Spark ALS on EMR

In this document I'm recording critical steps on achieving movie recommendation on spark on AWS EC2 instances. Totally there are 3 primary steps as following:

1.Set up AWS EMR

2. Go to Zeppelin on port 8890

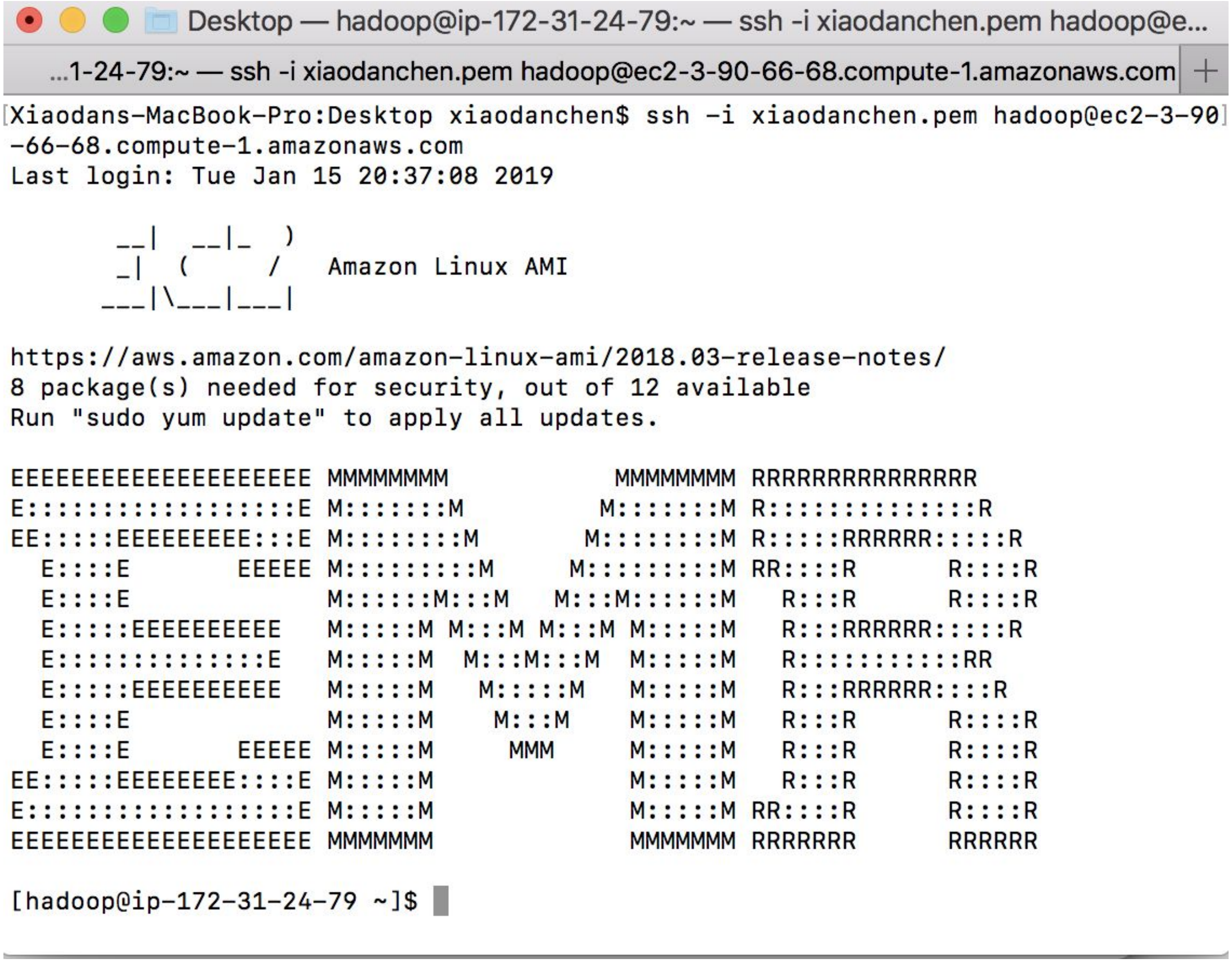
3. Achieving movie recommendation with Spark Mllib ALS algorithm

Here we go.

****

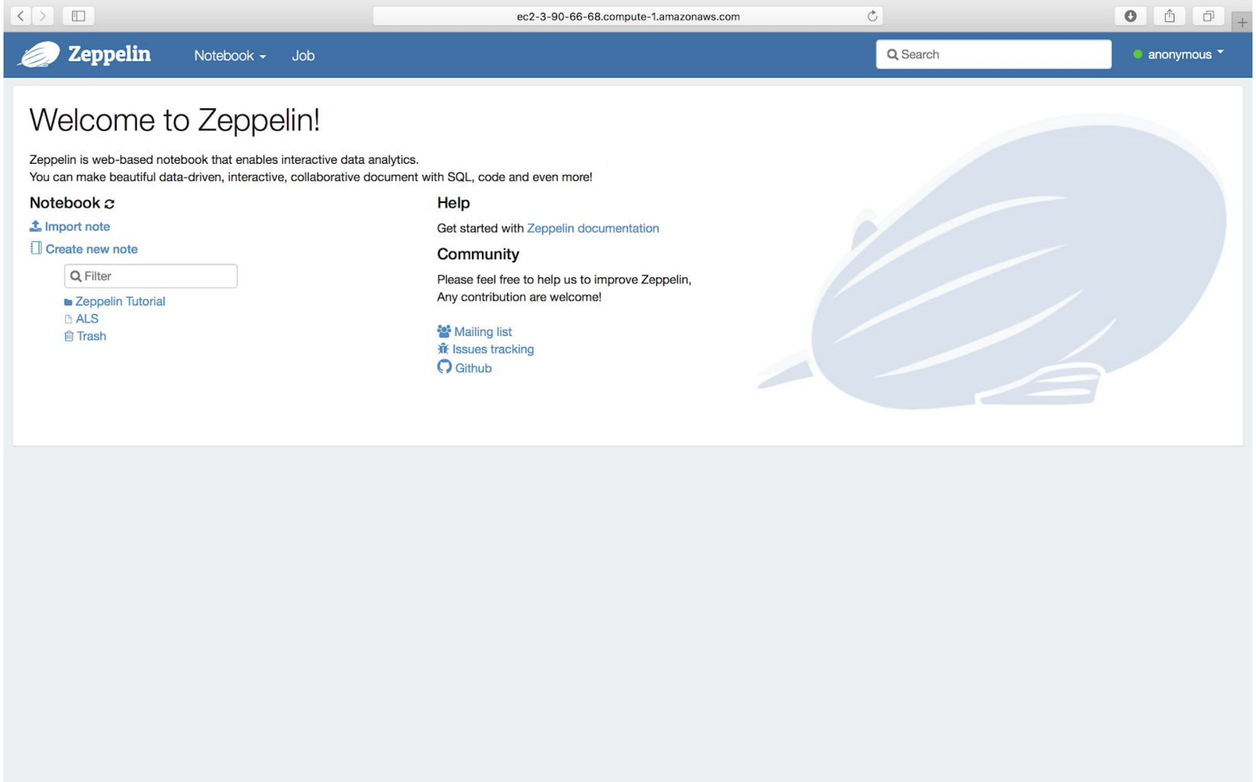
1.Set up AWS EMR

1.1 set AWS EMR and connect it in terminal


Desktop — hadoop@ip-172-31-24-79:~ — ssh -i xiaodanchen.pem hadoop@e...
...1-24-79:~ — ssh -i xiaodanchen.pem hadoop@ec2-3-90-66-68.compute-1.amazonaws.com
+

```
[Xiaodans-MacBook-Pro:Desktop xiaodanchen$ ssh -i xiaodanchen.pem hadoop@ec2-3-90]
-66-68.compute-1.amazonaws.com
Last login: Tue Jan 15 20:37:08 2019

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
8 package(s) needed for security, out of 12 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-24-79 ~]$
```

2. Go to Zeppelin on port 8890



Welcome to Zeppelin!

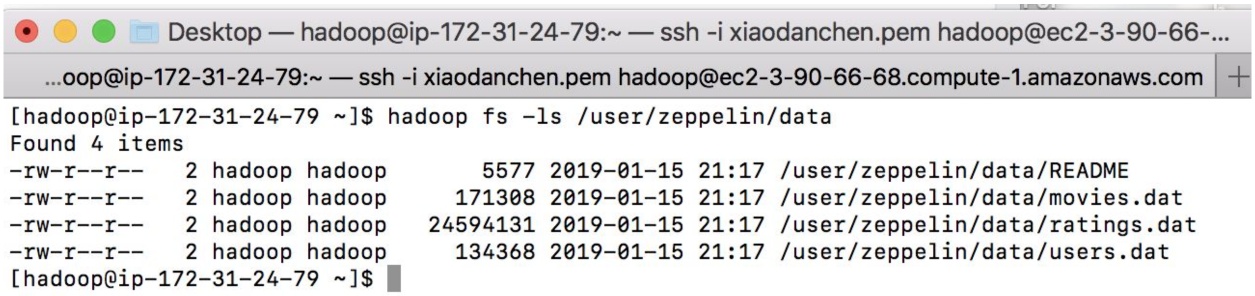
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Get started with Zeppelin documentation

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

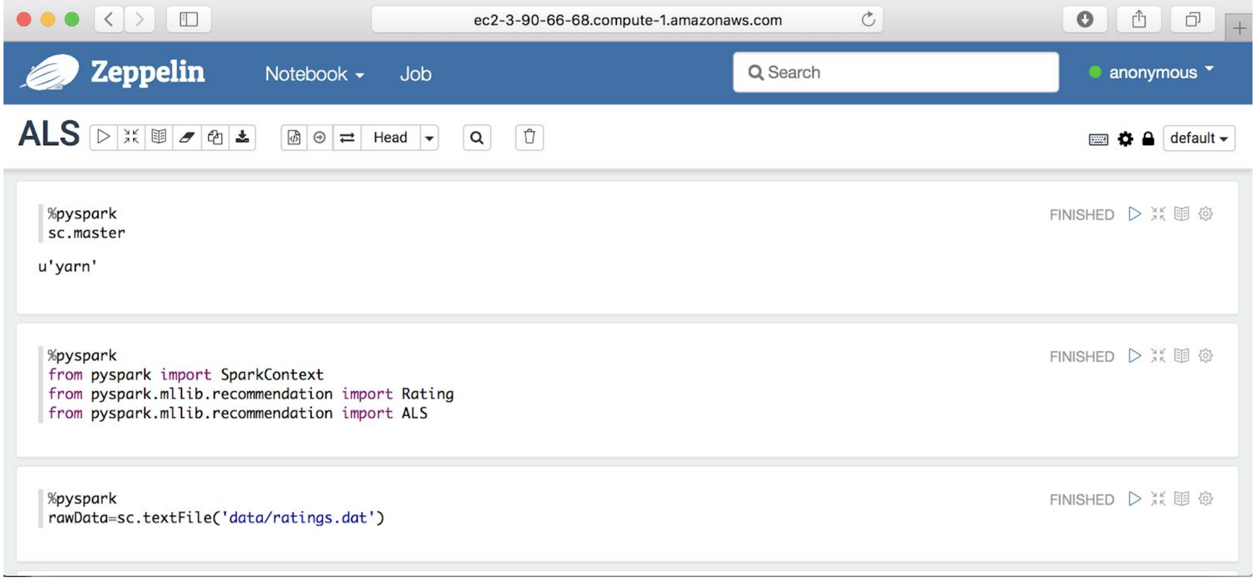
3. Achieving movie recommendation with Spark Mllib ALS algorithm

3.1 download data and save it on HDFS


Desktop — hadoop@ip-172-31-24-79:~ — ssh -i xiaodanchen.pem hadoop@ec2-3-90-66-...
...oop@ip-172-31-24-79:~ — ssh -i xiaodanchen.pem hadoop@ec2-3-90-66-68.compute-1.amazonaws.com

```
[hadoop@ip-172-31-24-79 ~]$ hadoop fs -ls /user/zeppelin/data
Found 4 items
-rw-r--r--   2 hadoop hadoop       5577 2019-01-15 21:17 /user/zeppelin/data/README
-rw-r--r--   2 hadoop hadoop     171308 2019-01-15 21:17 /user/zeppelin/data/movies.dat
-rw-r--r--   2 hadoop hadoop   24594131 2019-01-15 21:17 /user/zeppelin/data/ratings.dat
-rw-r--r--   2 hadoop hadoop     134368 2019-01-15 21:17 /user/zeppelin/data/users.dat
[hadoop@ip-172-31-24-79 ~]$ 
```

3.2 read data from HDFS

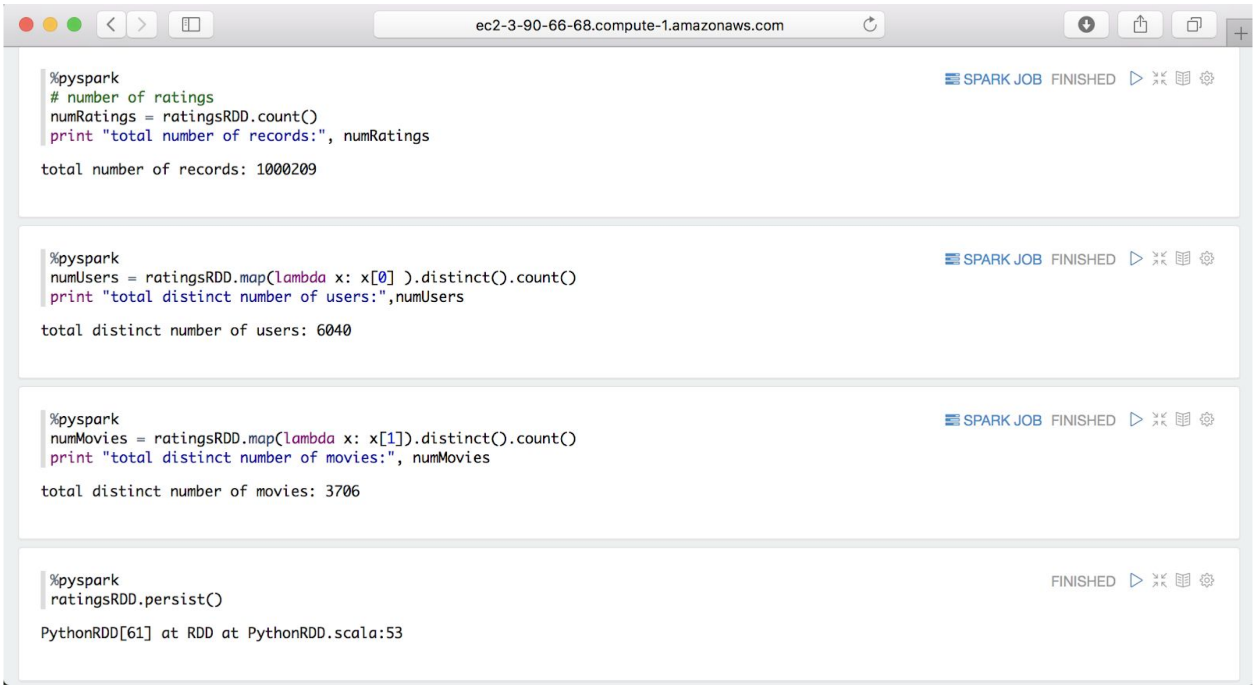


```
%pyspark
sc.master

u'yarn'
```

```
%pyspark
from pyspark import SparkContext
from pyspark.mllib.recommendation import Rating
from pyspark.mllib.recommendation import ALS
```

```
%pyspark
rawData=sc.textFile('data/ratings.dat')
```

3.3 Data stats



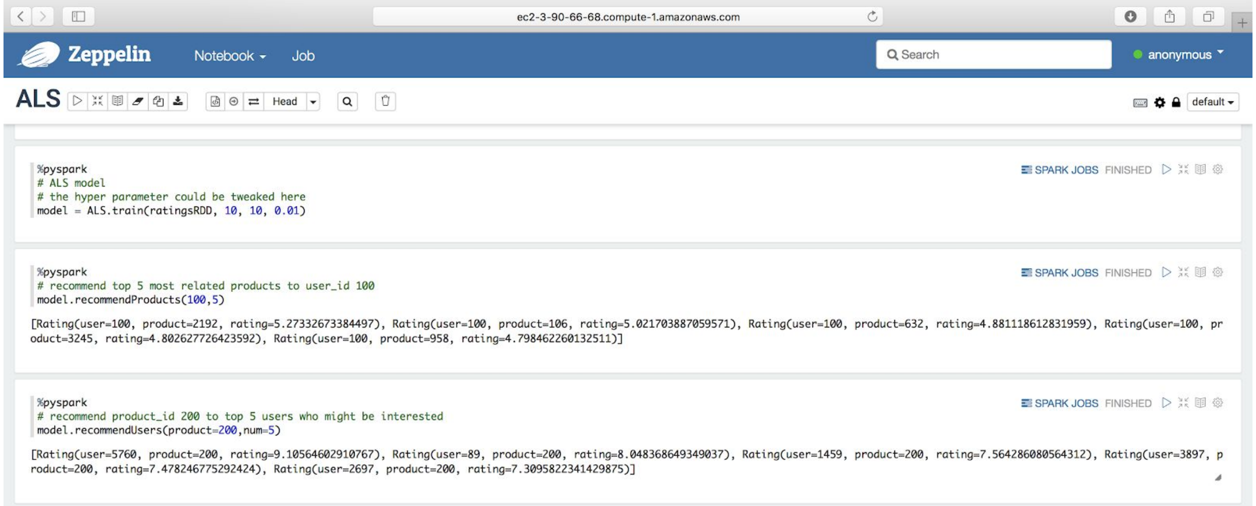
```
%pyspark
# number of ratings
numRatings = ratingsRDD.count()
print "total number of records:", numRatings

total number of records: 1000209
```

```
%pyspark
numUsers = ratingsRDD.map(lambda x: x[0] ).distinct().count()
print "total distinct number of users:",numUsers

total distinct number of users: 6040
```

```
%pyspark
numMovies = ratingsRDD.map(lambda x: x[1]).distinct().count()
print "total distinct number of movies:", numMovies

total distinct number of movies: 3706
```

```
%pyspark
ratingsRDD.persist()

PythonRDD[61] at RDD at PythonRDD.scala:53
```

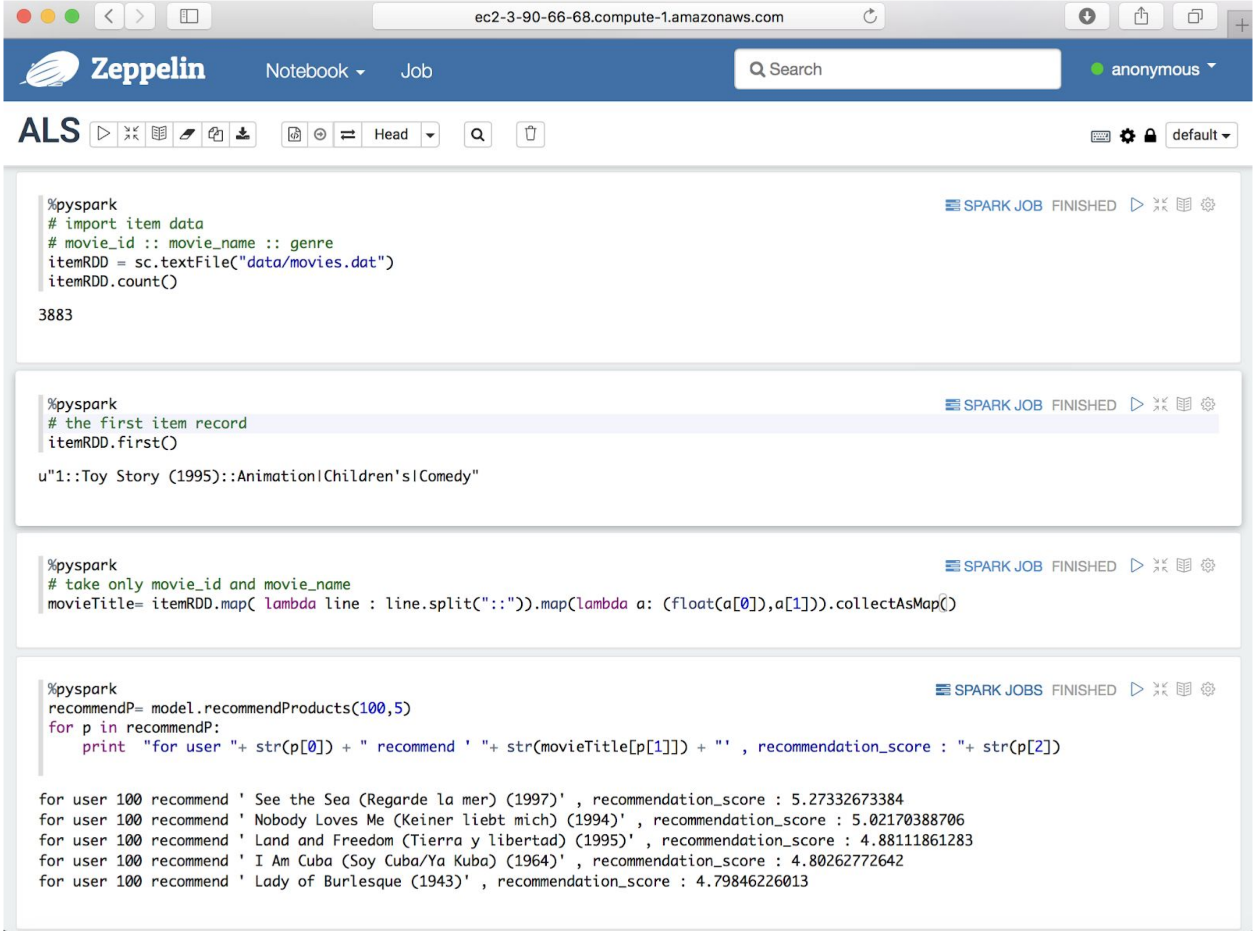
3.4 model and part of the code



```
%pyspark
# ALS model
# the hyper parameter could be tweaked here
model = ALS.train(ratingsRDD, 10, 10, 0.01)
```

```
%pyspark
# recommend top 5 most related products to user_id 100
model.recommendProducts(100,5)

[Rating(user=100, product=2192, rating=5.27332673384497), Rating(user=100, product=106, rating=5.021703887059571), Rating(user=100, product=632, rating=4.881118612831959), Rating(user=100, pr
oduct=3245, rating=4.802627726423592), Rating(user=100, product=958, rating=4.798462260132511)]
```

```
%pyspark
# recommend product_id 200 to top 5 users who might be interested
model.recommendUsers(product=200,num=5)

[Rating(user=5760, product=200, rating=9.10564602910767), Rating(user=89, product=200, rating=8.048368649349037), Rating(user=1459, product=200, rating=7.564286080564312), Rating(user=3897, p
roduct=200, rating=7.478246775292424), Rating(user=2697, product=200, rating=7.3095822341429875)]
```



```
%pyspark
# import item data
# movie_id :: movie_name :: genre
itemRDD = sc.textFile("data/movies.dat")
itemRDD.count()

3883
```

```
%pyspark
# the first item record
itemRDD.first()

u"1::Toy Story (1995)::Animation|Children's|Comedy"
```

```
%pyspark
# take only movie_id and movie_name
movieTitle= itemRDD.map( lambda line : line.split("::")).map(lambda a: (float(a[0]),a[1])).collectAsMap()
```

```
%pyspark
recommendP= model.recommendProducts(100,5)
for p in recommendP:
    print  "for user "+ str(p[0]) + " recommend ' "+ str(movieTitle[p[1]]) + "' , recommendation_score : "+ str(p[2])

for user 100 recommend ' See the Sea (Regarde la mer) (1997)' , recommendation_score : 5.27332673384
for user 100 recommend ' Nobody Loves Me (Keiner liebt mich) (1994)' , recommendation_score : 5.02170388706
for user 100 recommend ' Land and Freedom (Tierra y libertad) (1995)' , recommendation_score : 4.88111861283
for user 100 recommend ' I Am Cuba (Soy Cuba/Ya Kuba) (1964)' , recommendation_score : 4.80262772642
for user 100 recommend ' Lady of Burlesque (1943)' , recommendation_score : 4.79846226013
```