Homework 2

rz2356
Ruomeng Zhang

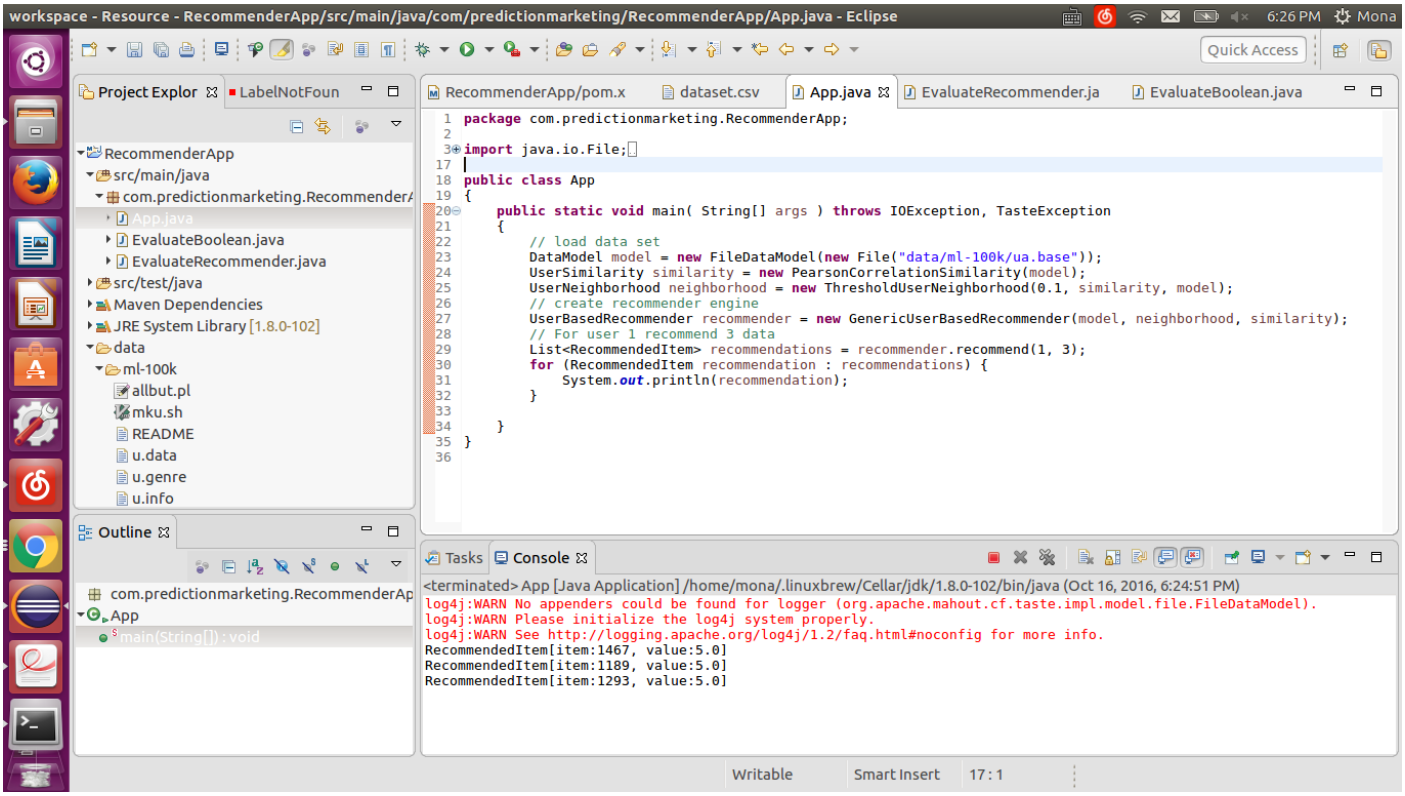
Mahout

1 Recommend

I got the data from the URL: http://grouplens.org/datasets/movielens/ (http://grouplens.org/datasets/movielens/). It is recommend by the book *Mahout in Action* I tried the samples to familiar the system.

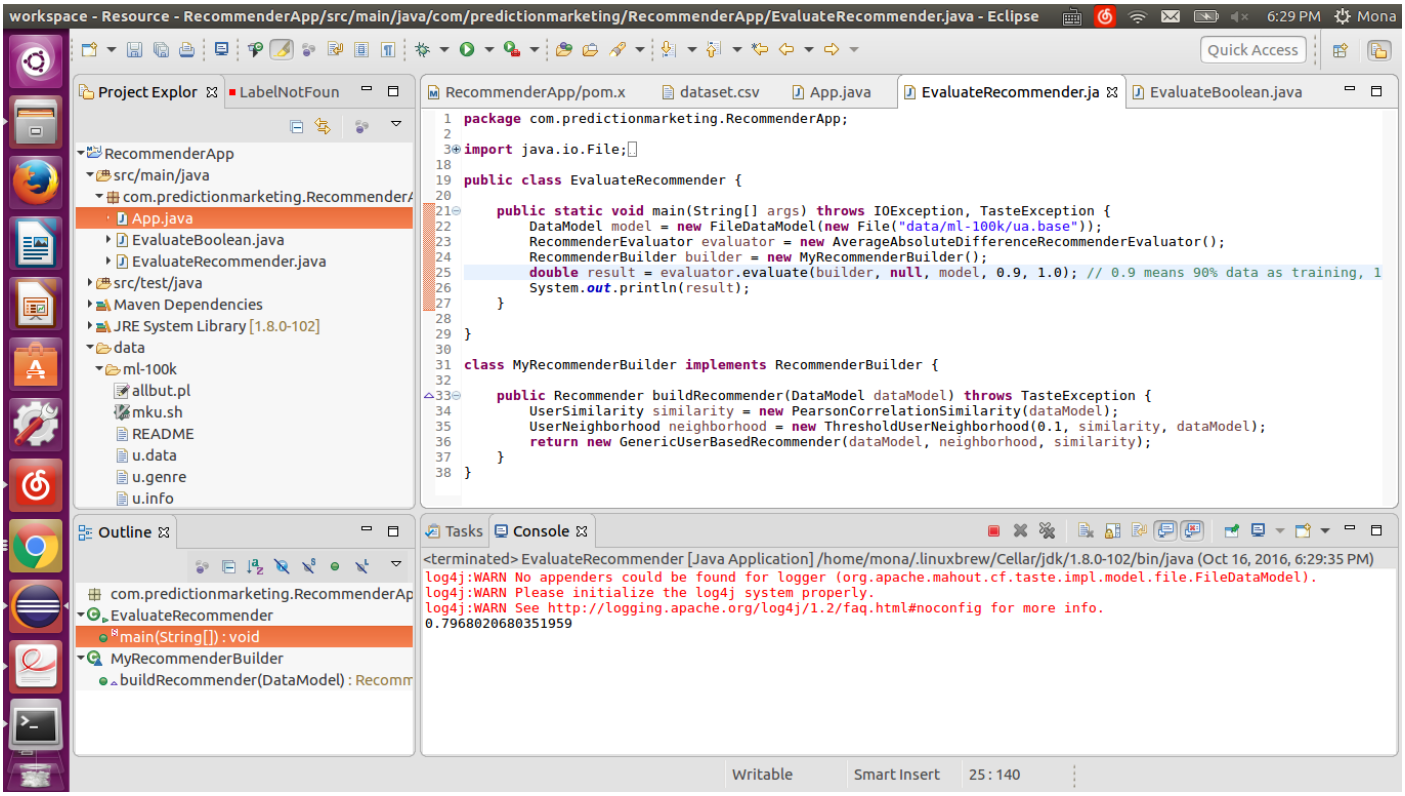
I got the data from data set Eric provided. I applied for the Yahoo Data, but it has pending for 3 days. The two sets are music recommendation and movie recommendation. I will use different kind of algorithms to do recommendation.

1.1 Recommender



A `DataModel` implementation stores and provides access to all the preference, user, and item data needed in the computation.
A `UserSimilarity` implementation provides some notion of how similar two users are; this could be based on one of many possible metrics or calculations.
A `UserNeighborhood` implementation defines a notion of a group of users that are most similar to a given user.
Finally, a `Recommender` implementation pulls all these components together to recommend items to users.

1.2 Evaluating



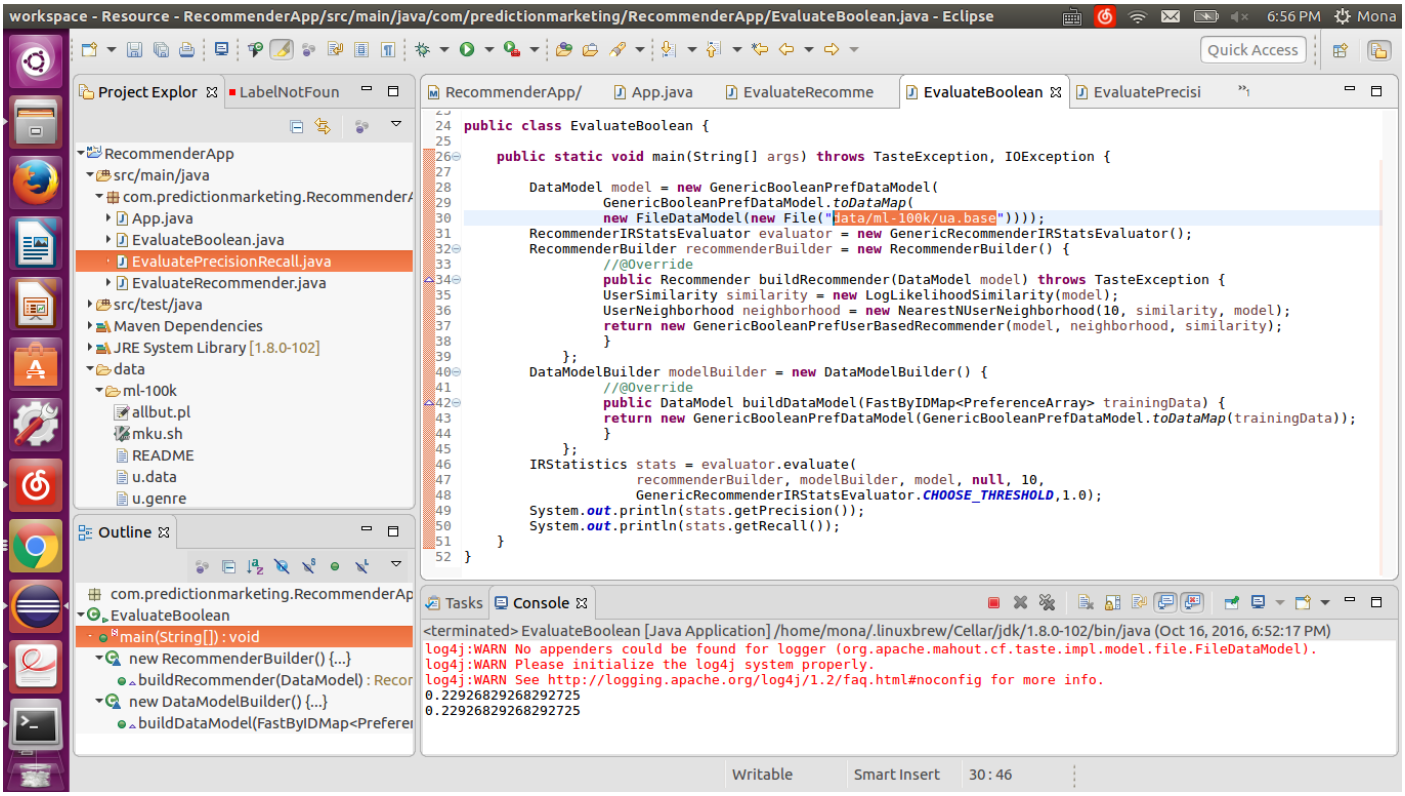
Most of the action happens in `evaluate()`: the `RecommenderEvaluator` splits the data into a training and test set, builds a new training `DataModel` and `Recommender` to test, and compares its estimated preferences to the actual test data.

There's no `Recommender` passed to `evaluate()`. That's because, inside, the method will need to build a `Recommender` around a newly created training `DataModel`. The caller must provide an object that can build a Recommender from a `DataModel`—a `RecommenderBuilder`.

What this value means depends on the implementation `AverageAbsoluteDifferenceRecommenderEvaluator`

A result of 0.7968020680351959 from this implementation means that, on average, the recommender estimates a preference that deviates from the actual preference by 0.7968020680351959.

1.3 Evaluating with Boolean data

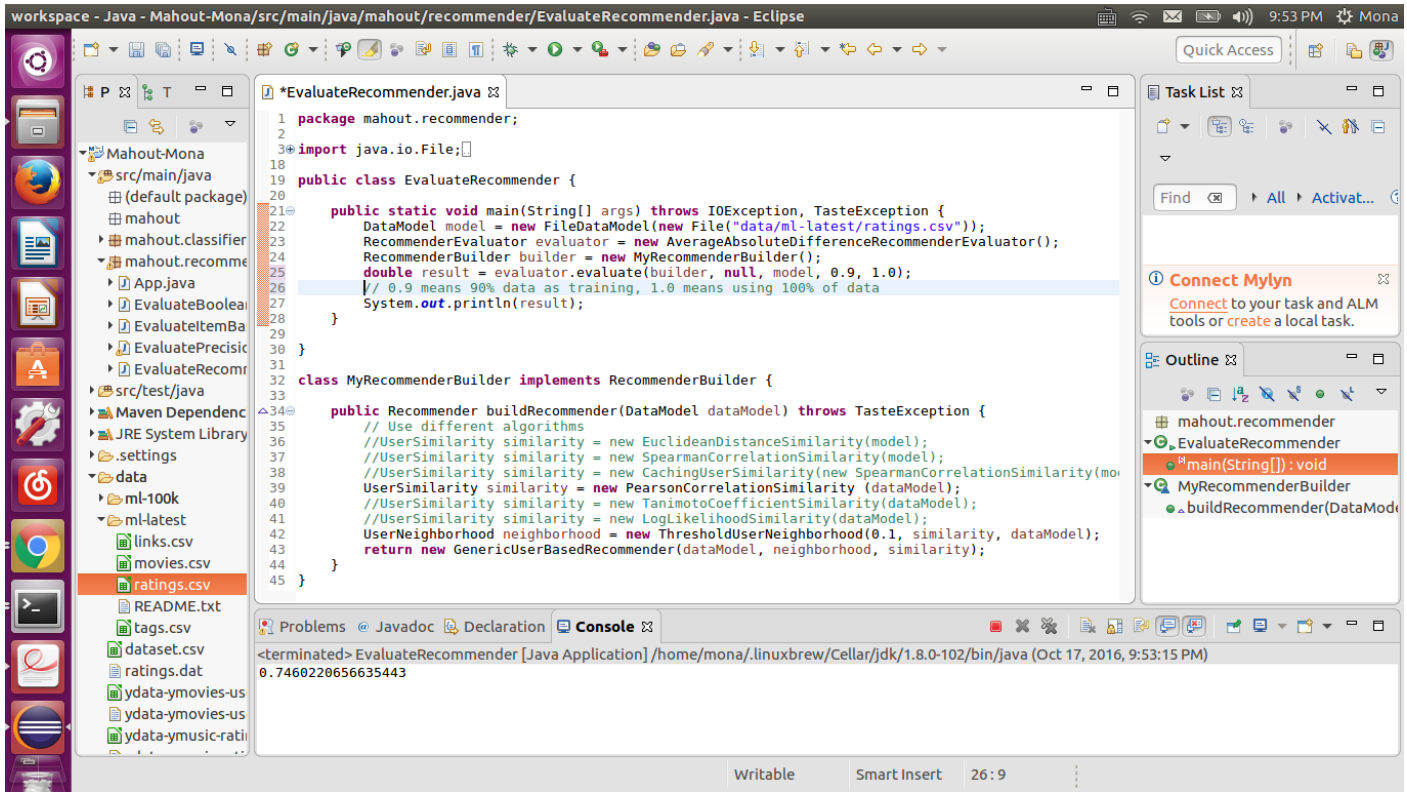


The `GenericBooleanPrefDataModel` takes its input in a slightly different way—as a bunch of `FastIDSets` rather than `PreferenceArrays` and the convenience method `toDataMap()` exists to translate between the two.

1.4 Algorithms -- UserSimilarity.

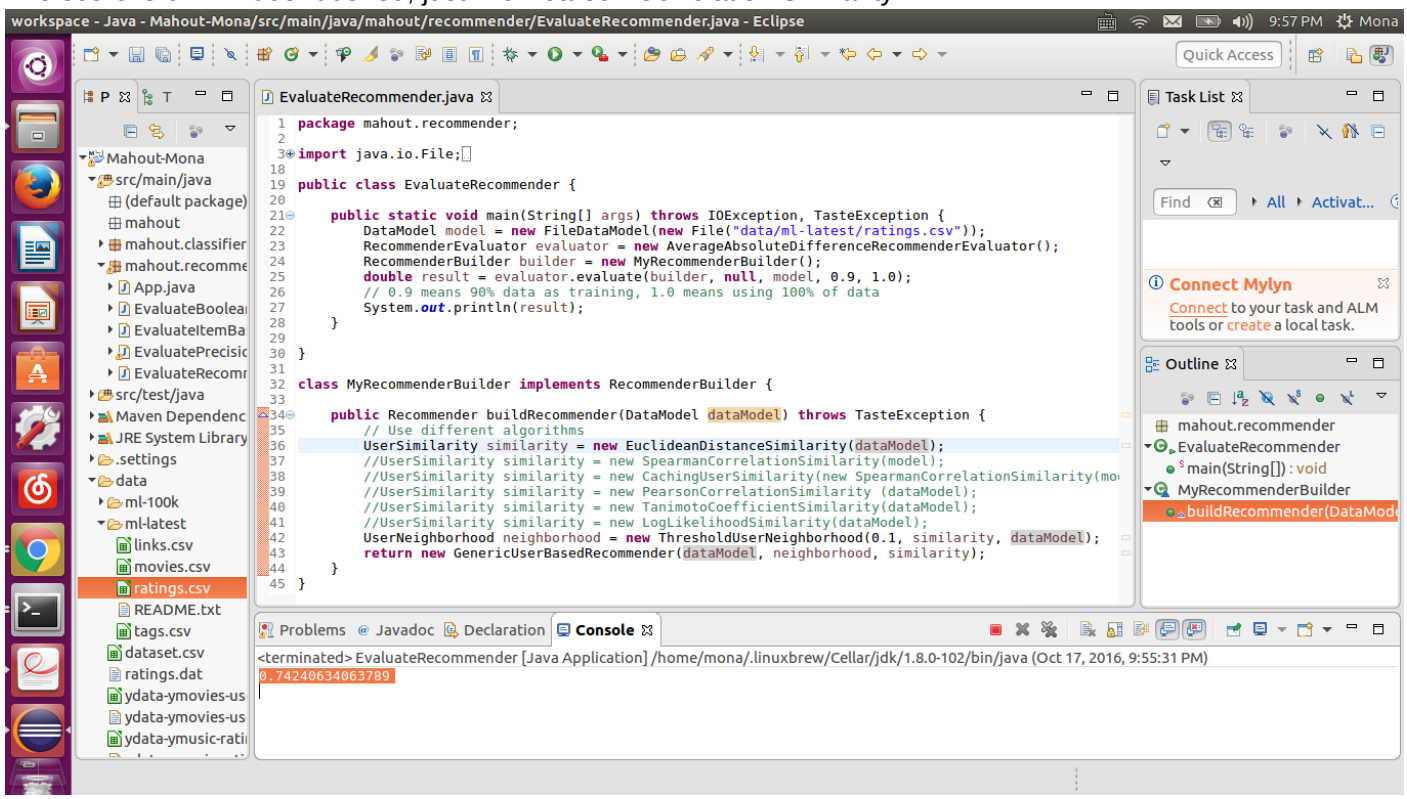
1.4.1 PearsonCorrelationSimilarity

The averange envaluation of the recommendation has a difference of 0.7460220656635443 out of 5.It is not really good.



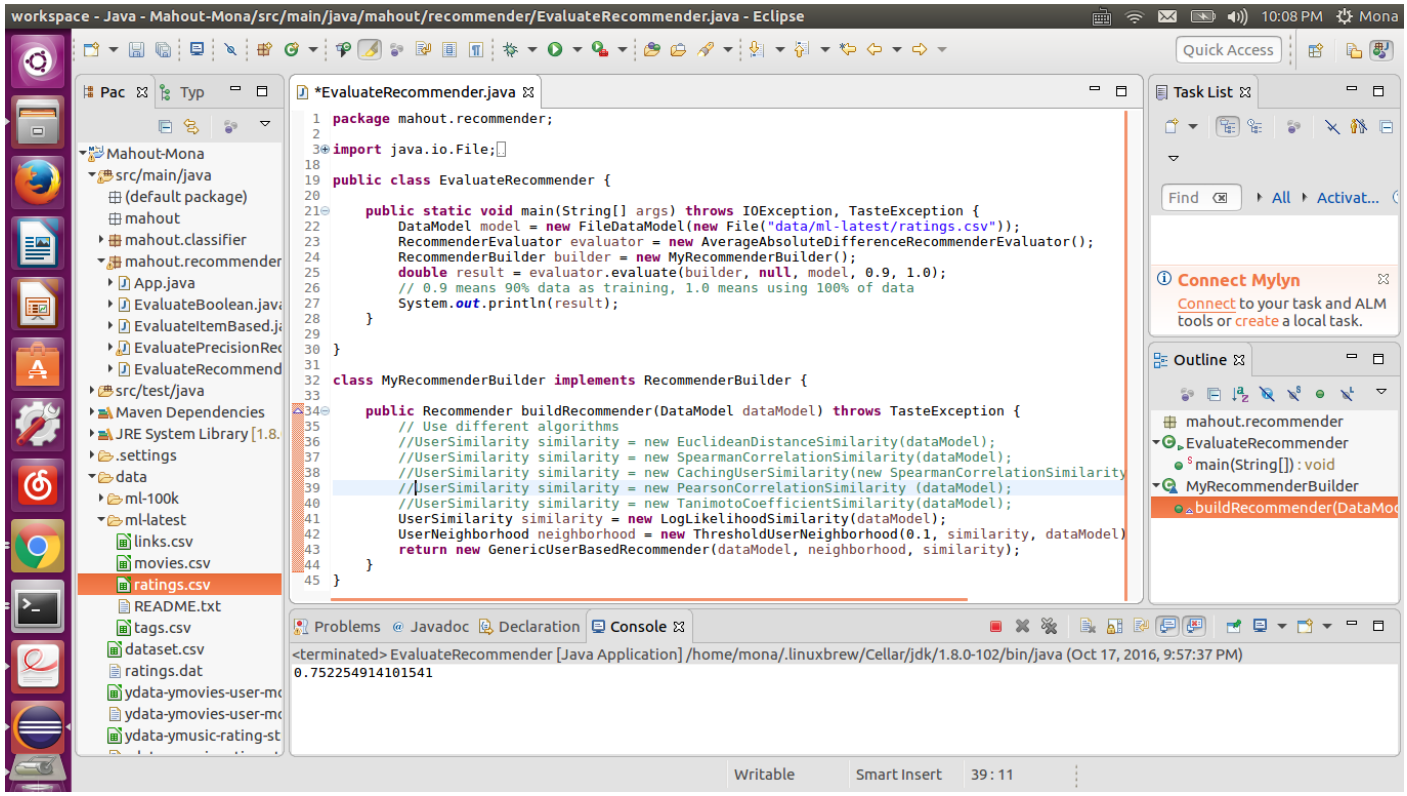
1.4.2 EuclideanDistanceSimilarity

The score is 0.74240634063789, just like Pearson Correlation Similarity.



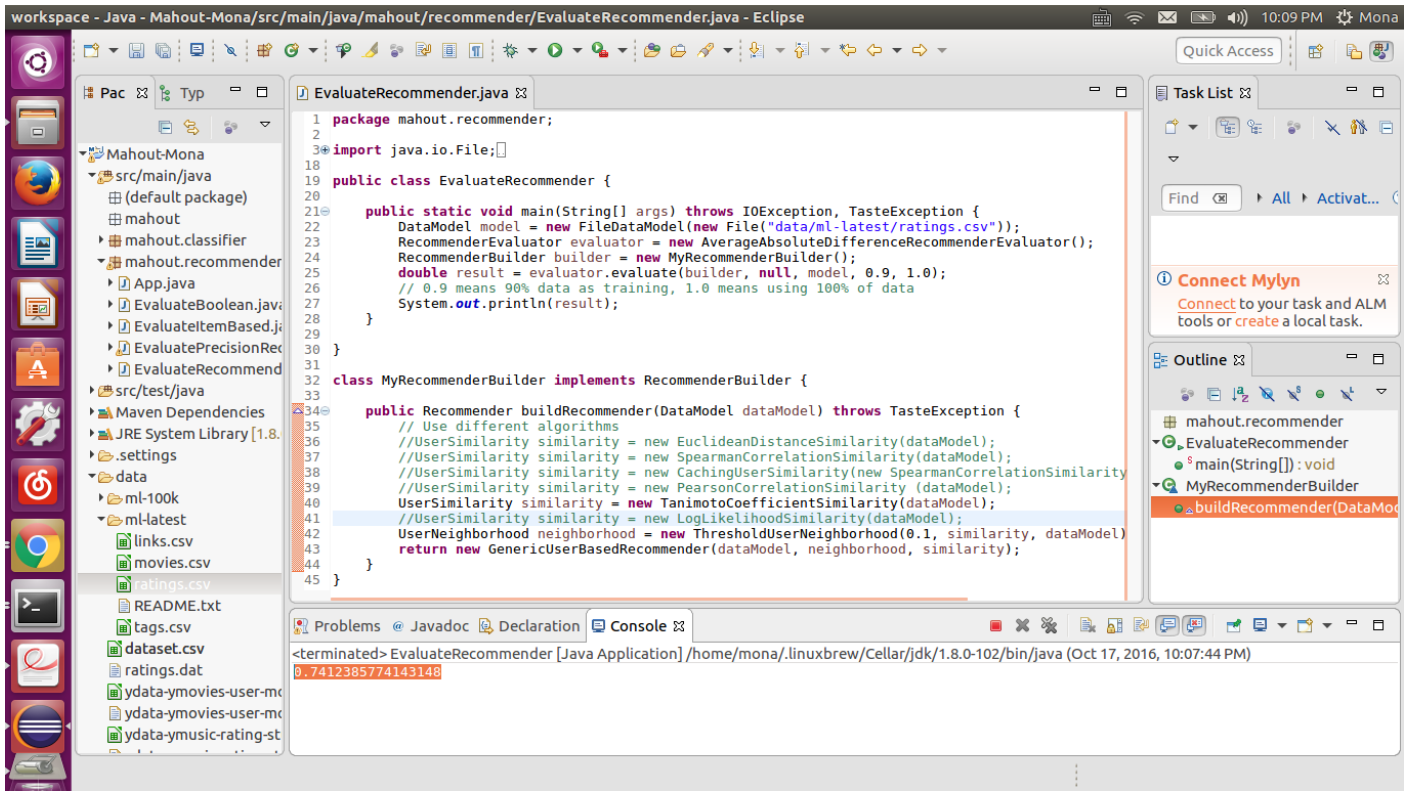
1.4.3 LogLikelihoodSimilarity

The result is 0.752254914101541.



1.4.4 TanimotoCoefficientSimilarity

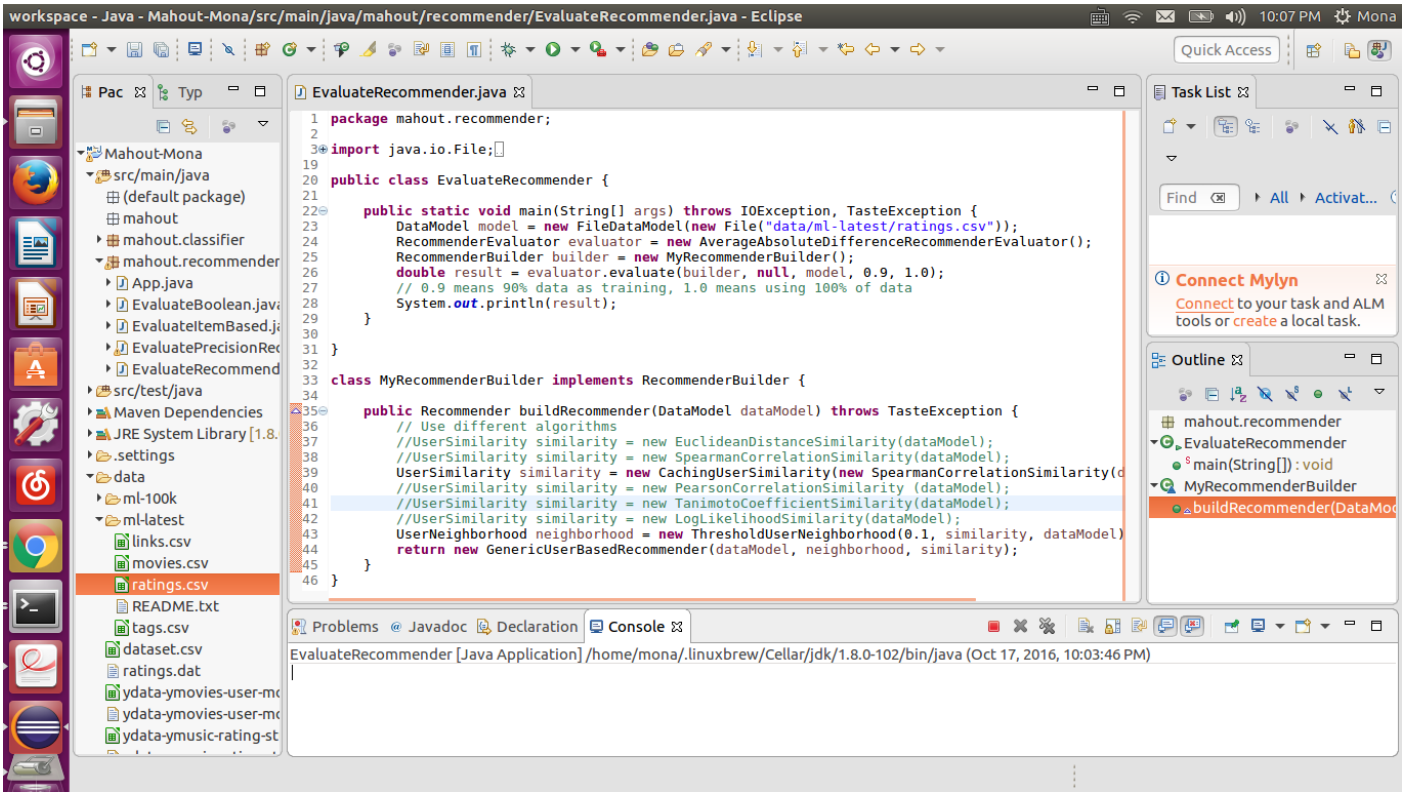
The result is 0.7412385774143148



1.4.5 SpearmanCorrelationSimilarity

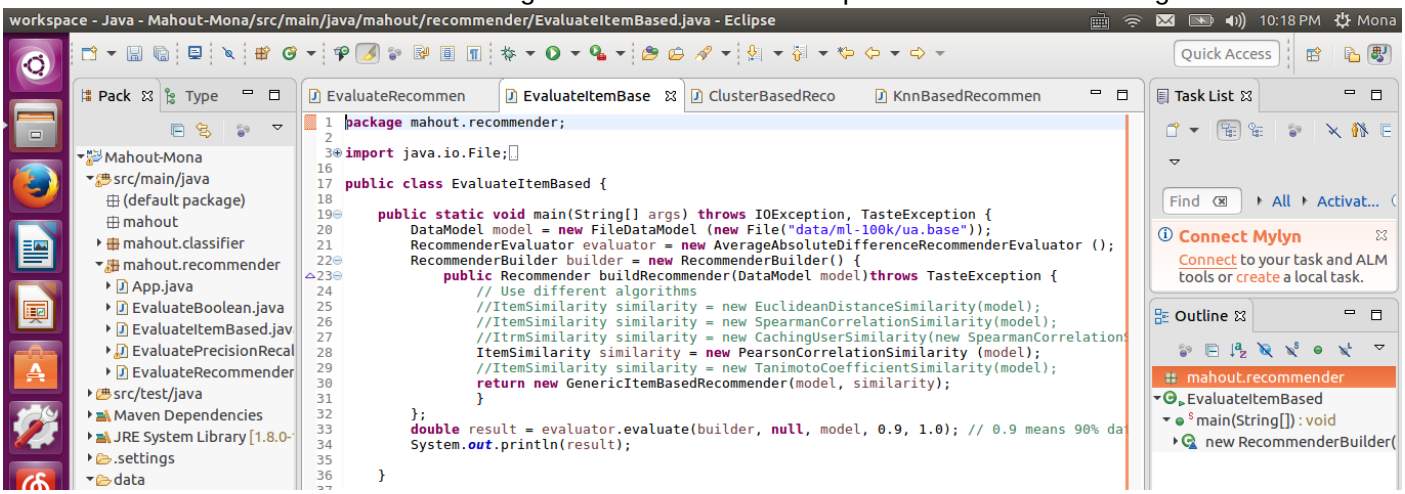
It spend a lot of time even I change the code to caching that data.

```
UserSimilarity similarity = new CachingUserSimilarity(new SpearmanCorrelatio
nSimilarity(dataModel), dataModel);
```



1.5 Algorithms -- ItemSimilarity

This suggests one reason that you might choose an item-based recommender: if the number of items is relatively low compared to the number of users, the performance advantage could be significant. The score is 0.8017587295632225. Looks like it is not good for this data set compare to the user-based algorithms.



2 Clastering

2.1 Reuters clustering using k-means

1.Download the file

From http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.tar.gz (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.tar.gz) or use the command line.

```
curl http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.tar.gz -o
 ${WORK_DIR}/reuters21578.tar.gz
```

The WORK_DIR is set as the palce I will store the file.

2.Extract the file

Make a new folder and extract the file to the new folder

```
mkdir -p ${WORK_DIR}/reuters-sgm
tar xzf ${WORK_DIR}/reuters21578.tar.gz -C ${WORK_DIR}/reuters-sgm
```

3.Put the data to Hadoop

```
hadoop fs -put ${WORK_DIR}/reuters-sgm ${WORK_DIR}/reuters-sgm
hadoop fs -put ${WORK_DIR}/reuters-out ${WORK_DIR}/reuters-out
```

put them in the same place as in local. At first, but I should mkdir at first.

4.Convert to sequence files from directory

Run the `SequenceFileFromDirectory` class. This will write the Reuters articles in the SequenceFile format.

```
bin/mahout seqdirectory -i ${WORK_DIR}/reuters-out -o ${WORK_DIR}/reuters-ou
t-seqdir -c UTF-8 -chunk 64 -xm sequential
```

The chunkSize `-chunk` in MegaBytes is default to 64.
The name of the character `-c` encoding of the input filesis default to UTF-8.
The execution method '-xm' to use sequential or mapreduce.

5.Convert to vectors

To do that, run the `SparseVectorsFromSequenceFiles` class using the Mahout launcher script.

```
bin/mahout seq2sparse \
    -i ${WORK_DIR}/reuters-out-seqdir/ \
    -o ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans --maxDFPercent 85 --name
dVector
```

The `-maxDFPercent` is the max percentage of docs for the DF. Can be used to remove really high frequency terms. Default is 99 and can be set from 0 to 99.
The `-nemedVector` means whether output vectors should be NamedVectors. It will be true if setted.

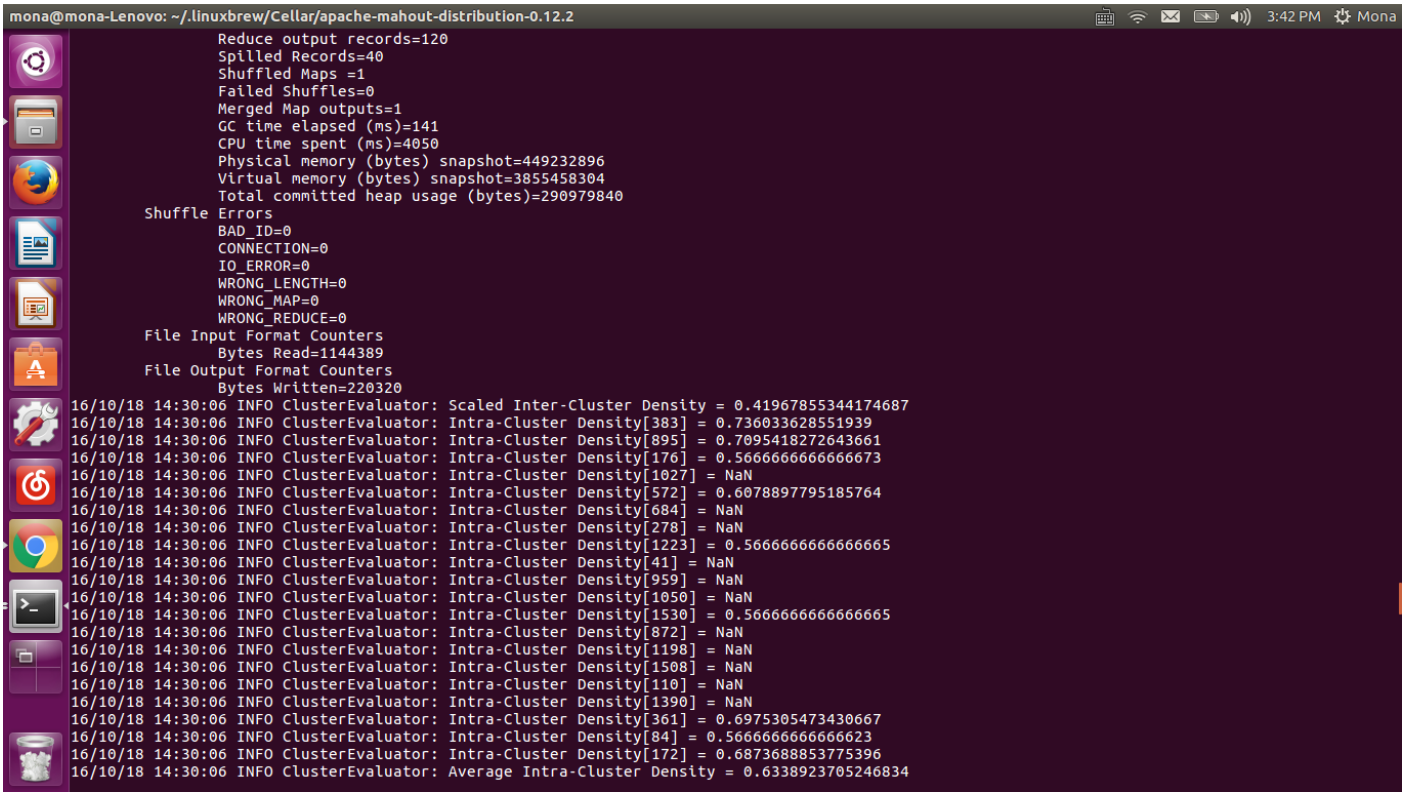
6.Do k-means clustering

```
bin\mahout kmeans \
    -i ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans/tfidf-vectors/ \
    -c ${WORK_DIR}/reuters-kmeans-clusters \
    -o ${WORK_DIR}/reuters-kmeans \
    -dm org.apache.mahout.common.distance.EuclideanDistanceMeasure \
    -x 10 -k 20 -ow --clustering
```

the meanings of the parameters are:

```
    -i <input vectors directory> \
    -c <input clusters directory> \
    -o <output working directory> \
    -k <optional number of initial clusters to sample from input vectors> \
    -dm <DistanceMeasure> \
    -x <maximum number of iterations> \
    -cd <optional convergence delta. Default is 0.5> \
    -ow <overwrite output directory if present>
    -cl <run input vector clustering after computing Canopies>
    -xm <execution method: sequential or mapreduce>
```

We’re using the Java execution plug-in of the Maven command to execute k-means clustering with the required command-line arguments.
The –k means 20 argument specifies that the centroids are randomly generated using RandomSeedGenerator and are writtento the input clusters folder.

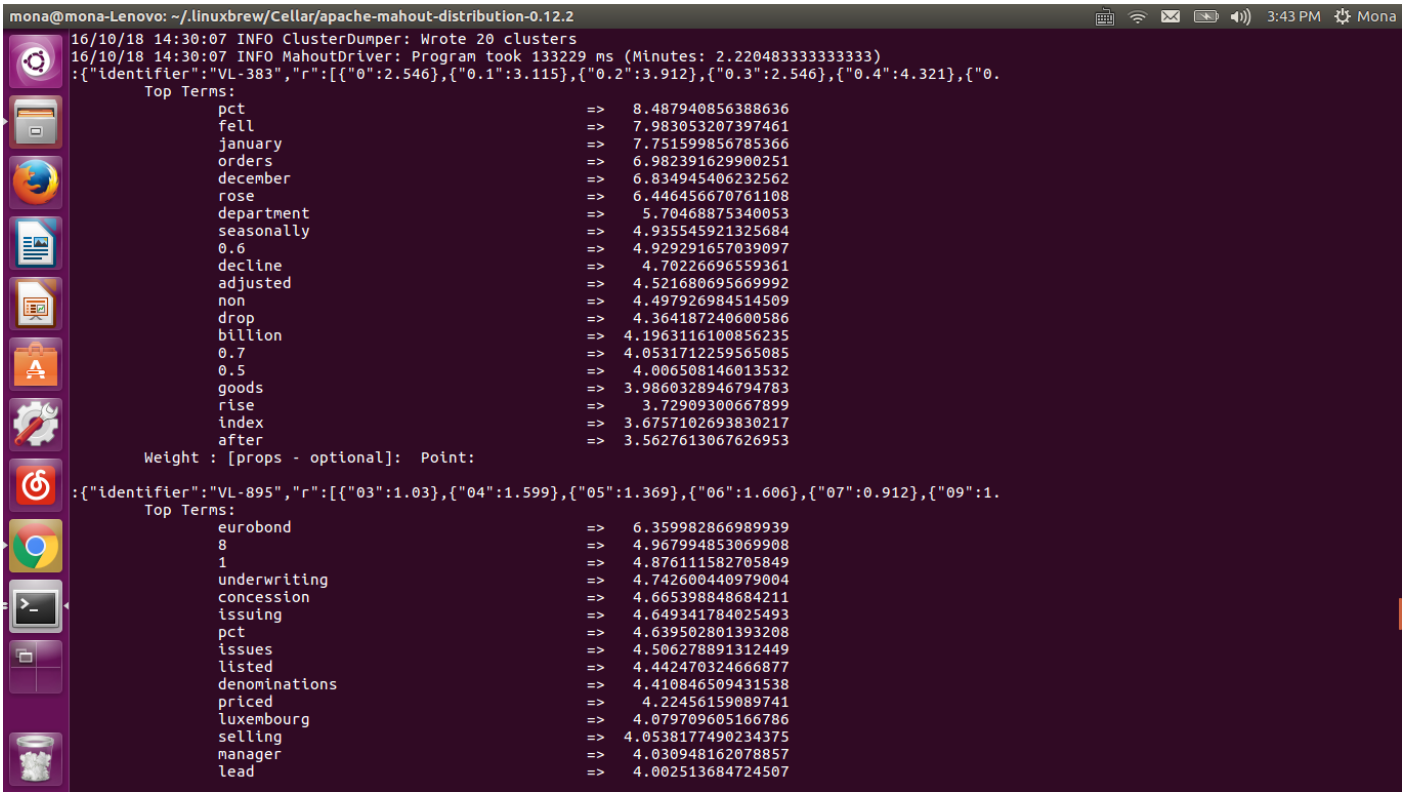


7.Read the output

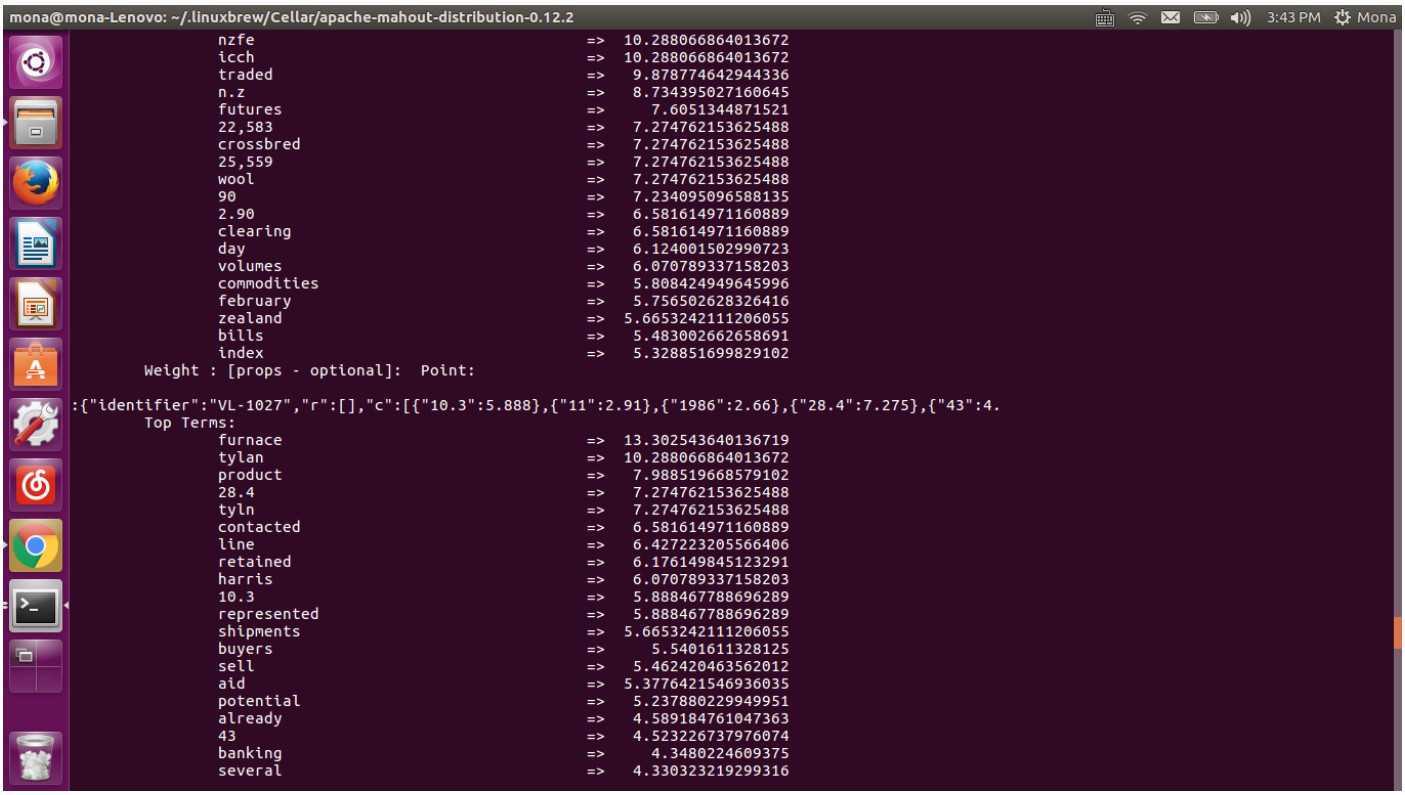
Once the clustering is done, there is a way to inspect the clusters and see how they're formed. Mahout has a utility called `org.apache.mahout.utils.clustering`.
`ClusterDumper` that can read the output of any clustering algorithm and show the top terms in each cluster and the documents belonging to that cluster.

```
bin/mahout clusterdump \
    -i `$DFS -ls -d ${WORK_DIR}/reuters-kmeans/clusters-*-final | awk '{prin
t $8}'` \
    -o ${WORK_DIR}/reuters-kmeans/clusterdump \
    -d ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans/dictionary.file-0 \
    -dt sequencefile -b 100 -n 20 --evaluate -dm org.apache.mahout.common.di
stance.EuclideanDistanceMeasure -sp 0 \
    --pointsDir ${WORK_DIR}/reuters-kmeans/clusteredPoints \
    && \
  cat ${WORK_DIR}/reuters-kmeans/clusterdump
```

ClusterDumper shows the top words of the cluster for each centroid. To get the actual mapping of the points to the clusters, read the `SequenceFiles` in the `clusteredPoints/` folder.




```
16/10/18 14:30:07 INFO ClusterDumper: Wrote 20 clusters
16/10/18 14:30:07 INFO MahoutDriver: Program took 133229 ms (Minutes: 2.220483333333333)
:{"identifier":"VL-383","r":[{"0":2.546},{"0.1":3.115},{"0.2":3.912},{"0.3":2.546},{"0.4":4.321},{"0.
        Top Terms:
                pct                                     =>   8.487940856388636
                fell                                    =>   7.983053207397461
                january                                 =>   7.751599856785366
                orders                                  =>   6.982391629900251
                december                                =>   6.834945406232562
                rose                                    =>   6.446456670761108
                department                              =>    5.70468875340053
                seasonally                              =>   4.935545921325684
                0.6                                     =>   4.929291657039097
                decline                                 =>    4.70226696559361
                adjusted                                =>   4.521680695669992
                non                                     =>   4.497926984514509
                drop                                    =>   4.364187240600586
                billion                                 =>  4.1963116100856235
                0.7                                     =>  4.0531712259565085
                0.5                                     =>   4.006508146013532
                goods                                   =>  3.9860328946794783
                rise                                    =>    3.72909300667899
                index                                   =>  3.6757102693830217
                after                                   =>  3.5627613067626953
        Weight : [props - optional]:  Point:

:{"identifier":"VL-895","r":[{"03":1.03},{"04":1.599},{"05":1.369},{"06":1.606},{"07":0.912},{"09":1.
        Top Terms:
                eurobond                                =>   6.359982866989939
                8                                       =>   4.967994853069908
                1                                       =>   4.876111582705849
                underwriting                            =>   4.742600440979004
                concession                              =>   4.665398848684211
                issuing                                 =>   4.649341784025493
                pct                                     =>   4.639502801393208
                issues                                  =>   4.506278891312449
                listed                                  =>   4.442470324666877
                denominations                           =>   4.410846509431538
                priced                                  =>    4.22456159089741
                luxembourg                              =>   4.079709605166786
                selling                                 =>  4.0538177490234375
                manager                                 =>   4.030948162078857
                lead                                    =>   4.002513684724507
```




```
                nzfe                                    =>  10.288066864013672
                icch                                    =>  10.288066864013672
                traded                                  =>   9.878774642944336
                n.z                                     =>   8.734395027160645
                futures                                 =>     7.6051344871521
                22,583                                  =>   7.274762153625488
                crossbred                               =>   7.274762153625488
                25,559                                  =>   7.274762153625488
                wool                                    =>   7.274762153625488
                90                                      =>   7.234095096588135
                2.90                                    =>   6.581614971160889
                clearing                                =>   6.581614971160889
                day                                     =>   6.124001502990723
                volumes                                 =>   6.070789337158203
                commodities                             =>   5.808424949645996
                february                                =>   5.756502628326416
                zealand                                 =>  5.6653242111206055
                bills                                   =>   5.483002662658691
                index                                   =>   5.328851699829102
        Weight : [props - optional]:  Point:

:{"identifier":"VL-1027","r":[],"c":[{"10.3":5.888},{"11":2.91},{"1986":2.66},{"28.4":7.275},{"43":4.
        Top Terms:
                furnace                                 =>  13.302543640136719
                tylan                                   =>  10.288066864013672
                product                                 =>   7.988519668579102
                28.4                                    =>   7.274762153625488
                tyln                                    =>   7.274762153625488
                contacted                               =>   6.581614971160889
                line                                    =>   6.427223205566406
                retained                                =>   6.176149845123291
                harris                                  =>   6.070789337158203
                10.3                                    =>   5.888467788696289
                represented                             =>   5.888467788696289
                shipments                               =>  5.6653242111206055
                buyers                                  =>     5.5401611328125
                sell                                    =>   5.462420463562012
                aid                                     =>  5.3776421546936035
                potential                               =>   5.237880229949951
                already                                 =>   4.589184761047363
                43                                      =>   4.523226737976074
                banking                                 =>     4.3480224609375
                several                                 =>   4.330323219299316
```

2.2 Wikipedia articles clustering using

I wrote a script based on the example to clusting wikipedia data set. It include download, extact file, upload to HDFS, using different kind algorithm to do clustering.
As my computer memory is too small for map reduce job, it crashed down many times when excuting clustering. Thus, I use the small set in that script. But it can be changed in 71 line of the code.
Check the code on the gitgub https://github.com/wasabi233/BigData/ (https://github.com/wasabi233/BigData/)

2.2.1 Download and extract dataset

```
source $MAHOUT_HOME/examples/bin/set-dfs-commands.sh
export WORK_DIR=/tmp/mahout-work-wiki
mkdir -p ${WORK_DIR}
mkdir -p ${WORK_DIR}/wikixml$
```

Download the data

```
curl https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles
1.xml-p000000010p000030302.bz2 -o ${WORK_DIR}/wikixml/enwiki-latest-pages-ar
ticles.xml.bz2
cd ${WORK_DIR}/wikixml
```

Unzip the data

```
bunzip2 enwiki-latest-pages-articles.xml.bz2
```

Upload to HDFS

```
$DFSRM ${WORK_DIR}/wikixml
$DFS -mkdir -p ${WORK_DIR}
$DFS -put ${WORK_DIR}/wikixml ${WORK_DIR}/wikixml
```

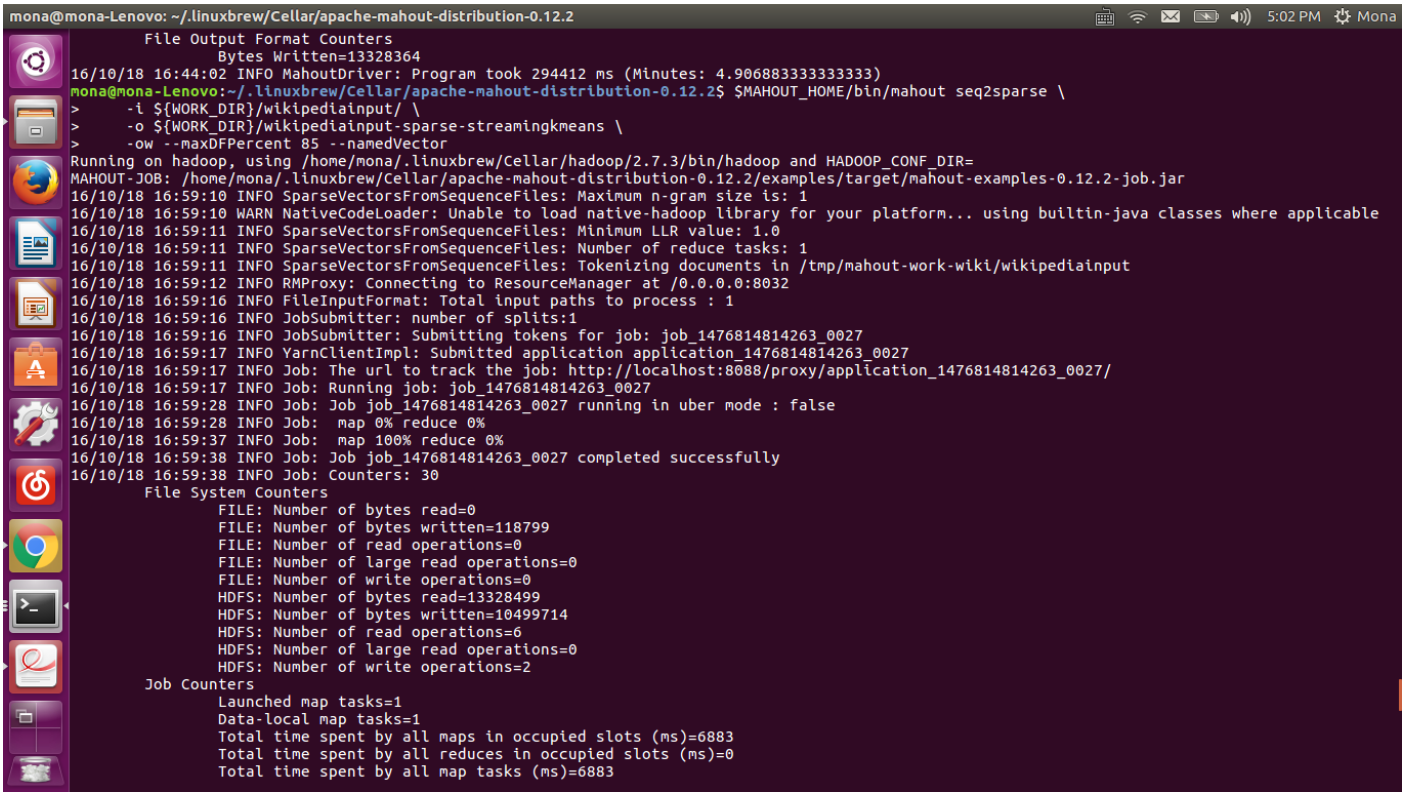
2.2.2 Creating sequence files from wikiXML

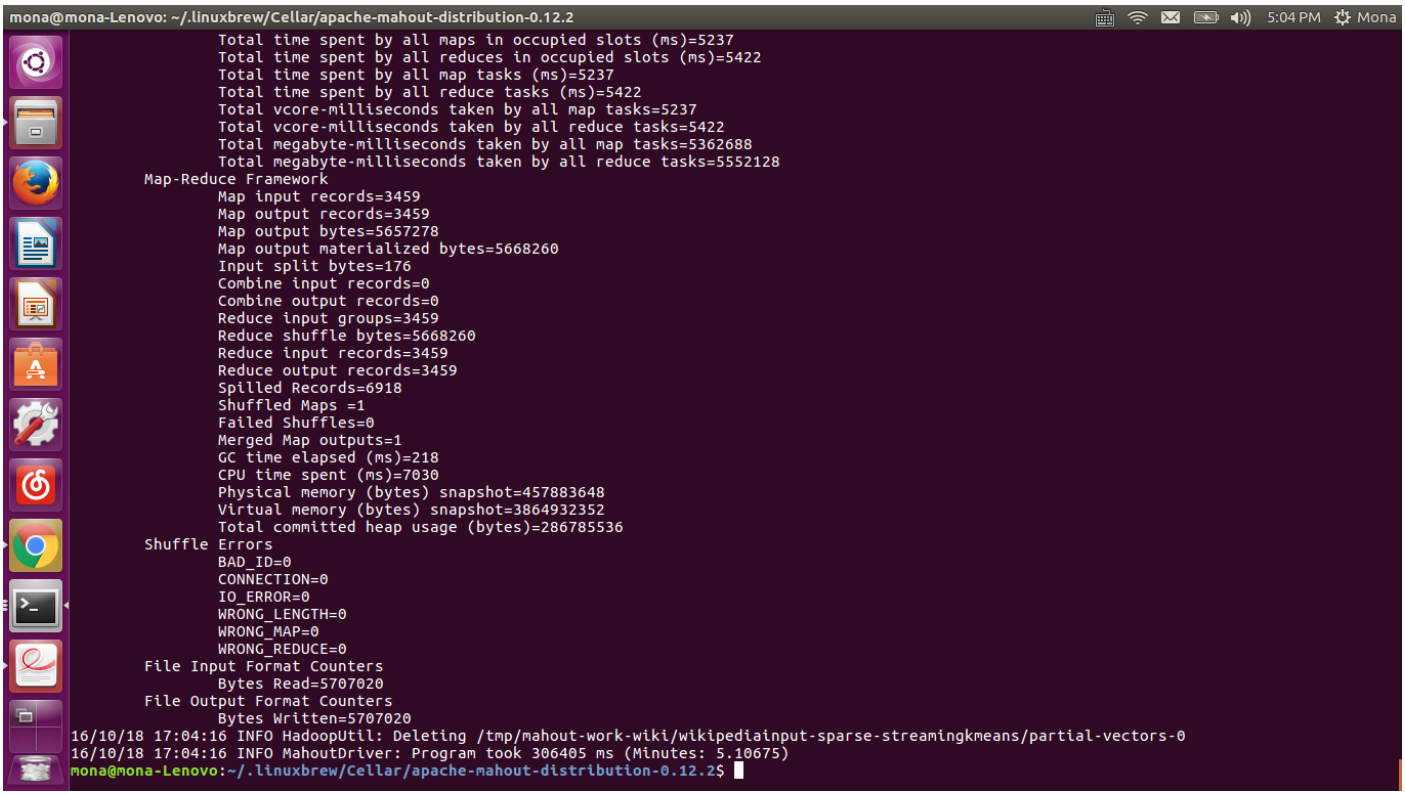
Use the categories "country".

```
echo "United States" > ${WORK_DIR}/categories.txt
$MAHOUT_HOME/bin/mahout seqwiki -c ${WORK_DIR}/categories.txt -i ${WORK_DI
R}/wikixml/enwiki-latest-pages-articles.xml -o ${WORK_DIR}/wikipediainput
```

2.2.3 Change to sparse

```
$MAHOUT_HOME/bin/mahout seq2sparse \
    -i ${WORK_DIR}/wikipediainput/ \
    -o ${WORK_DIR}/wikipediainput-sparse \
    -ow --maxDFPercent 85 --namedVector
```



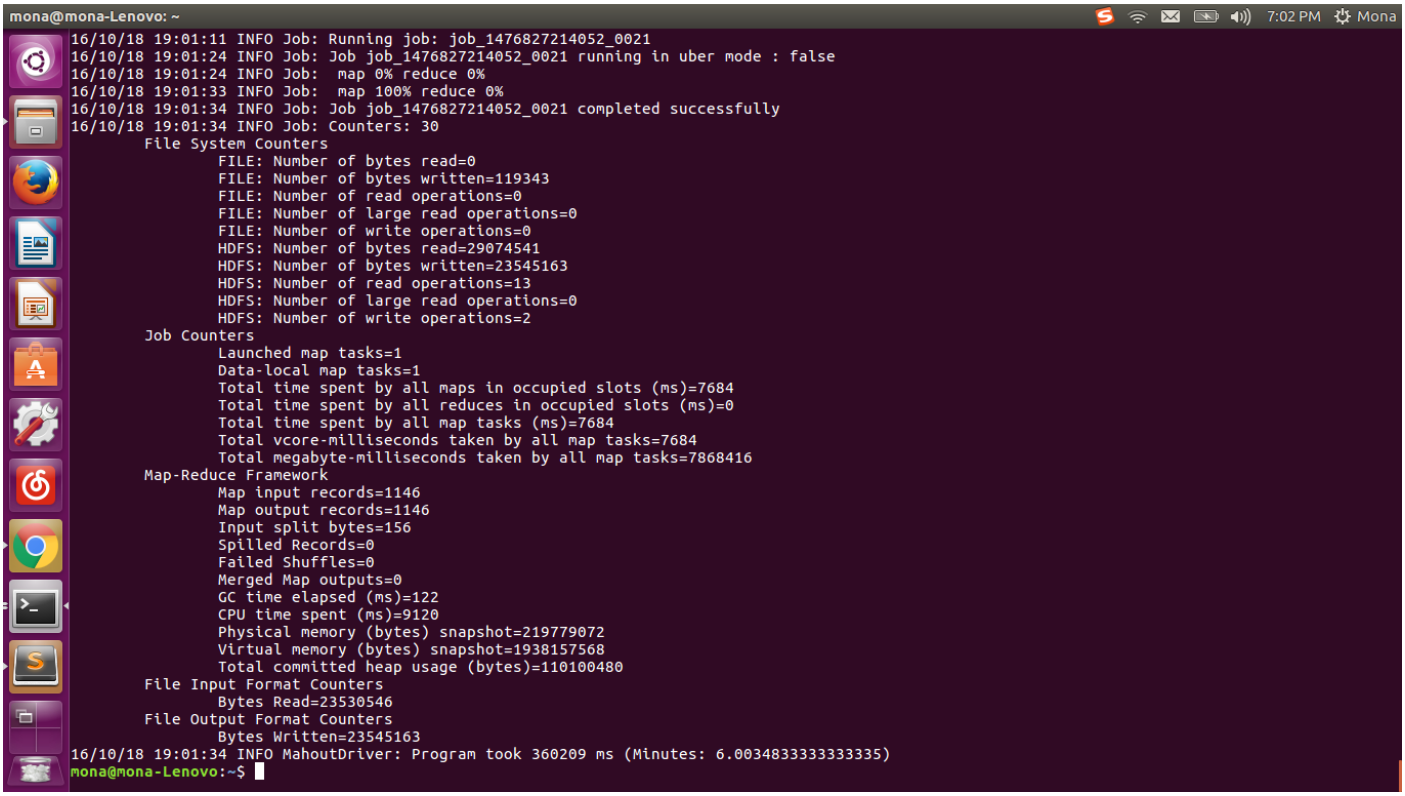


2.2.4 Using streamingkmeans

I decided to use streamingkmeans algorithm. The explain of the algorithm is here:
https://mahout.apache.org/users/clustering/streaming-k-means.html
(https://mahout.apache.org/users/clustering/streaming-k-means.html)

```
$MAHOUT_HOME/bin/mahout streamingkmeans \
    -i ${WORK_DIR}/wikipediainput-sparse/tfidf-vectors/ \
    --tempDir ${WORK_DIR}/tmp \
    -o ${WORK_DIR}/wikipedia-streamingkmeans \
    -sc org.apache.mahout.math.neighborhood.FastProjectionSearch \
    -dm org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure \
    -k 10 -km 100 -ow
```

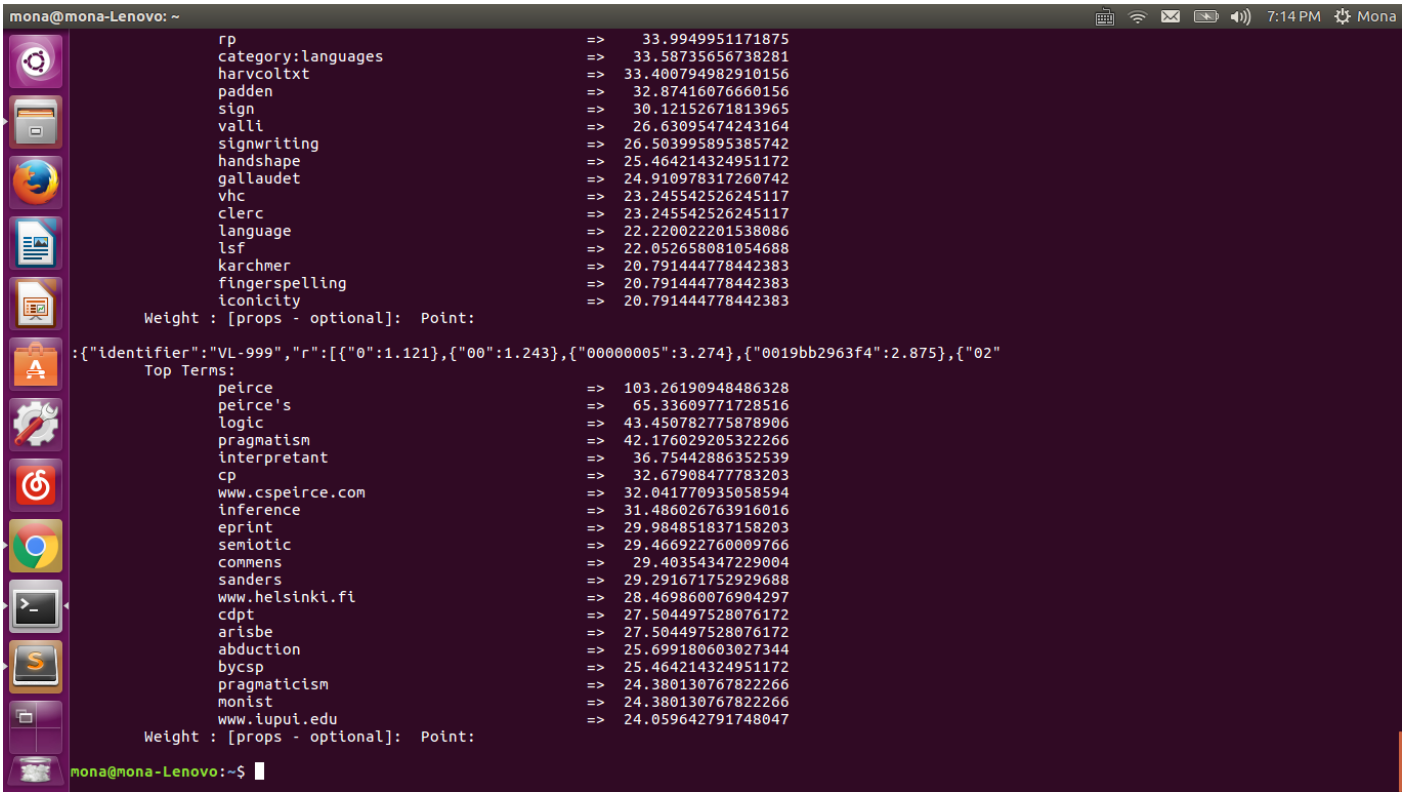
some details about the code above:
`-numClusters (-k)` k: The k in k-Means. Approximately this many clusters will be generated.
`-estimatedNumMapClusters (-km)` is estimatedNumMapClusters: The estimated number of clusters to use for the Map phase of the job when running StreamingKMeans. This should be around k * log(n), where k is the final number of clusters and n is the total number of data points to cluster.



2.2.5 Show the output

Use dumper as before to read the output

```
$MAHOUT_HOME/bin/mahout clusterdump \
    -i `$DFS -ls -d ${WORK_DIR}/wiki-kmeans/clusters-*-final | awk '{print
 $8}'` \
    -o ${WORK_DIR}/wiki-kmeans/clusterdump \
    -d ${WORK_DIR}/wikipediainput-sparse/dictionary.file-0 \
    -dt sequencefile -b 100 -n 20 --evaluate -dm org.apache.mahout.common.di
stance.EuclideanDistanceMeasure -sp 0 \
    --pointsDir ${WORK_DIR}/wiki-kmeans/clusteredPoints \
    && \
  cat ${WORK_DIR}/wiki-kmeans/clusterdump
```


mona@mona-Lenovo: ~
7:14 PM
Mona
rp => 33.9949951171875
category:languages => 33.58735656738281
harvcoltxt => 33.400794982910156
padden => 32.87416076660156
sign => 30.12152671813965
valli => 26.63095474243164
signwriting => 26.503995895385742
handshape => 25.464214324951172
gallaudet => 24.910978317260742
vhc => 23.245542526245117
clerc => 23.245542526245117
language => 22.220022201538086
lsf => 22.052658081054688
karchmer => 20.791444778442383
fingerspelling => 20.791444778442383
iconicity => 20.791444778442383
Weight : [props - optional]: Point:
:{"identifier":"VL-999","r":[{"0":1.121},{"00":1.243},{"00000005":3.274},{"0019bb2963f4":2.875},{"02"
Top Terms:
peirce => 103.26190948486328
peirce's => 65.33609771728516
logic => 43.450782775878906
pragmatism => 42.176029205322266
interpretant => 36.75442886352539
cp => 32.67908477783203
www.cspeirce.com => 32.041770935058594
inference => 31.486026763916016
eprint => 29.984851837158203
semiotic => 29.466922760009766
commens => 29.40354347229004
sanders => 29.291671752929688
www.helsinki.fi => 28.469860076904297
cdpt => 27.504497528076172
arisbe => 27.504497528076172
abduction => 25.699180603027344
bycsp => 25.464214324951172
pragmaticism => 24.380130767822266
monist => 24.380130767822266
www.iupui.edu => 24.059642791748047
Weight : [props - optional]: Point:
mona@mona-Lenovo:~$

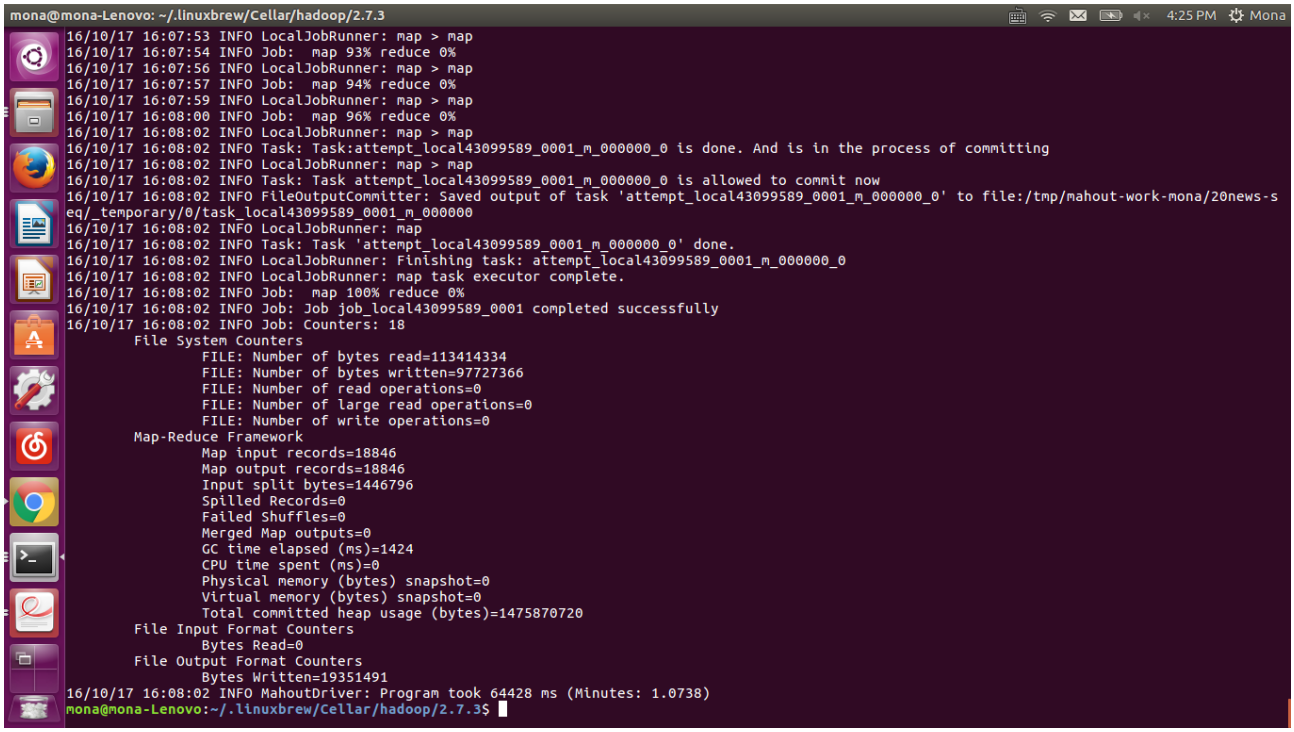
3 Classifier

3.1 End to end commands to build a CBayes model for 20 newsgroups

1. Download Dataset To begin, download the 20 newsgroups data set from this URL: http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate.tar.gz (http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate.tar.gz). This version of the data set splits training and test data by date and keeps all of the header lines.
2. Convert the full 20 newsgroups dataset into a < Text, Text > SequenceFile
   Use the code

   ```
   mahout seqdirectory -i ${WORK_DIR}/20news-all -o ${WORK_DIR}/20news-s
   eq -ow
   ```

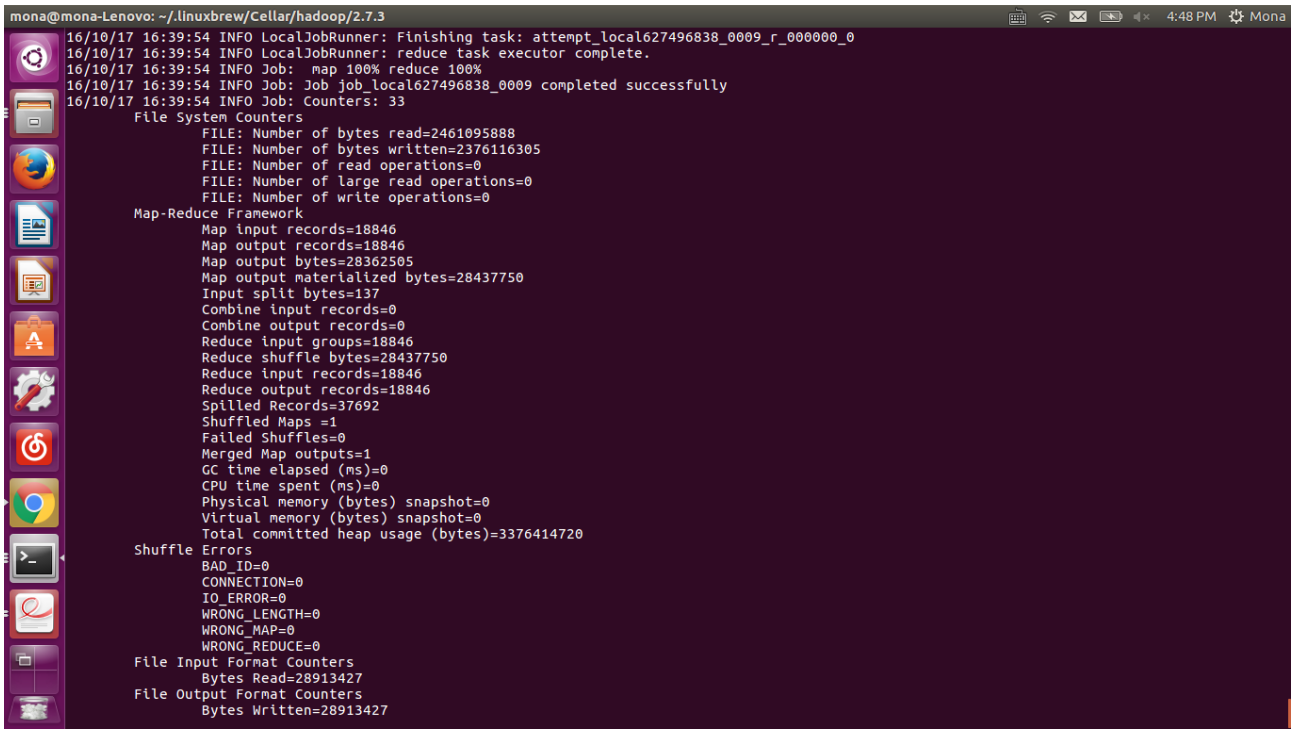
   Using `SequenceFileFromDirectory` will write the article in `Sequencefile` format.
   Above is a launch script for that. `-i` followed by the input file and `-o` followed by the outcome. To launch the file, we should set the `$JAVA_HOME` in environments.
   In mahout, `-ow` is used to denote whether or not tooverwrite the output folder.



3. Convert and preprocesses the dataset into a < Text, VectorWritable > SequenceFile containing term frequencies for each document.
   Using to convert data to vector.

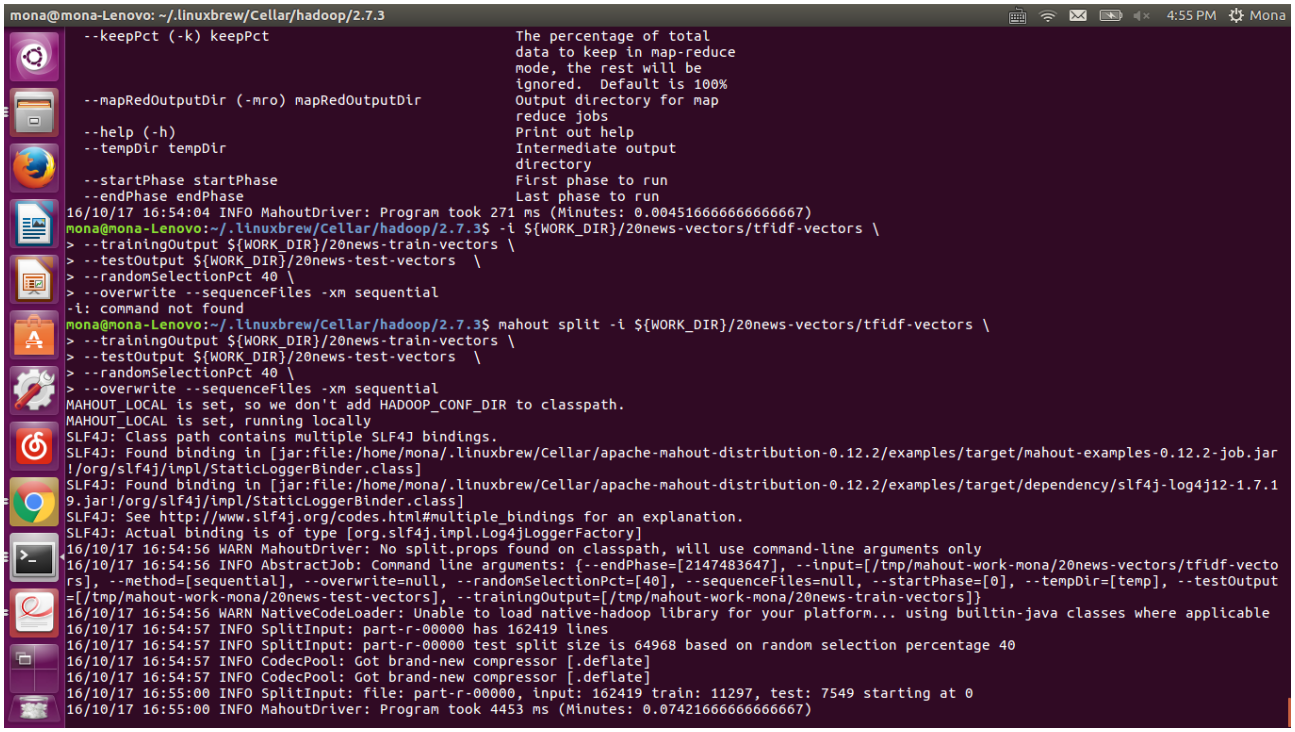
   ```
   mahout seq2sparse \
   -i ${WORK_DIR}/20news-seq \
   -o ${WORK_DIR}/20news-vectors \
   -lnorm \
   -nv \
   -wt tfidf
   ```

The meaning of the code above:
Weighting(String):`-wt` The weighting scheme to use: tf for termfrequency based weighting and tfidf for TFIDF based weighting. `tfidf` default.
The `seq2sparse` command in the Mahout launcher script reads the Reuters data from `SequenceFile` and writes the vector generated by the dictionary-based vectorizer to the output folder using the default options listed shortly in table 8.2 in *Mahout in Action* Page. 140. or (http://mahout.apache.org/users/basics/creating-vectors-from-text.html (http://mahout.apache.org/users/basics/creating-vectors-from-text.html))



4. Split the preprocessed dataset into training and testing sets

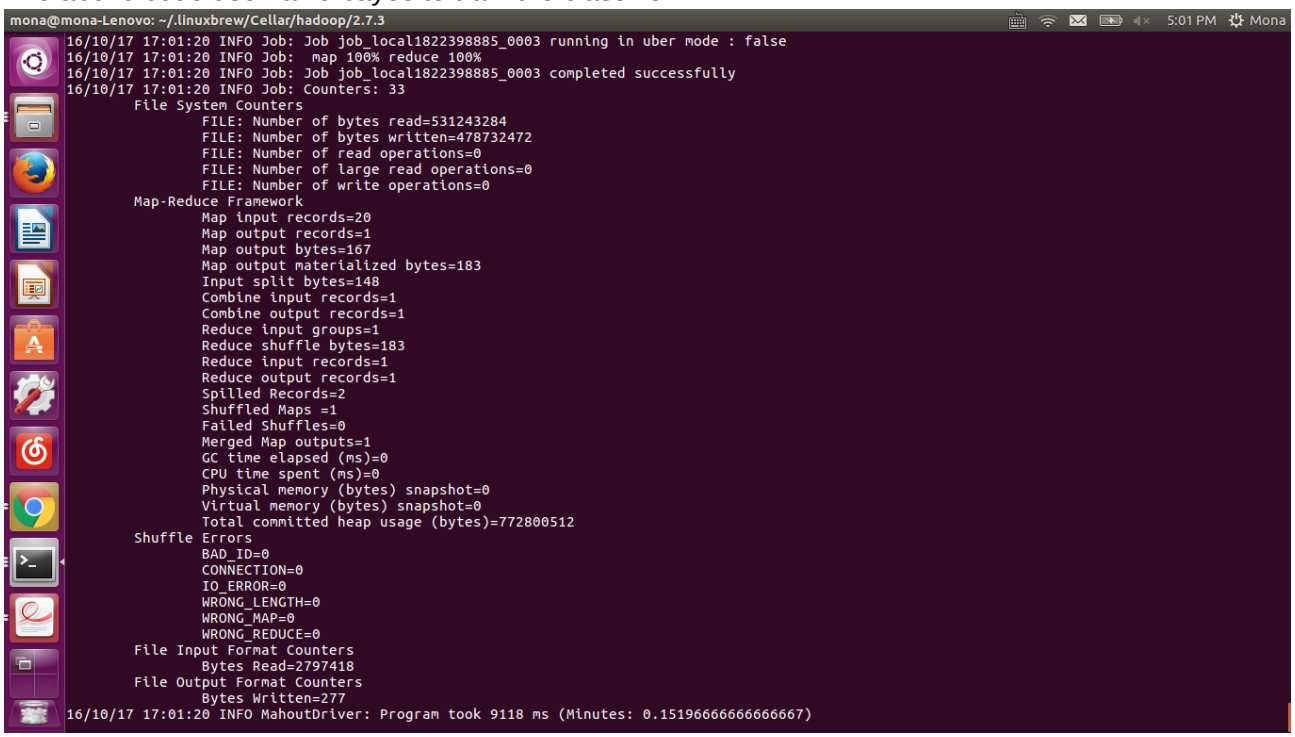
```
mahout split -i ${WORK_DIR}/20news-vectors/tfidf-vectors \
--trainingOutput ${WORK_DIR}/20news-train-vectors \
--testOutput ${WORK_DIR}/20news-test-vectors  \
--randomSelectionPct 40\
--overwrite --sequenceFiles -xm sequential
```



5. Train the classifier

```
mahout trainnb -i {WORK_DIR}/20news-train-vectors -o {WORK_DIR}/model
 -li {WORK_DIR}/labelindex -ow -c
```

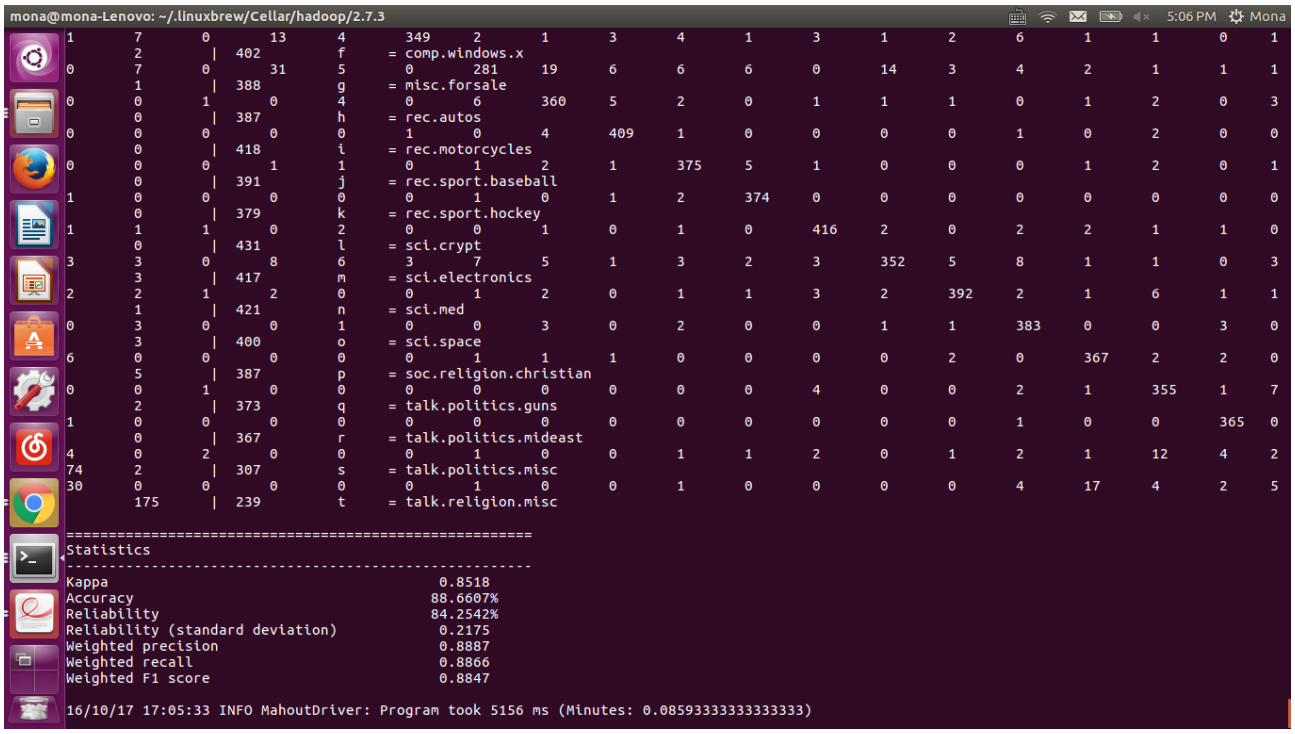
The above code use navie bayes to train the classifier.



6. Test the classifier

```
mahout testnb -i ${WORK_DIR}/20news-test-vectors -m ${WORK_DIR}/model
 -l ${WORK_DIR}/labelindex -ow -o ${WORK_DIR}/20news-testing -c
```


```
16/10/17 17:05:33 INFO TestNaiveBayesDriver: Complementary Results:
=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :       6693       88.6607%
Incorrectly Classified Instances        :        856       11.3393%
Total Classified Instances              :       7549

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       h       i       j       k       l       m       n       o       p       q       r       s
t       <--Classified as
269     0       0       0       1       0       0       0       0       0       0       2       0       1       4       3       0       0       1
        8       |  289          a     = alt.atheism
1       305     4       17      6       15      11      2       1       1       2       8       4       3       5       3       3       0       1
        3       |  395          b     = comp.graphics
1       24      254     56      15      8       13      4       3       5       4       7       3       2       3       4       0       1       4
        1       |  412          c     = comp.os.ms-windows.misc
1       7       5       297     9       4       8       1       1       1       0       3       3       2       2       0       4       2       0
        0       |  350          d     = comp.sys.ibm.pc.hardware
2       4       5       10      341     0       6       4       4       3       1       0       7       0       4       1       1       2       0
        1       |  396          e     = comp.sys.mac.hardware
1       7       0       13      4       349     2       1       3       4       1       3       1       2       6       1       1       0       1
        2       |  402          f     = comp.windows.x
0       7       0       31      5       0       281     19      6       6       6       0       14      3       4       2       1       1       1
        1       |  388          g     = misc.forsale
0       0       1       0       4       0       6       360     5       2       0       1       1       1       0       1       2       0       3
        0       |  387          h     = rec.autos
0       0       0       0       0       1       0       4       409     1       0       0       0       0       1       0       2       0       0
        0       |  418          i     = rec.motorcycles
0       0       0       1       1       0       1       2       1       375     5       1       0       0       0       1       2       0       1
        0       |  391          j     = rec.sport.baseball
1       0       0       0       0       0       1       0       1       2       374     0       0       0       0       0       0       0       0
        0       |  379          k     = rec.sport.hockey
1       1       1       0       2       0       0       1       0       1       0       416     2       0       2       2       1       1       0
        0       |  431          l     = sci.crypt
3       3       0       8       6       3       7       5       1       3       2       3       352     5       8       1       1       0       3
        3       |  417          m     = sci.electronics
2       2       1       2       0       0       1       2       0       1       1       3       2       392     2       1       6       1       1
        1       |  421          n     = sci.med
0       3       0       0       1       0       0       3       0       2       0       0       1       1       383     0       0       3       0
        3       |  400          o     = sci.space
```

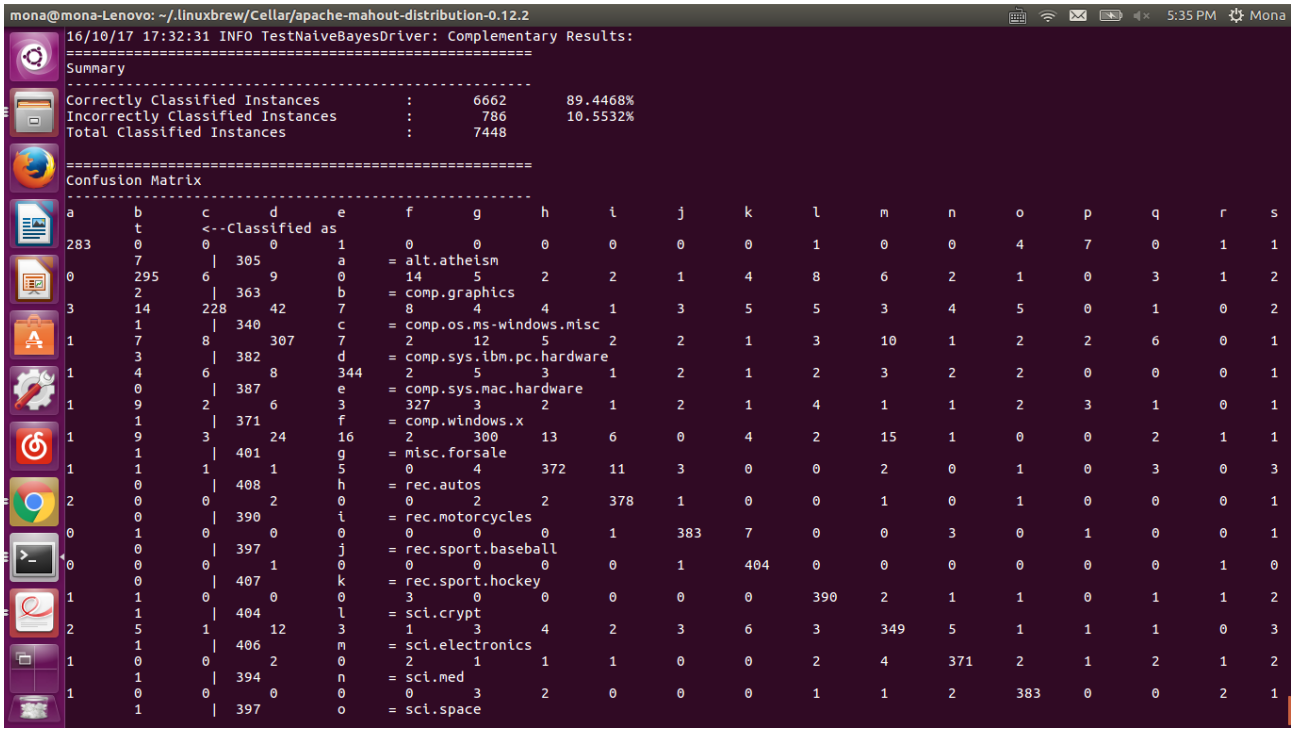



```
1       7       0       13      4       349     2       1       3       4       1       3       1       2       6       1       1       0       1
        2       |  402          f     = comp.windows.x
0       7       0       31      5       0       281     19      6       6       6       0       14      3       4       2       1       1       1
        1       |  388          g     = misc.forsale
0       0       1       0       4       0       6       360     5       2       0       1       1       1       0       1       2       0       3
        0       |  387          h     = rec.autos
0       0       0       0       0       1       0       4       409     1       0       0       0       0       1       0       2       0       0
        0       |  418          i     = rec.motorcycles
0       0       0       1       1       0       1       2       1       375     5       1       0       0       0       1       2       0       1
        0       |  391          j     = rec.sport.baseball
1       0       0       0       0       0       1       0       1       2       374     0       0       0       0       0       0       0       0
        0       |  379          k     = rec.sport.hockey
1       1       1       0       2       0       0       1       0       1       0       416     2       0       2       2       1       1       0
        0       |  431          l     = sci.crypt
3       3       0       8       6       3       7       5       1       3       2       3       352     5       8       1       1       0       3
        3       |  417          m     = sci.electronics
2       2       1       2       0       0       1       2       0       1       1       3       2       392     2       1       6       1       1
        1       |  421          n     = sci.med
0       3       0       0       1       0       0       3       0       2       0       0       1       1       383     0       0       3       0
        3       |  400          o     = sci.space
6       0       0       0       0       0       1       1       1       0       0       0       0       2       0       367     2       2       0
        5       |  387          p     = soc.religion.christian
0       0       1       0       0       0       0       0       0       0       0       4       0       0       2       1       355     1       7
        2       |  373          q     = talk.politics.guns
1       0       0       0       0       0       0       0       0       0       0       0       0       0       1       0       0       365     0
        0       |  367          r     = talk.politics.mideast
4       0       2       0       0       0       1       0       1       1       2       0       1       2       1       12      4       2
74      2       |  307          s     = talk.politics.misc
30      0       0       0       0       0       1       0       0       1       0       0       0       0       4       17      4       2       5
        175     |  239          t     = talk.religion.misc

=======================================================
Statistics
-------------------------------------------------------
Kappa                                       0.8518
Accuracy                                   88.6607%
Reliability                                84.2542%
Reliability (standard deviation)            0.2175
Weighted precision                          0.8887
Weighted recall                             0.8866
Weighted F1 score                           0.8847

16/10/17 17:05:33 INFO MahoutDriver: Program took 5156 ms (Minutes: 0.08593333333333333)
```

3.2 Compare different algorithms

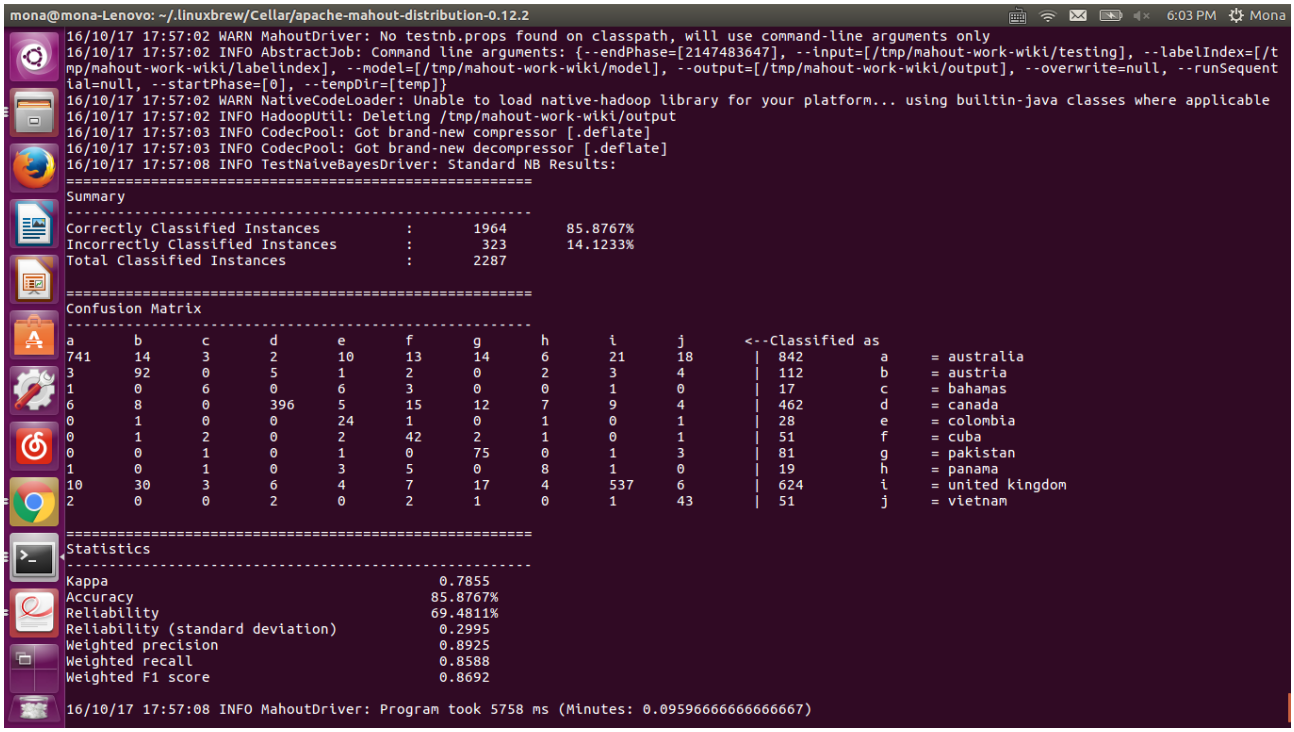
1. Complement Naive Bayes

Statistics	
Kappa	0.859
Accuracy	89.4468%
Reliability	84.8085%
Reliability (standard deviation)	0.2146
Weighted precision	0.8944
Weighted recall	0.8945
Weighted F1 score	0.8928



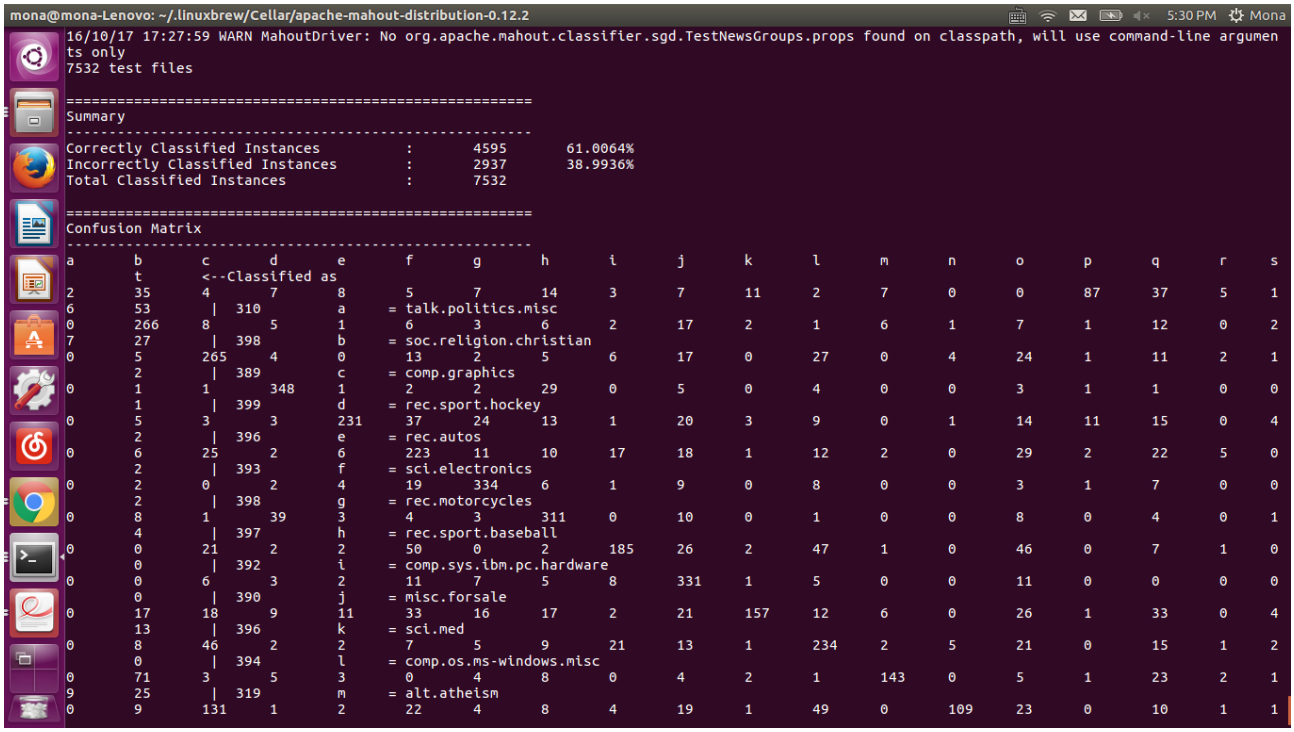
2. Naive Bayes

Statistics	
Kappa	0.879
Accuracy	90.7368%
Reliability	86.2658%
Reliability (standard deviation)	0.2144
Weighted precision	0.9117
Weighted recall	0.9074
Weighted F1 score	0.9068



3. Stochastic Gradient Descent

```
Statistics
Kappa                                       0.5657
Accuracy                                   61.0064%
Reliability                                 56.815%
Reliability (standard deviation)            0.2571
Weighted precision                          0.6617
Weighted recall                             0.6101
Weighted F1 score                           0.5909
Log-likelihood                mean      :    -1.6909
                             25%-ile    :    -2.3882
                             75%-ile    :    -0.3078
```




```
16/10/17 17:27:59 WARN MahoutDriver: No org.apache.mahout.classifier.sgd.TestNewsGroups.props found on classpath, will use command-line argumen
ts only
7532 test files

=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :       4595       61.0064%
Incorrectly Classified Instances        :       2937       38.9936%
Total Classified Instances              :       7532

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       h       i       j       k       l       m       n       o       p       q       r       s
t       <--Classified as
2       35      4       7       8       5       7       14      3       7       11      2       7       0       0       87      37      5       1
6       53       |  310        a     = talk.politics.misc
0       266     8       5       1       6       3       6       2       17      2       1       6       1       7       1       12      0       2
7       27       |  398        b     = soc.religion.christian
0       5       265     4       0       13      2       5       6       17      0       27      0       4       24      1       11      2       1
        2        |  389        c     = comp.graphics
0       1       1       348     1       2       2       29      0       5       0       4       0       0       3       1       1       0       0
        1        |  399        d     = rec.sport.hockey
0       5       3       3       231     37      24      13      1       20      3       9       0       1       14      11      15      0       4
        2        |  396        e     = rec.autos
0       6       25      2       6       223     11      10      17      18      1       12      2       0       29      2       22      5       0
        2        |  393        f     = sci.electronics
0       2       0       2       4       19      334     6       1       9       0       8       0       0       3       1       7       0       0
        2        |  398        g     = rec.motorcycles
0       8       1       39      3       4       3       311     0       10      0       1       0       0       8       0       4       0       1
        4        |  397        h     = rec.sport.baseball
0       0       21      2       2       50      0       2       185     26      2       47      1       0       46      0       7       1       0
        0        |  392        i     = comp.sys.ibm.pc.hardware
0       0       6       3       2       11      7       5       8       331     1       5       0       0       11      0       0       0       0
        0        |  390        j     = misc.forsale
0       17      18      9       11      33      16      17      2       21      157     12      6       0       26      1       33      0       4
        13       |  396        k     = sci.med
0       8       46      2       2       7       5       9       21      13      1       234     2       5       21      0       15      1       2
        0        |  394        l     = comp.os.ms-windows.misc
0       71      3       5       3       0       4       8       0       4       2       1       143     0       5       1       23      2       1
9       25       |  319        m     = alt.atheism
0       9       131     1       2       22      4       8       4       19      1       49      0       109     23      0       10      1       1
```

Compare the above outputs, the Naive Bayes performance best accurancy of classification.

3.3 Wikipedia dataset

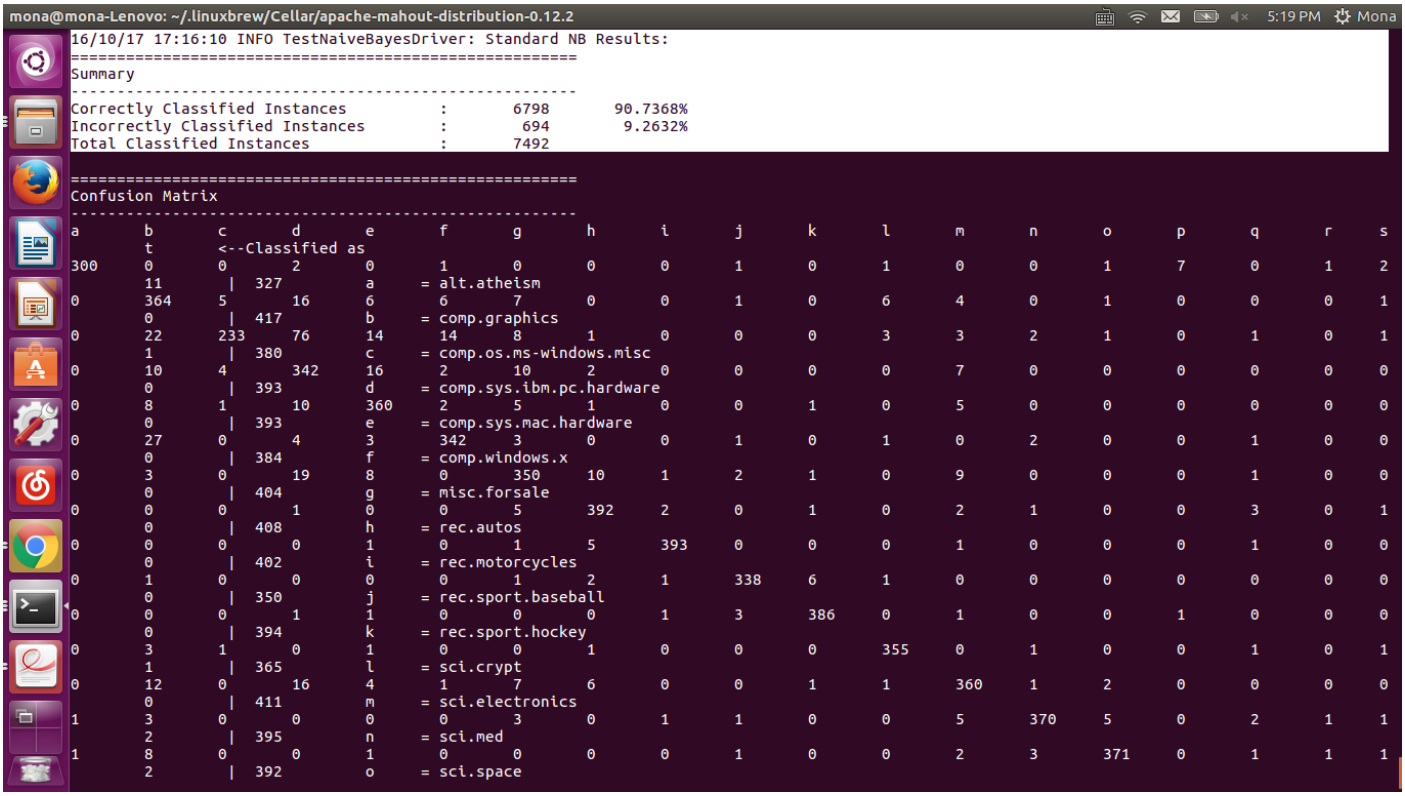
Download data from the URL: https://dumps.wikimedia.org/enwiki/latest/ (https://dumps.wikimedia.org/enwiki/latest/)
We use the middle size of data: https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles10.xml-p002336425p003046511.bz2 (https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles10.xml-p002336425p003046511.bz2)
Unzip the data and do the processing like above. The processing is almost same as the 20news data set. The only different is the source data is xml format, which we should use a different class to pre-processing those data. Use the code

```
$MAHOUT_HOME/bin/mahout seqwiki -c ${WORK_DIR}/country.txt \
-i ${WORK_DIR}/wikixml/enwiki-latest-pages-articles.xml \
-o ${WORK_DIR}/wikipediainput
```

rather than seqditectory to the wikipedia XML file in order to convert the data set into < text, text> format. Creating sequence files from wikiXML.

- Outputs

TestNaiveBayesDriver: Standard NB Results




```
16/10/17 17:16:10 INFO TestNaiveBayesDriver: Standard NB Results:
=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :       6798       90.7368%
Incorrectly Classified Instances        :        694        9.2632%
Total Classified Instances              :       7492

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       h       i       j       k       l       m       n       o       p       q       r       s
        t       <--Classified as
300     0       0       2       0       1       0       0       0       1       0       1       0       0       1       7       0       1       2
        11       |  327         a     = alt.atheism
0       364     5       16      6       6       7       0       0       1       0       6       4       0       1       0       0       0       1
        0        |  417         b     = comp.graphics
0       22      233     76      14      14      8       1       0       0       0       3       3       2       1       0       1       0       1
        1        |  380         c     = comp.os.ms-windows.misc
0       10      4       342     16      2       10      2       0       0       0       0       7       0       0       0       0       0       0
        0        |  393         d     = comp.sys.ibm.pc.hardware
0       8       1       10      360     2       5       1       0       0       1       0       5       0       0       0       0       0       0
        0        |  393         e     = comp.sys.mac.hardware
0       27      0       4       3       342     3       0       0       1       0       1       0       2       0       0       1       0       0
        0        |  384         f     = comp.windows.x
0       3       0       19      8       0       350     10      1       2       1       0       9       0       0       0       1       0       0
        0        |  404         g     = misc.forsale
0       0       0       1       0       0       5       392     2       0       1       0       2       1       0       0       3       0       1
        0        |  408         h     = rec.autos
0       0       0       0       1       0       1       5       393     0       0       0       1       0       0       0       1       0       0
        0        |  402         i     = rec.motorcycles
0       1       0       0       0       0       1       2       1       338     6       1       0       0       0       0       0       0       0
        0        |  350         j     = rec.sport.baseball
0       0       0       1       1       0       0       0       1       3       386     0       1       0       0       1       0       0       0
        0        |  394         k     = rec.sport.hockey
0       3       1       0       1       0       0       1       0       0       0       355     0       1       0       0       1       0       1
        1        |  365         l     = sci.crypt
0       12      0       16      4       1       7       6       0       0       1       1       360     1       2       0       0       0       0
        0        |  411         m     = sci.electronics
1       3       0       0       0       0       3       0       1       1       0       0       5       370     5       0       2       1       1
        2        |  395         n     = sci.med
1       8       0       0       1       0       0       0       0       1       0       0       2       3       371     0       1       1       1
        2        |  392         o     = sci.space
```

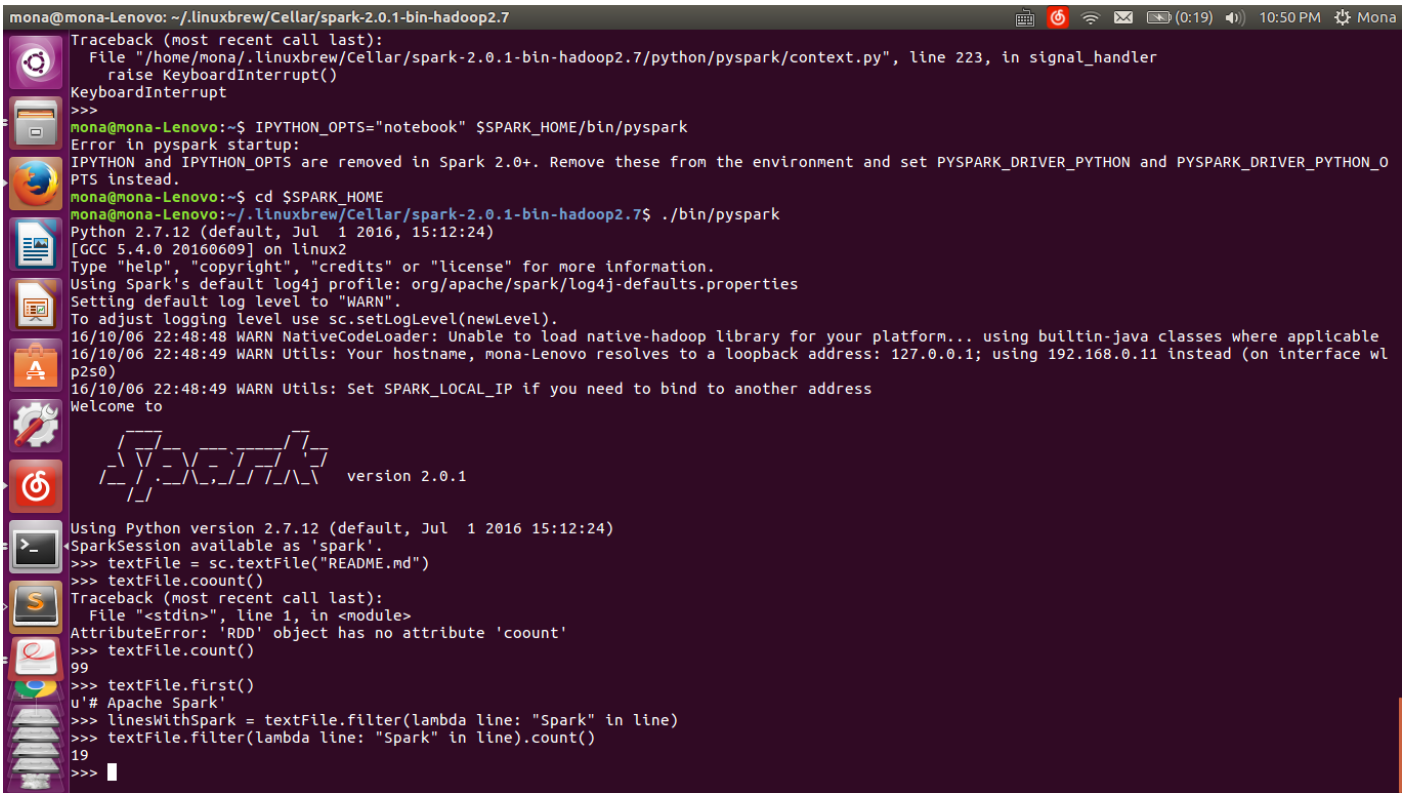
TestNaiveBayesDriver: Complementary Results


```
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=837287936
```

Spark

1 Installation

Just download Spark and unzip it in a folder. Set the environment varibles. Then go to `bin/pyspark` to start spark.
Like the screen shot below. It counts the words of `readme.md` from the spark root file.

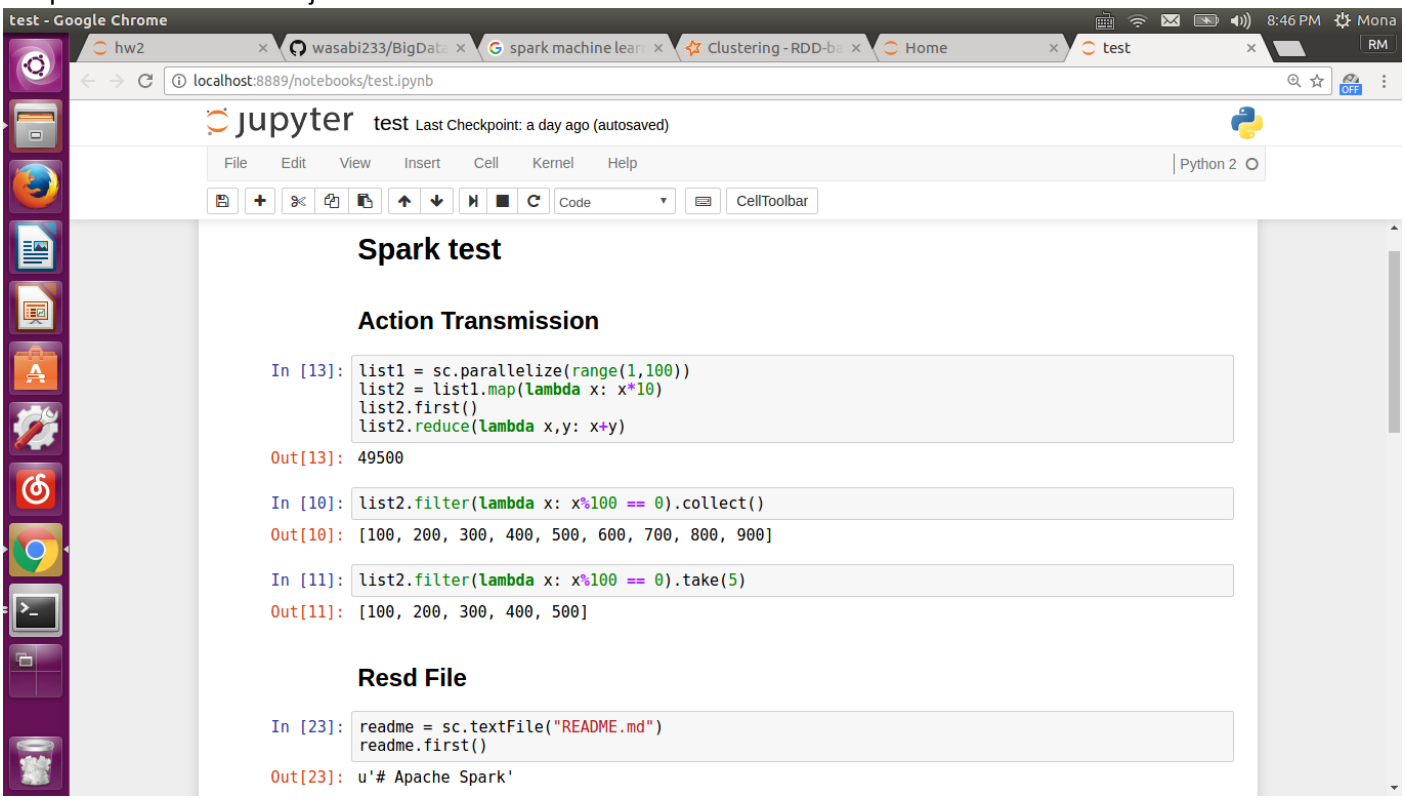


2 MLlib

As I already installed ipython notebook before, which is jupyter notebook now. I open the pyspark shell from ipython notebook for convenience.
In the spark root folder and type

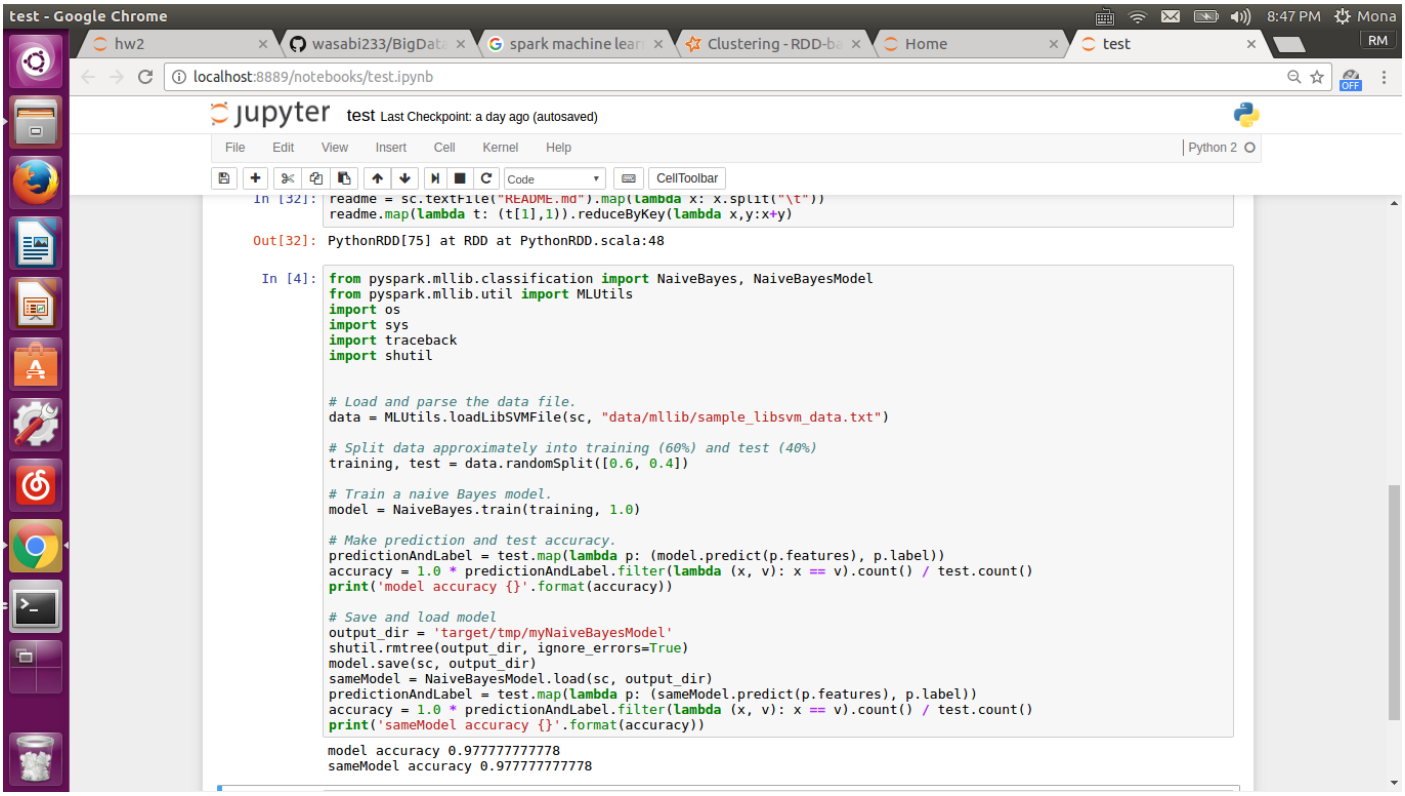
```
PYSPARK_DRIVER_PYTHON=ipython PYSPARK_DRIVER_PYTHON_OPTS="notebook" ./bin/pyspark
```

to open the window. It is just like that. I tried some functions on it. Here are the screen shots.



2.1 Classification using MLlib

NaiveBayes implements multinomial naive Bayes. It takes an RDD of LabeledPoint and an optionally smoothing parameter lambda as input, and output a NaiveBayesModel, which can be used for evaluation and prediction.


test - Google Chrome
8:47 PM
Mona
hw2
wasabi233/BigData
spark machine learn
Clustering - RDD-ba
Home
test
localhost:8889/notebooks/test.ipynb
jupyter
test Last Checkpoint: a day ago (autosaved)
File Edit View Insert Cell Kernel Help
Python 2
Code
CellToolbar
In [32]:
readme = sc.textFile("README.md").map(lambda x: x.split("\t"))
readme.map(lambda t: (t[1],1)).reduceByKey(lambda x,y:x+y)
Out[32]:
PythonRDD[75] at RDD at PythonRDD.scala:48
In [4]:
from pyspark.mllib.classification import NaiveBayes, NaiveBayesModel
from pyspark.mllib.util import MLUtils
import os
import sys
import traceback
import shutil
# Load and parse the data file.
data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")
# Split data approximately into training (60%) and test (40%)
training, test = data.randomSplit([0.6, 0.4])
# Train a naive Bayes model.
model = NaiveBayes.train(training, 1.0)
# Make prediction and test accuracy.
predictionAndLabel = test.map(lambda p: (model.predict(p.features), p.label))
accuracy = 1.0 * predictionAndLabel.filter(lambda (x, v): x == v).count() / test.count()
print('model accuracy {}'.format(accuracy))
# Save and load model
output_dir = 'target/tmp/myNaiveBayesModel'
shutil.rmtree(output_dir, ignore_errors=True)
model.save(sc, output_dir)
sameModel = NaiveBayesModel.load(sc, output_dir)
predictionAndLabel = test.map(lambda p: (sameModel.predict(p.features), p.label))
accuracy = 1.0 * predictionAndLabel.filter(lambda (x, v): x == v).count() / test.count()
print('sameModel accuracy {}'.format(accuracy))
model accuracy 0.977777777778
sameModel accuracy 0.977777777778