Team 1

A Recommender System for Movies

0. Team members:

Changrong Chen

Haobo Zhao

1. Overview

1) Goals

1) Practicing Scala and Apache Spark, and enhance general programming skills, including Git, etc.
2) Building a decent recommender system specifically for movies
3) Building teamwork spirit through collaborations
4) Achieving the accredited criteria of course
5) Practicing configurations on AWS
6) Meeting deadlines of each sprint

2) Acceptance Criterion

Expected: Root Mean Square Error (RMSE) should be less than 1

Achieved: Best Root Mean Square Error (RMSE) for data model is 0.7813

Our goal is met.

3) User Cases

1) Based on the preferences and searching histories of users, categories, rating, etc., provide potential movies that might be most appealing to the clients.

2) Calibrate the services to the preferences of users through continuous usage of the recommender system by users.

3) Recommend most popular movies to new users.

4) Milestones

1) Week 1 (November 8 2019 - November 14 2019): Setup the development of Scala and Spark for the project and run some small dataset for prototyping

2) Week 2 (November 15 2019 - November 21 2019): Early-state implementation (data cleaning, ingestion, etc.)

3) Week 3(November 22 2019 - November 28 2019): Model building, training and testing

4) Week 4(November 29 2019 - December 6 2019): More testing, deployment and wrapping up the project for presentation

2. Architecture:

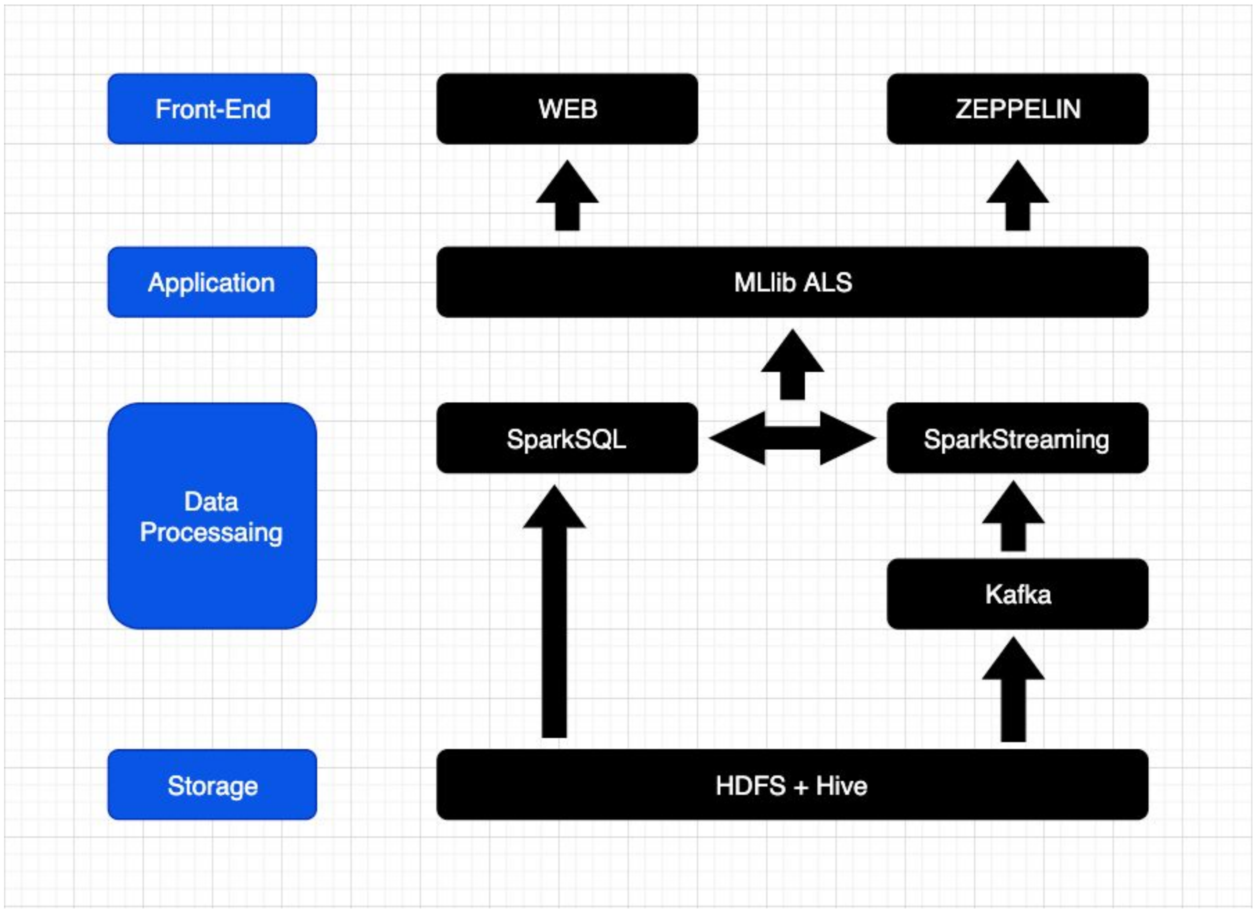
1) HDFS: File System

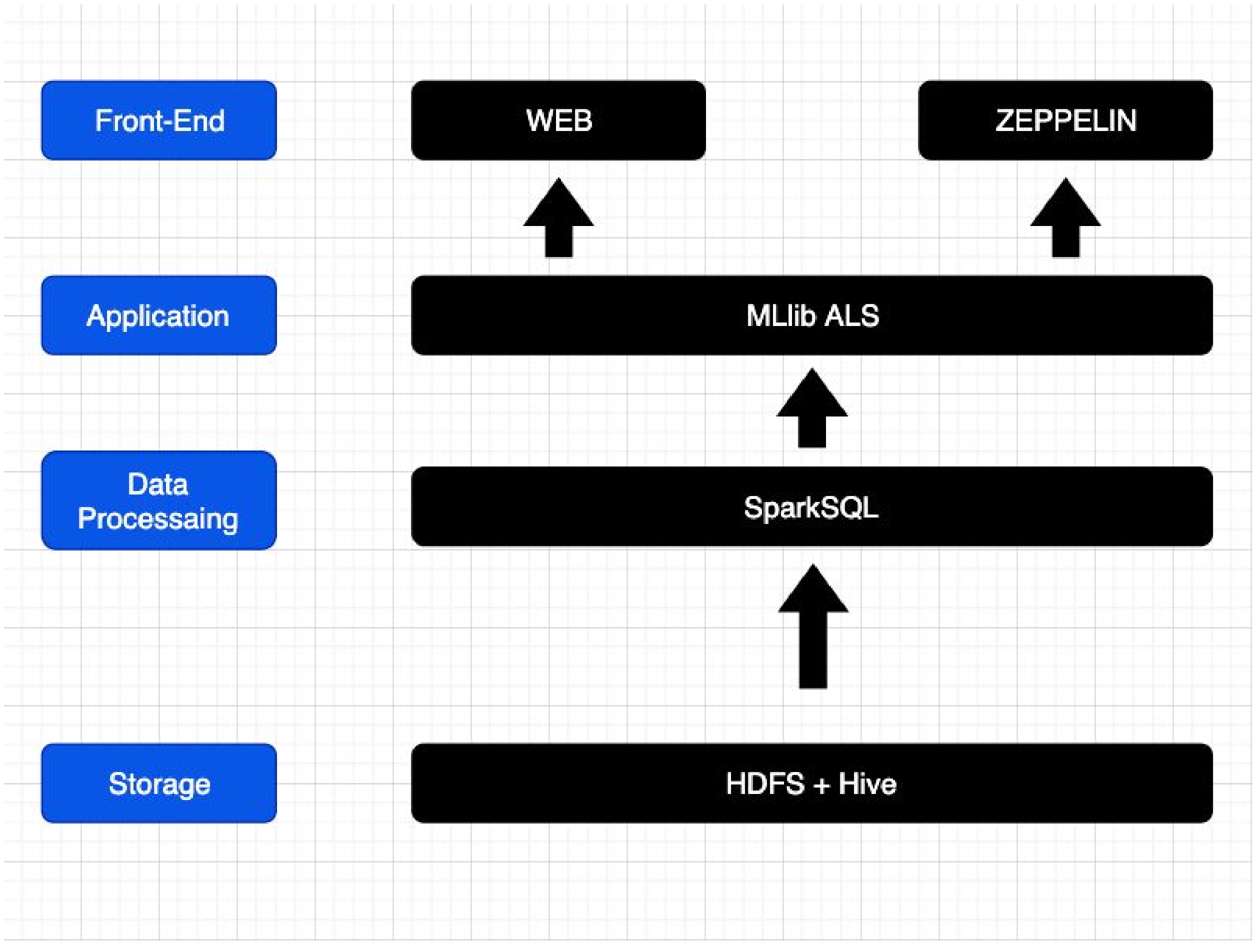
2) Hive: Data Warehouse

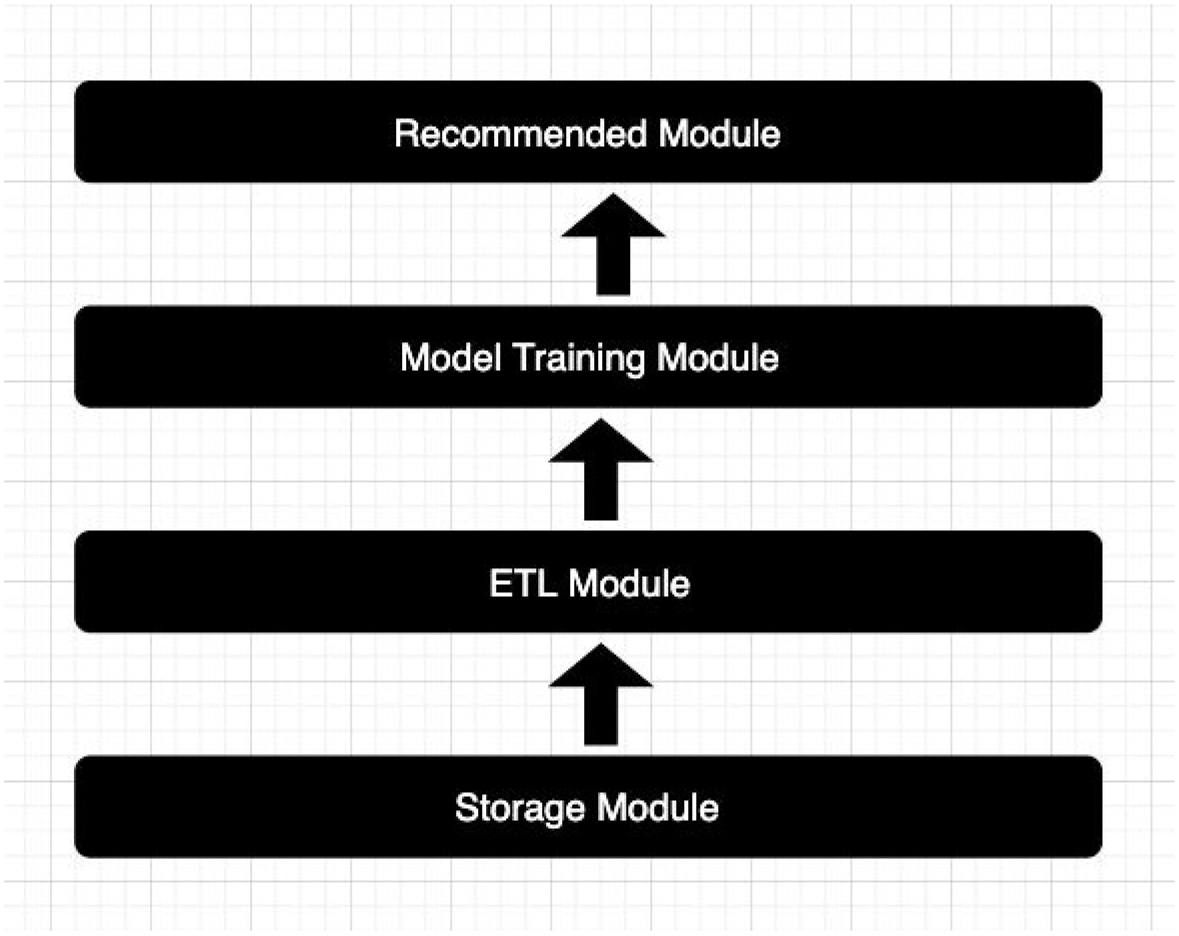
3) Offline DataProcessing: SparkSQL

4) Online DataProcessing: Kafka + SparkStreaming

5) Data Application: MLlib




Front-End
WEB
ZEPPELIN
Application
MLlib ALS
Data Processaing
SparkSQL
Storage
HDFS + Hive



3. Project Setup:

1. Data

   Data source: http://files.grouplens.org/datasets/movielens/

   We used:

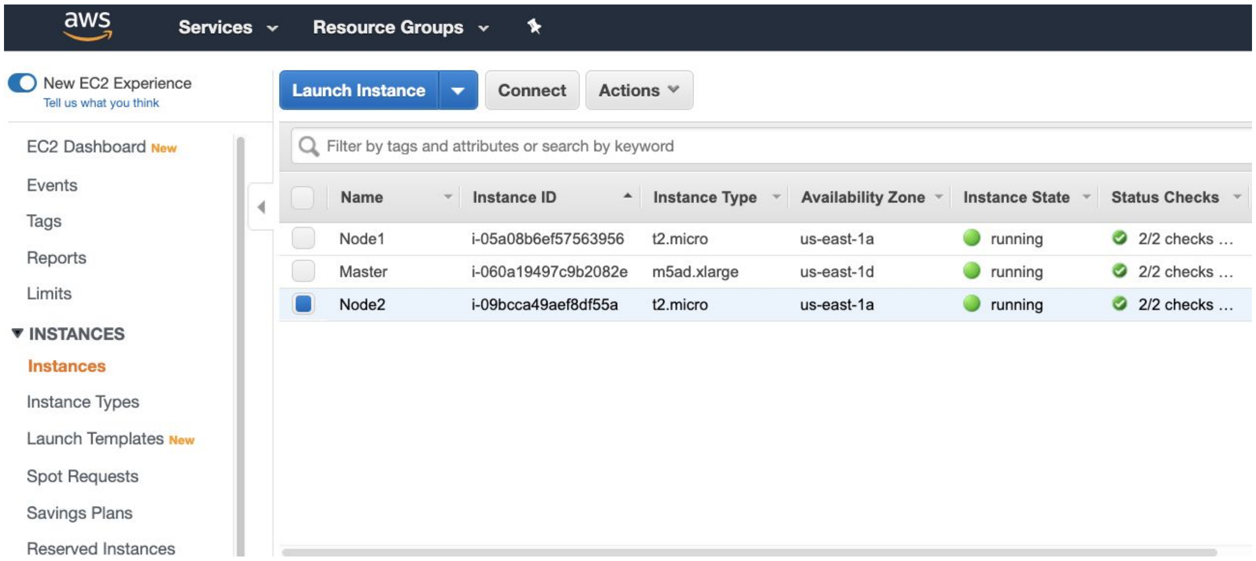
   1) links.csv to link movieid and the website of imdb movie page.

   2) movies.csv to get the movie tags and training model for movies, and calculate the similarity of movies.

3) ratings.csv to do ALS model training, user and item based recommendations.

2. Intellij

3. AWS



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
Node1	i-05a08b6ef57563956	t2.micro	us-east-1a	● running	✓ 2/2 checks ...
Master	i-060a19497c9b2082e	m5ad.xlarge	us-east-1d	● running	✓ 2/2 checks ...
Node2	i-09bcca49aef8df55a	t2.micro	us-east-1a	● running	✓ 2/2 checks ...

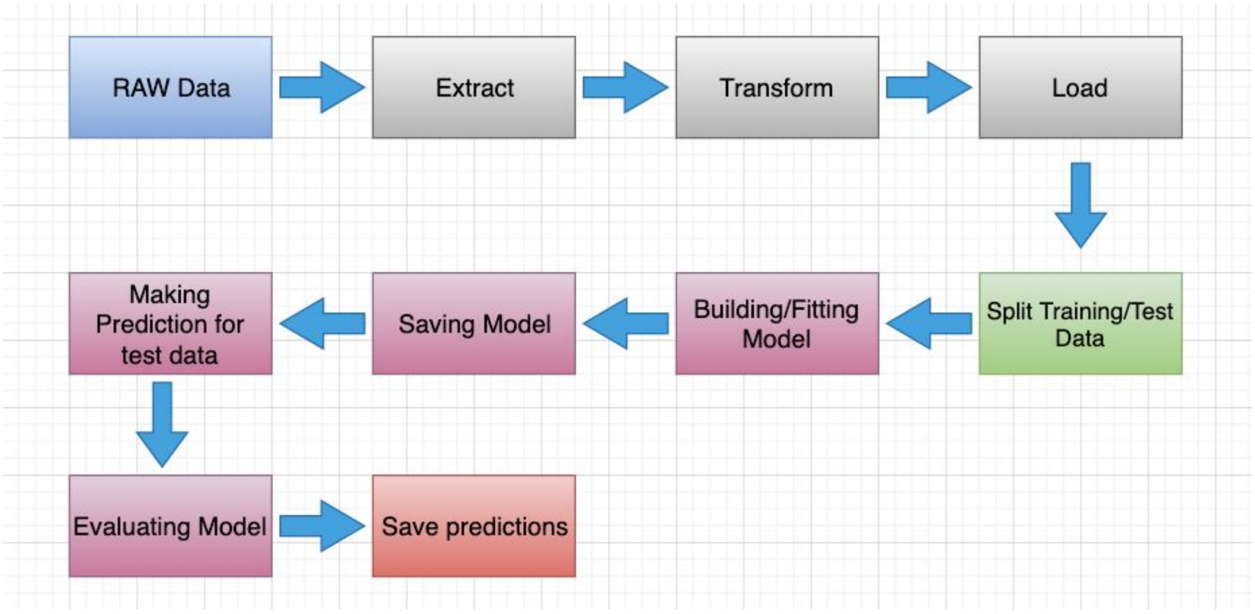
4. Hadoop

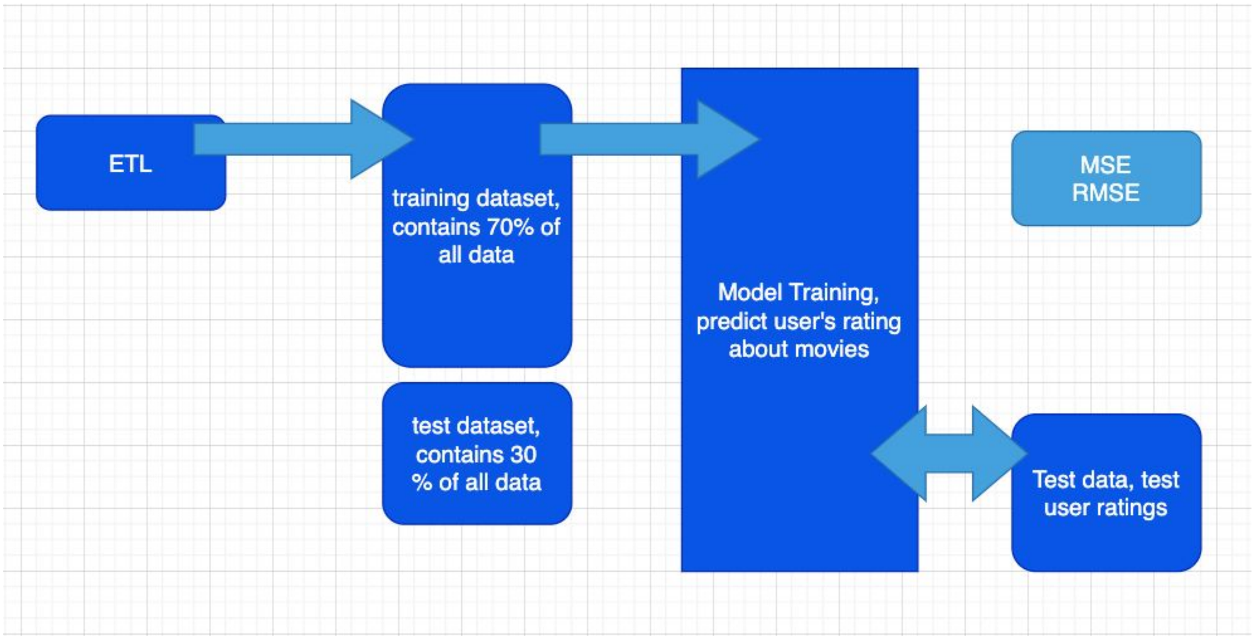
5. MySQL

6. Hive

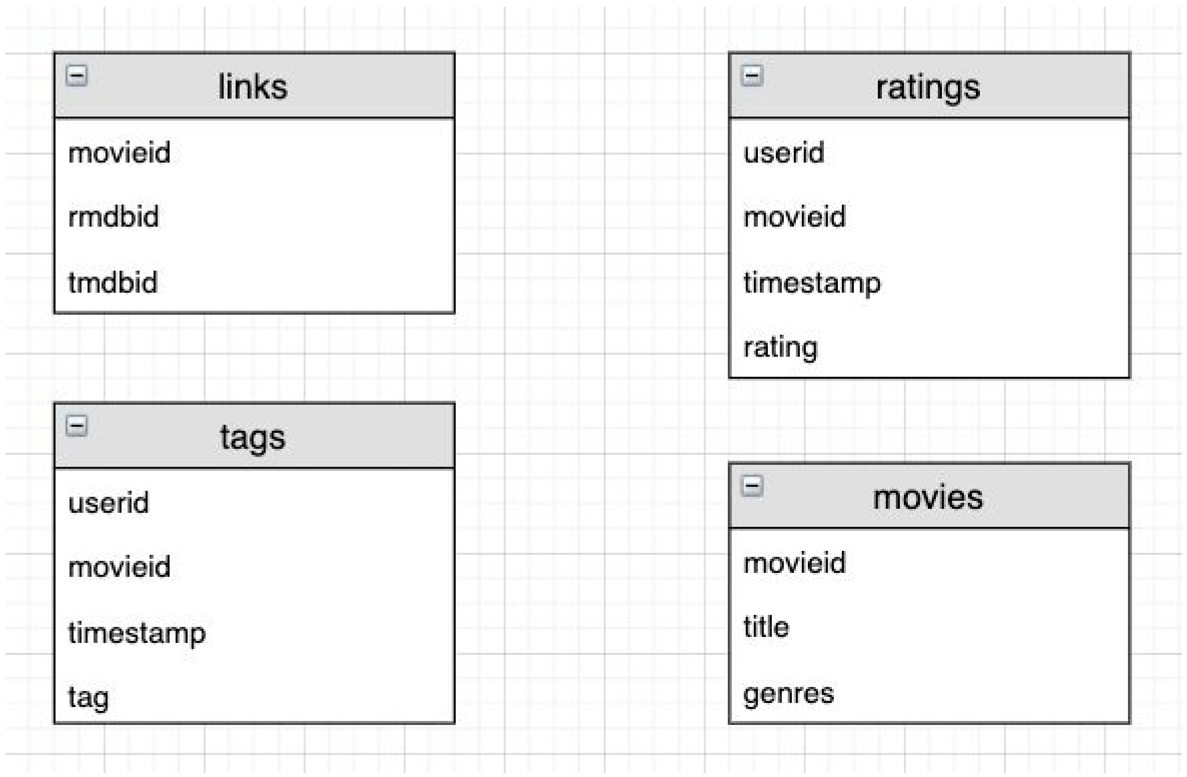
7. Spark

4. Data Modeling




ETL
training dataset, contains 70% of all data
test dataset, contains 30 % of all data
Model Training, predict user's rating about movies
MSE RMSE
Test data, test user ratings

1) Data characteristics



1. Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site (e.g., id `1` corresponds to the URL <https://movielens.org/movies/1>).

2. For the ratings.txt, Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

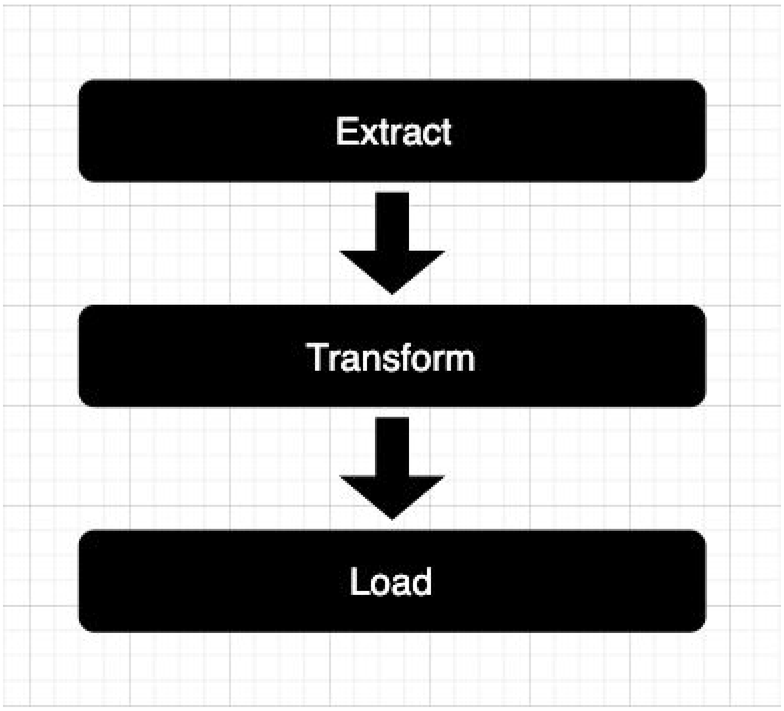
3. Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.

4. Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses.

5. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

2) Data cleaning

ETL Module: Extract, Transform, Load

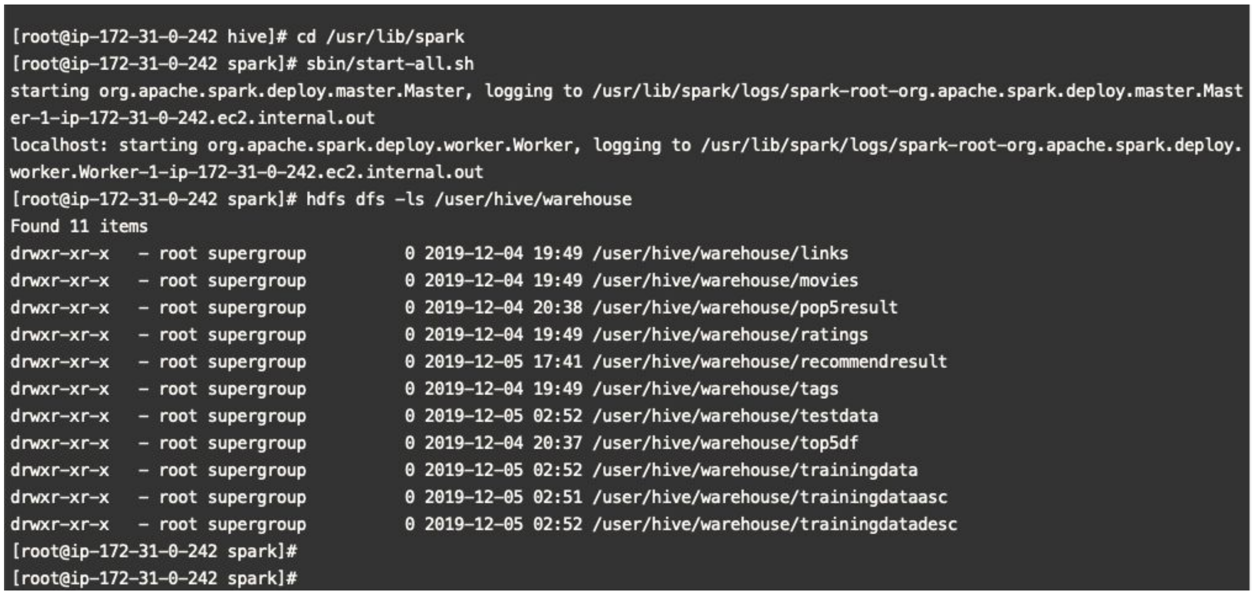


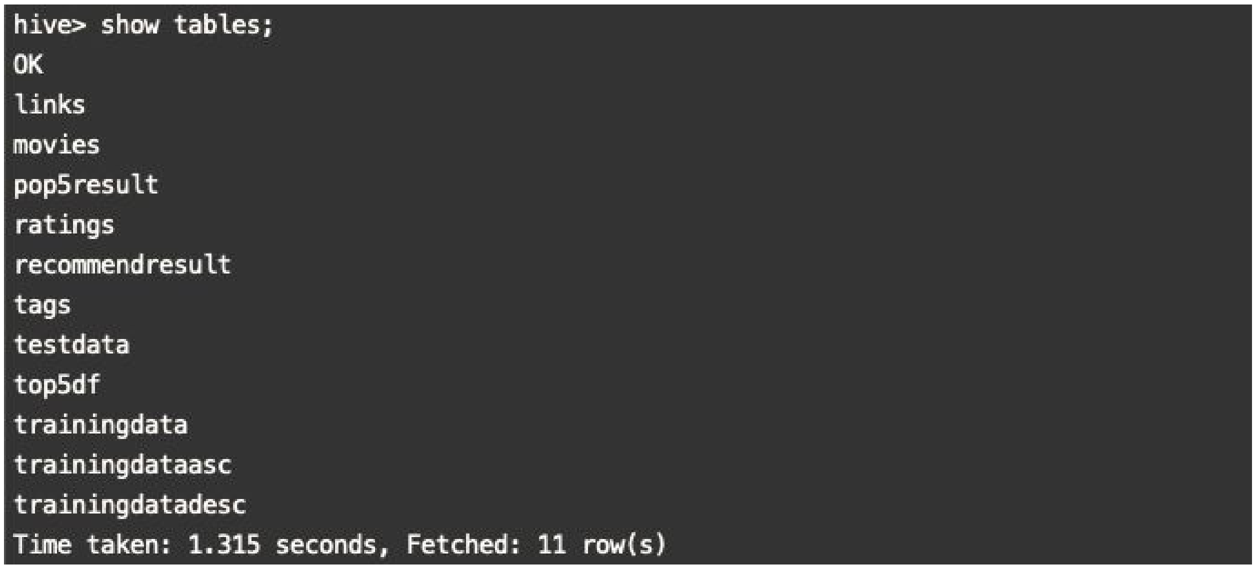
3) Data Modeling

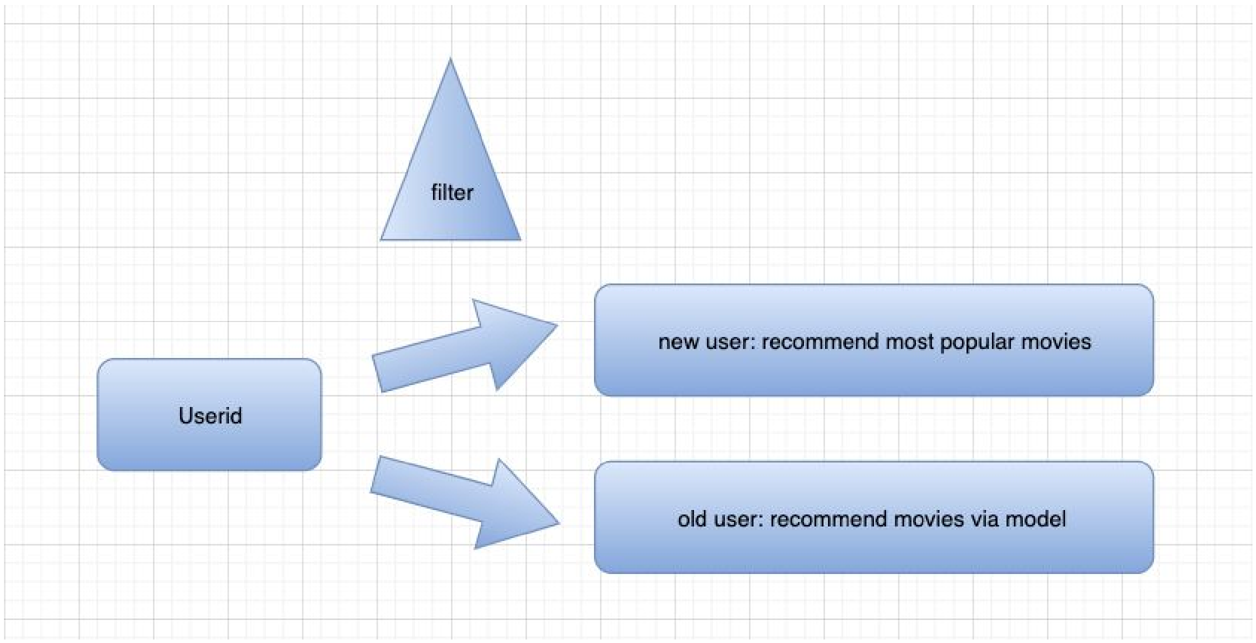
Alternating Least Square (ALS)

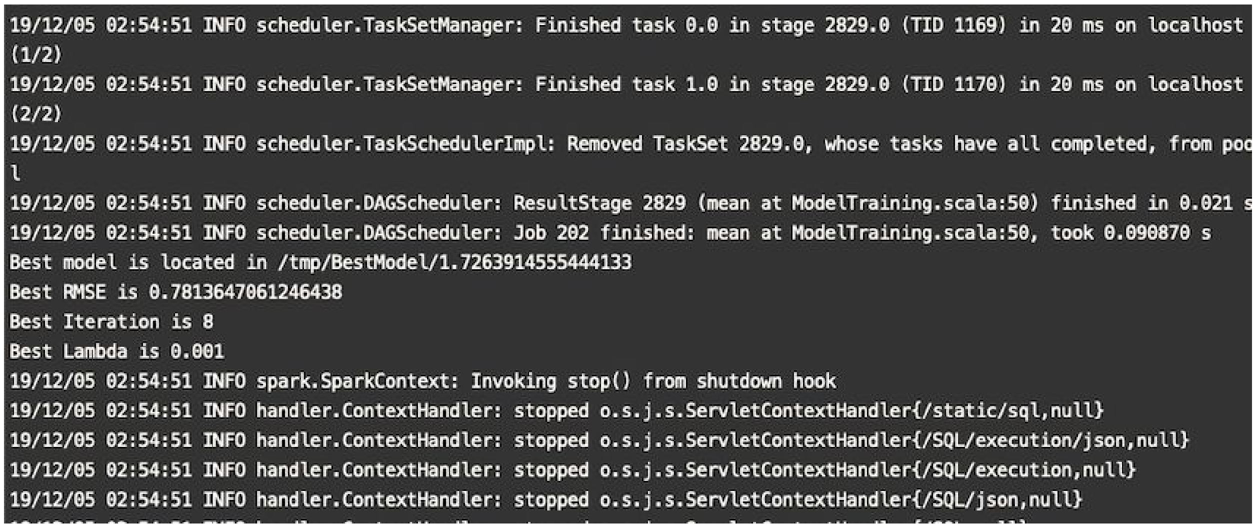
[1] Collaborative Filtering for Implicit Feedback Datasets
https://ieeexplore.ieee.org/abstract/document/4781121

[2] Large-Scale Parallel Collaborative Filtering for the Netflix Prize
https://liuxiaofei.com.cn/blog/wp-content/uploads/2014/01/netflix_aaim08submitted.pdf

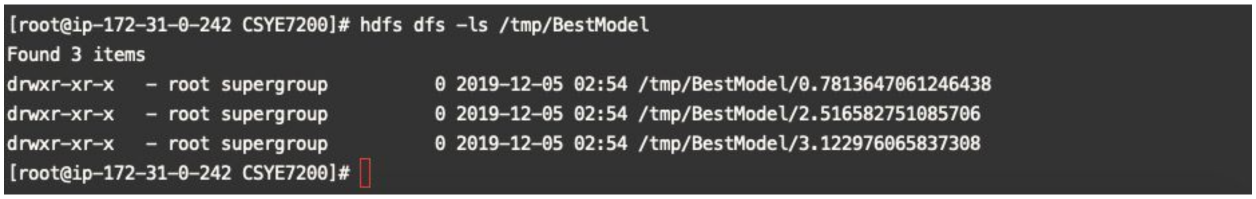

```
[root@ip-172-31-0-242 hive]# cd /usr/lib/spark
[root@ip-172-31-0-242 spark]# sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/lib/spark/logs/spark-root-org.apache.spark.deploy.master.Mast
er-1-ip-172-31-0-242.ec2.internal.out
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /usr/lib/spark/logs/spark-root-org.apache.spark.deploy.
worker.Worker-1-ip-172-31-0-242.ec2.internal.out
[root@ip-172-31-0-242 spark]# hdfs dfs -ls /user/hive/warehouse
Found 11 items
drwxr-xr-x   - root supergroup          0 2019-12-04 19:49 /user/hive/warehouse/links
drwxr-xr-x   - root supergroup          0 2019-12-04 19:49 /user/hive/warehouse/movies
drwxr-xr-x   - root supergroup          0 2019-12-04 20:38 /user/hive/warehouse/pop5result
drwxr-xr-x   - root supergroup          0 2019-12-04 19:49 /user/hive/warehouse/ratings
drwxr-xr-x   - root supergroup          0 2019-12-05 17:41 /user/hive/warehouse/recommendresult
drwxr-xr-x   - root supergroup          0 2019-12-04 19:49 /user/hive/warehouse/tags
drwxr-xr-x   - root supergroup          0 2019-12-05 02:52 /user/hive/warehouse/testdata
drwxr-xr-x   - root supergroup          0 2019-12-04 20:37 /user/hive/warehouse/top5df
drwxr-xr-x   - root supergroup          0 2019-12-05 02:52 /user/hive/warehouse/trainingdata
drwxr-xr-x   - root supergroup          0 2019-12-05 02:51 /user/hive/warehouse/trainingdataasc
drwxr-xr-x   - root supergroup          0 2019-12-05 02:52 /user/hive/warehouse/trainingdatadesc
[root@ip-172-31-0-242 spark]#
[root@ip-172-31-0-242 spark]#
```


```
hive> show tables;
OK
links
movies
pop5result
ratings
recommendresult
tags
testdata
top5df
trainingdata
trainingdataasc
trainingdatadesc
Time taken: 1.315 seconds, Fetched: 11 row(s)
```

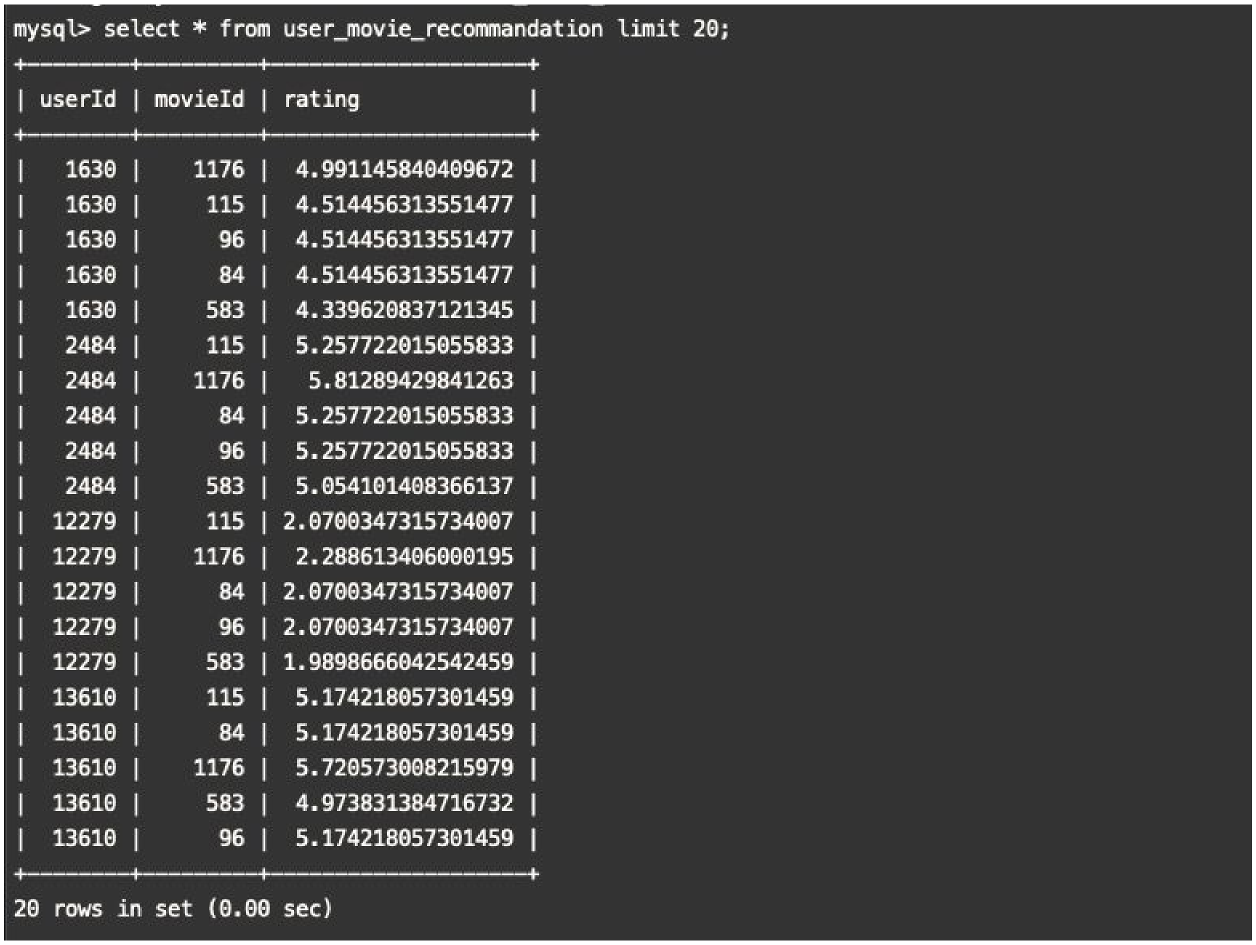

filter
Userid
new user: recommend most popular movies
old user: recommend movies via model



```
19/12/05 02:54:51 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 2829.0 (TID 1169) in 20 ms on localhost
(1/2)
19/12/05 02:54:51 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 2829.0 (TID 1170) in 20 ms on localhost
(2/2)
19/12/05 02:54:51 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 2829.0, whose tasks have all completed, from poo
l
19/12/05 02:54:51 INFO scheduler.DAGScheduler: ResultStage 2829 (mean at ModelTraining.scala:50) finished in 0.021 s
19/12/05 02:54:51 INFO scheduler.DAGScheduler: Job 202 finished: mean at ModelTraining.scala:50, took 0.090870 s
Best model is located in /tmp/BestModel/1.7263914555444133
Best RMSE is 0.7813647061246438
Best Iteration is 8
Best Lambda is 0.001
19/12/05 02:54:51 INFO spark.SparkContext: Invoking stop() from shutdown hook
19/12/05 02:54:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/static/sql,null}
19/12/05 02:54:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/SQL/execution/json,null}
19/12/05 02:54:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/SQL/execution,null}
19/12/05 02:54:51 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/SQL/json,null}
```



```
[root@ip-172-31-0-242 CSYE7200]# hdfs dfs -ls /tmp/BestModel
Found 3 items
drwxr-xr-x   - root supergroup          0 2019-12-05 02:54 /tmp/BestModel/0.7813647061246438
drwxr-xr-x   - root supergroup          0 2019-12-05 02:54 /tmp/BestModel/2.516582751085706
drwxr-xr-x   - root supergroup          0 2019-12-05 02:54 /tmp/BestModel/3.122976065837308
[root@ip-172-31-0-242 CSYE7200]#
```



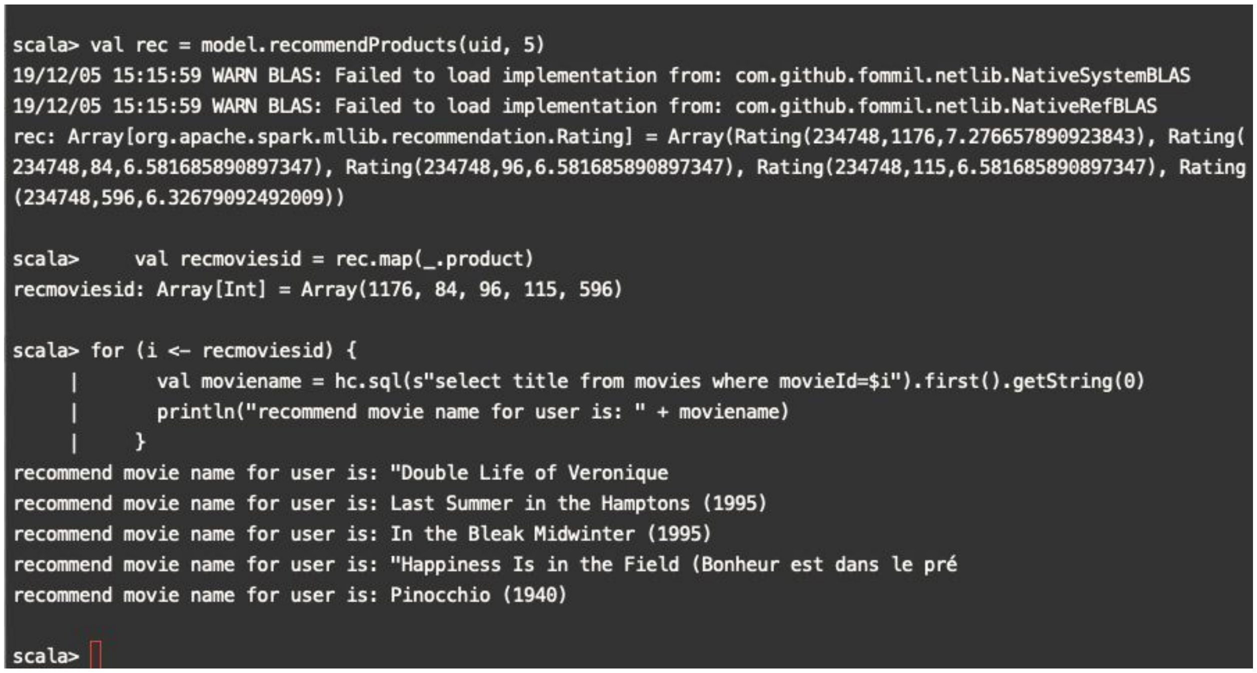
```
mysql> select * from user_movie_recommandation limit 20;
+--------+---------+--------------------+
| userId | movieId | rating             |
+--------+---------+--------------------+
|   1630 |    1176 |  4.991145840409672 |
|   1630 |     115 |  4.514456313551477 |
|   1630 |      96 |  4.514456313551477 |
|   1630 |      84 |  4.514456313551477 |
|   1630 |     583 |  4.339620837121345 |
|   2484 |     115 |  5.257722015055833 |
|   2484 |    1176 |   5.81289429841263 |
|   2484 |      84 |  5.257722015055833 |
|   2484 |      96 |  5.257722015055833 |
|   2484 |     583 |  5.054101408366137 |
|  12279 |     115 | 2.0700347315734007 |
|  12279 |    1176 |  2.288613406000195 |
|  12279 |      84 | 2.0700347315734007 |
|  12279 |      96 | 2.0700347315734007 |
|  12279 |     583 | 1.9898666042542459 |
|  13610 |     115 |  5.174218057301459 |
|  13610 |      84 |  5.174218057301459 |
|  13610 |    1176 |  5.720573008215979 |
|  13610 |     583 |  4.973831384716732 |
|  13610 |      96 |  5.174218057301459 |
+--------+---------+--------------------+
20 rows in set (0.00 sec)
```

5. Testing

1) Spark: Word Count

2) Hadoop

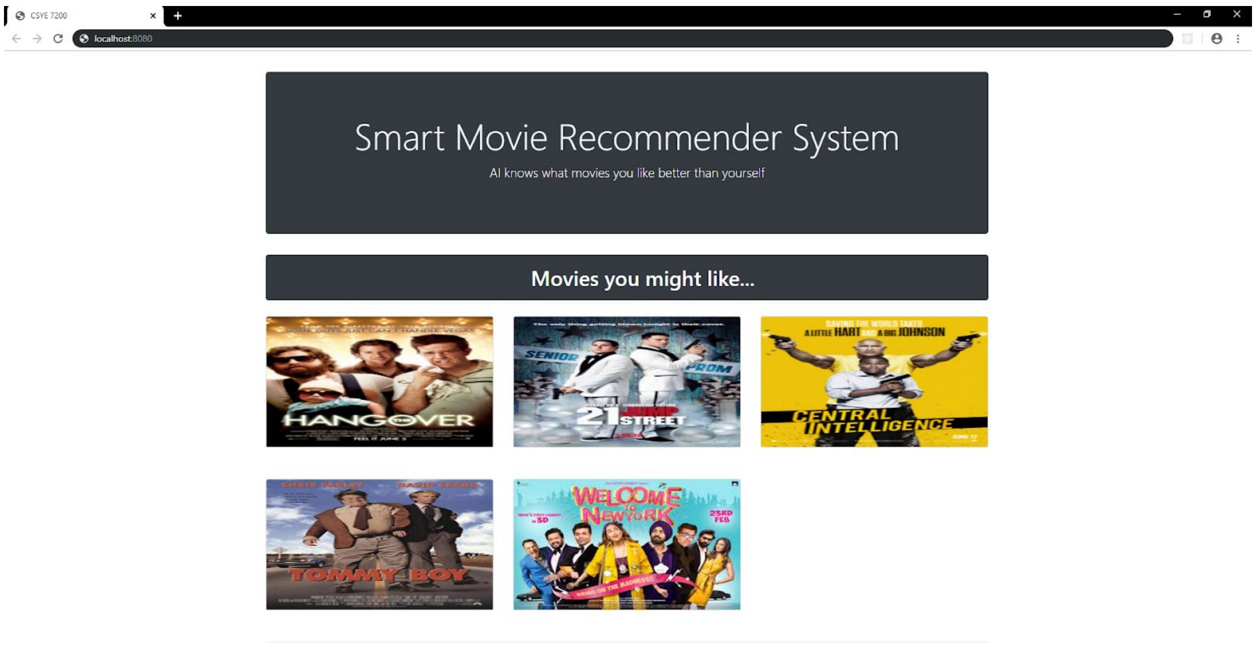
3) Single user testing



```
scala> val rec = model.recommendProducts(uid, 5)
19/12/05 15:15:59 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
19/12/05 15:15:59 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
rec: Array[org.apache.spark.mllib.recommendation.Rating] = Array(Rating(234748,1176,7.276657890923843), Rating(
234748,84,6.581685890897347), Rating(234748,96,6.581685890897347), Rating(234748,115,6.581685890897347), Rating
(234748,596,6.32679092492009))

scala>     val recmoviesid = rec.map(_.product)
recmoviesid: Array[Int] = Array(1176, 84, 96, 115, 596)

scala> for (i <- recmoviesid) {
     |       val moviename = hc.sql(s"select title from movies where movieId=$i").first().getString(0)
     |       println("recommend movie name for user is: " + moviename)
     |     }
recommend movie name for user is: "Double Life of Veronique
recommend movie name for user is: Last Summer in the Hamptons (1995)
recommend movie name for user is: In the Bleak Midwinter (1995)
recommend movie name for user is: "Happiness Is in the Field (Bonheur est dans le pré
recommend movie name for user is: Pinocchio (1940)

scala> 
```

6. User Interface

1) Screenshot for a current user

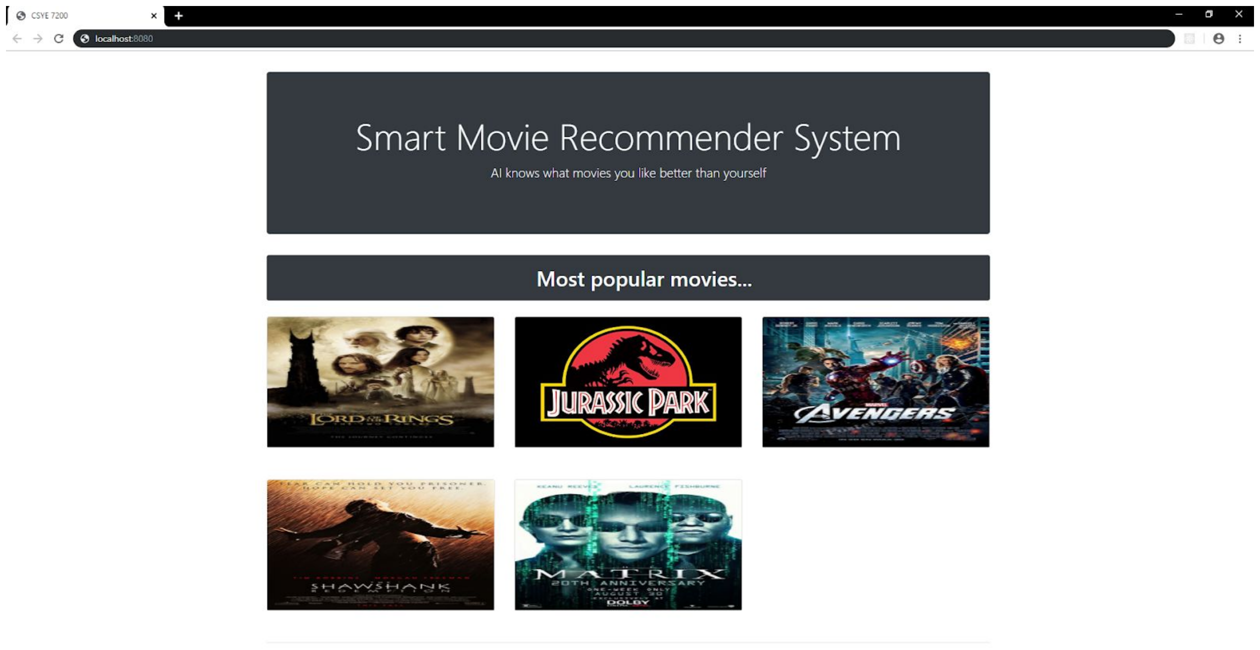


Smart Movie Recommender System

AI knows what movies you like better than yourself

Movies you might like...

2) Screenshot for a new user



Smart Movie Recommender System

AI knows what movies you like better than yourself

Most popular movies...

7. Summary

1) Improved knowledge and skills on Scala, Apache Spark and general programming skills, including Git, etc.

2) Improved knowledge and skills on AWS

3) Delivery of a workable solution

4) Improved presentation skills