-how-to-build-a-spam-filter-server-with-mahout/
However, i am having some problems with materializing the bayes model:

```
NaiveBayesModel model = NaiveBayesModel.materialize(new Path(modelPath), configuration);
```

Heres is the post with curl:

```
curl http://localhost:8080/antispam -H “Content-T-Type: text/xml” –data-binary @ham.txt
```

Here is the error i receive when i do the post:

```
014-06-09 21:16:44.065:INFO::Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
java.lang.IllegalArgumentException: Unknown flags set: %d [-1110101]
at com.google.common.base.Preconditions.checkArgument(Preconditions.java:148)
at org.apache.mahout.math.VectorWritable.readFields(VectorWritable.java:88)
at org.apache.mahout.math.VectorWritable.readVector(VectorWritable.java:199)
at org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:112)
at org.example.SpamClassifier.classify(SpamClassifier.java:49)
at org.example.SpamClassifierServlet.doPost(SpamClassifierServlet.java:24)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:820)
at org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:533)
at org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:475)
at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:119)
at org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:514)
at org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:226)
at org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:920)
at org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:403)
at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:184)
at org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:856)
at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:117)
at
org.eclipse.jetty.server.handler.ContextHandlerCollection.handle(ContextHandlerCollection.java:247)
at org.eclipse.jetty.server.handler.HandlerCollection.handle(HandlerCollection.java:151)
at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:114)
at org.eclipse.jetty.server.Server.handle(Server.java:352)
at org.eclipse.jetty.server.HttpConnection.handleRequest(HttpConnection.java:596)
at org.eclipse.jetty.server.HttpConnection$RequestHandler.content(HttpConnection.java:1066)
at org.eclipse.jetty.http.HttpParser.parseNext(HttpParser.java:805)
at org.eclipse.jetty.http.HttpParser.parseAvailable(HttpParser.java:218)
at org.eclipse.jetty.server.HttpConnection.handle(HttpConnection.java:426)
at org.eclipse.jetty.io.nio.SelectChannelEndPoint.handle(SelectChannelEndPoint.java:510)
at org.eclipse.jetty.io.nio.SelectChannelEndPoint.access$000(SelectChannelEndPoint.java:34)
at org.eclipse.jetty.io.nio.SelectChannelEndPoint$1.run(SelectChannelEndPoint.java:40)
at org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedThreadPool.java:450)
at java.lang.Thread.run(Thread.java:745)
```

Am i missing something? The project is at github (https://github.com/danielneis/mahout-spamclassifier-servlet) , i've tried several versions of mahout/hadoop-common/hadoop-core/hadoop-hdfs in the pom file.

Reply

tz says:
2014/12/18 at 2:20 am
Same problem here, have you solved it?

Reply

Sandeepan Jindal says:
2015/09/19 at 7:08 am
I was facing the same issue. Fixing the pom.xml file fixed it for me.

Remove all the dependencies and use the ones below. From mahout 0.10, they have partitioned mahout-core.jar into mahout-mr and mahout-hdfs.

org.apache.mahout
mahout-mr
0.11.0

org.apache.mahout
mahout-hdfs
0.11.0

org.apache.mahout
mahout-math
0.11.0

Tyler Rockwood says:
2015/10/28 at 3:42 pm
Same…. Has anyone solved it?

Hubert says:
2015/11/19 at 3:39 pm

Yes 🙂
This exception occurs because you trained the model with -c flag (Complementary Naive Bayes)
Just replace

```
StandardNaiveBayesClassifier classifier = new StandardNaiveBayesClassifier(model);
```

to

```
ComplementaryNaiveBayesClassifier classifier = new
ComplementaryNaiveBayesClassifier(model);
```

and it will work. Remember about the imports 🙂

Joanna says:
2015/08/24 at 3:37 am
I have the same problem. have you solved it?

Reply

suprichan says:
2014/09/08 at 12:46 pm
I dont understan this part

```
$ java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar
com.chimpler.example.bayes.TweetTSVToSeq data/tweets-train.tsv tweets-seq
```

Reply

suprichan says:
2014/09/08 at 3:31 pm

```
$ mvn clean package assembly:single

[INFO] Scanning for projects…
[INFO] ————————————————————————————————————
[INFO] BUILD FAILURE
[INFO] ————————————————————————————————————
[INFO] Total time: 0.135s
[INFO] Finished at: Tue Sep 09 04:30:25 JST 2014
[INFO] Final Memory: 4M/82M
[INFO] ————————————————————————————————————
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory
(/Users/apple). Please verify you invoked Maven from the correct directory. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following
articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProjectException
```

How to fix? What wrong?

Reply

Toandq says:

2015/01/08 at 12:43 pm
You should make a double check with your project for this “requires a project to execute but there is no POM in this directory (/Users/apple). “

Reply
Supri Amir says:
2014/09/10 at 3:04 am

```
java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar
com.chimpler.example.bayes.TweetTSVToSeq data/tweets-train.tsv tweets-seq
```

I got error in this part

Message

```
Error :Could not find or load main class com.mycompany.app.TweetTSVToSeq
```

Reply
Ian Tivey says:
2014/09/20 at 6:30 am
Hi,

Thanks for the post, very useful.

FYI, I received the following error when running twitter_fetcher.py

“UnicodeEncodeError: ‘ascii’ codec can’t encode character”

which I fixed by changing line 40 from

“print results.text”
to
“print results.text.encode(‘utf-8’)”

Reply
Ian Tivey says:
2014/09/20 at 6:33 am
In fact, should that line even be in the python script at all?

Reply
Skanda Prasad says:
2014/10/10 at 9:27 am
Excellent article.

Reply
Nandini says:
2014/10/22 at 7:39 am
While executing Classifier.java program in eclipse we are getting the following error
Can anyone help us in resolving this?

```
Exception in thread “main” java.lang.NoSuchMethodError:
org.apache.mahout.math.SparseRowMatrix.(II)V
at
```

```
org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:115)
at com.chimpler.example.bayes.Classifier.main(Classifier.java:56)
```

Reply

Nandini says:
2014/10/27 at 1:31 am
Please can any one answer for the above question as its very urgent

Reply

Nandini says:
2014/11/07 at 4:34 am
Add jar files from mahout library of CDH if working on cloudera. If we use downloaded jar file we get such errors

Patricia says:
2014/10/29 at 5:52 pm
hi, when I try: mahout seq2sparse -i tweets-seq -o tweets-vectors,
I get the error:

```
Exception in thread "main" org.apache.hadoop.mapreduce.lib.input.InvalidInputException: Input path does not exist: hdfs://localhost:9000/user/hduser/tweets-seq
at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.listStatus(FileInputFormat.java:285)
```

Any idea?
thx!

Reply

Patricia says:
2014/10/29 at 6:37 pm
I get it, I need to create the dir. /user/hduser/tweets-seq in hdfs 🙂 now it's working

Reply

Archana says:
2014/11/06 at 1:07 pm
Hi chimpler
I have huge doubt.I can find only text classification examples on internet .I am creating heart disease prediction system(cleveland dataset from UCI repository) and I want to write classifier for the same after creating model.I am not able to understand how to write ?I am using twitter classifier code but its not working,can you suggest something for this?

Reply

Tamer Yousef says:
2014/12/04 at 3:29 pm
This is by far the best mahout tutorial I found!! seriously, you touch on several points such as how to run a test on an existing data set from a model that is already generated, with details….this article is great!!!!! THUMBS UP>>>>

Reply

Toandq says:
2015/01/08 at 12:41 pm

Hello,

I’ve got a problem when executing this command: $ java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar com.chimpler.example.bayes.Classifier model labelindex dictionary.file-0 df-count data/tweets-to-classify.tsv

```
Exception in thread “main” java.lang.IllegalArgumentException: Unknown flags set: %d [-1011000]
at com.google.common.base.Preconditions.checkArgument(Preconditions.java:119)
at org.apache.mahout.math.VectorWritable.readFields(VectorWritable.java:88)
at org.apache.mahout.math.VectorWritable.readVector(VectorWritable.java:199)
at org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:112)
at com.chimpler.example.bayes.Classifier.main(Classifier.java:83)
```

Everything is okay with the previous commands. Thank you so much.

Reply

Ben Youb says:
2015/04/04 at 6:15 am
when i tried to classify new tweets with this command:

```
java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar
com.chimpler.example.bayes.Classifier model labelindex dictionary.file-0 df-count data/tweets-to-classify.tsv
```

i got something like that:

```
Exception in thread “main” java.lang.IllegalArgumentException: Unknown flags set: %d [-1011000]
at com.google.common.base.Preconditions.checkArgument(Preconditions.java:119)
at org.apache.mahout.math.VectorWritable.readFields(VectorWritable.java:88)
at org.apache.mahout.math.VectorWritable.readVector(VectorWritable.java:199)
at org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:112)
at com.chimpler.example.bayes.Classifier.main(Classifier.java:83)
```

need help !!!!

Reply

Sting says:
2015/05/14 at 7:54 pm
have you fix it yet? can you share the solution with me please?

Reply

Sandeepan Jindal says:
2015/09/19 at 7:08 am
I was facing the same issue. Fixing the pom.xml file fixed it for me.

Remove all the dependencies and use the ones below. From mahout 0.10, they have partitioned mahout-core.jar into mahout-mr and mahout-hdfs.

```
org.apache.mahout
mahout-mr
0.11.0
```

```
org.apache.mahout
mahout-hdfs
0.11.0

org.apache.mahout
mahout-math
0.11.0
```

Hubert says:
2015/11/19 at 3:39 pm
SOLVED:
This exception occurs because you trained the model with -c flag (Complementary Naive Bayes)
Just replace

```
StandardNaiveBayesClassifier classifier = new StandardNaiveBayesClassifier(model);
```

to

```
ComplementaryNaiveBayesClassifier classifier = new
ComplementaryNaiveBayesClassifier(model);
```

and it will work. Remember about the imports 🙂

Joanna says:
2015/08/25 at 6:29 am
hi, I have the same problem and solve it today. I find that if I keep the version of mahout used in the project and version of the mahout jar used in hadoop be same, the problem is solve.

Reply

Pingback: Apache Mahout – Quick and dirty | Luiz Henrique Zambom Santana

avinashkumar721988 says:
2015/06/23 at 8:25 am
I tried to run the below command and I got the error something like this. Can you help me to come out Please.

```
java -cp target/twitter-naive-bayes-example-1.0-jar-with-dependencies.jar
com.chimpler.example.bayes.TopCategoryWords model labelindex dictionary.file-0 df-count
Exception in thread “main” java.lang.IllegalArgumentException: Unknown flags set: %d [-1011000]
at com.google.common.base.Preconditions.checkArgument(Preconditions.java:119)
at org.apache.mahout.math.VectorWritable.readFields(VectorWritable.java:88)
at org.apache.mahout.math.VectorWritable.readVector(VectorWritable.java:199)
at org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:112)
at com.chimpler.example.bayes.TopCategoryWords.main(TopCategoryWords.java:118)
```

Reply

avinashkumar721988 says:
2015/08/18 at 5:30 am
Hi Chimpler,

I am getting following issue when I am running classifier.

```
Exception in thread “main” java.lang.IllegalArgumentException: Unknown flags set: %d [11001]
at com.google.common.base.Preconditions.checkArgument(Preconditions.java:119)
at org.apache.mahout.math.VectorWritable.readFields(VectorWritable.java:88)
at org.apache.mahout.math.VectorWritable.readVector(VectorWritable.java:199)
at org.apache.mahout.classifier.naivebayes.NaiveBayesModel.materialize(NaiveBayesModel.java:111)
at com.chimpler.example.bayes.Classifier.main(Classifier.java:80).
```

Could you please look at this. Why I am getting this problem.
Thank You.

Reply

Create a free website or blog at WordPress.com.