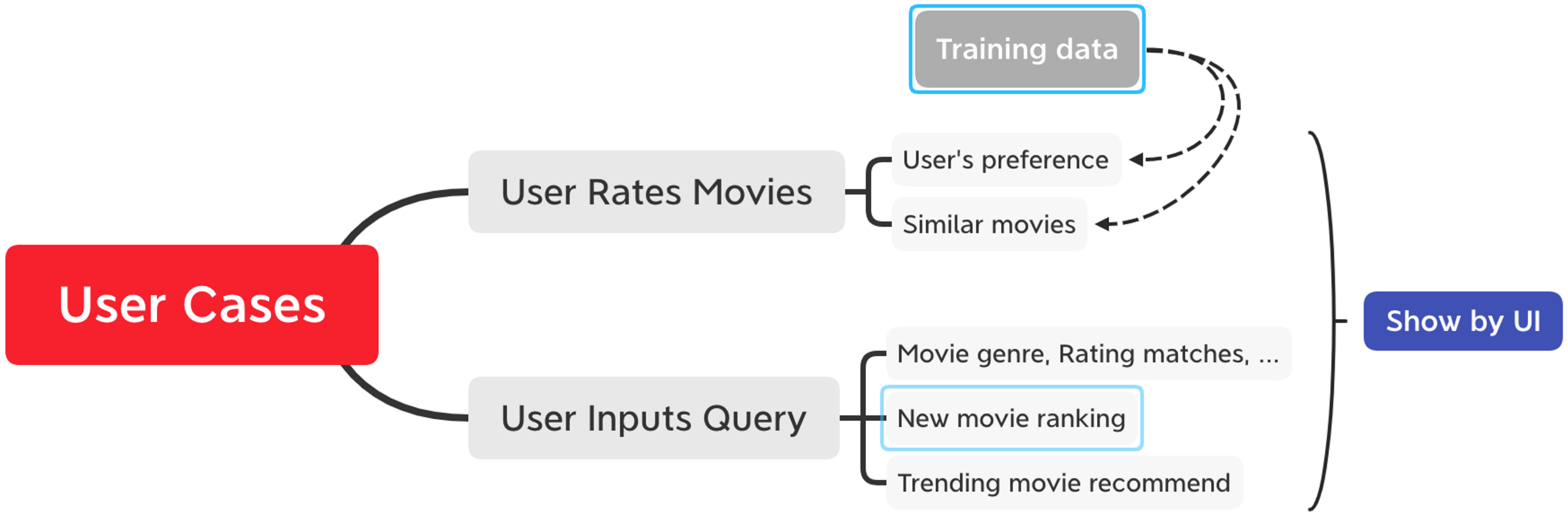
Movie Recommendation

Team III

Mengzhe ZHANG

Rongqi SUN

Yue FANG

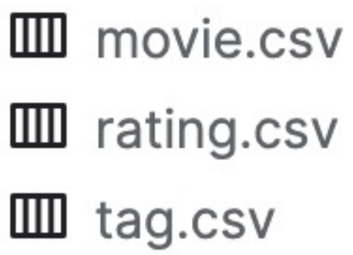

User Cases
User Rates Movies
User's preference
Similar movies
Training data
User Inputs Query
Movie genre, Rating matches, ...
New movie ranking
Trending movie recommend
Show by UI

Methodology

- **Statistics:**
  - Use Spark Core + Spark SQL deal dataset
    - Load dataset with MongoDB through Spark SQL
- **Recommendation:**
  - Algorithms use ALS to summarize user's preference and similar movies
  - Use Spark Core + Spark MLlib to implement recommendation methods

Data Source

- **MovieLens 20M Dataset**
- https://www.kaggle.com/grouplens/movielens-20m-dataset
- **It contains 20000263 ratings and 465564 tag applications across 27278 movies.**
- **These data were created by 138493 users between January 09, 1995 and March 31, 2015.**
- **Users were selected at random for inclusion. All selected users had rated at least 20 movies.**


movie.csv
rating.csv
tag.csv

Milestones

Milestones	Time
Data cleaning and processing Unit Test	3.22 – 3.28
Recommendation methods implementing Unit Test	3.29 – 4.7
Setup UI Implement visualization	4.8 – 4.14
Final model and use cases testing System Test	4.15 – 4.21

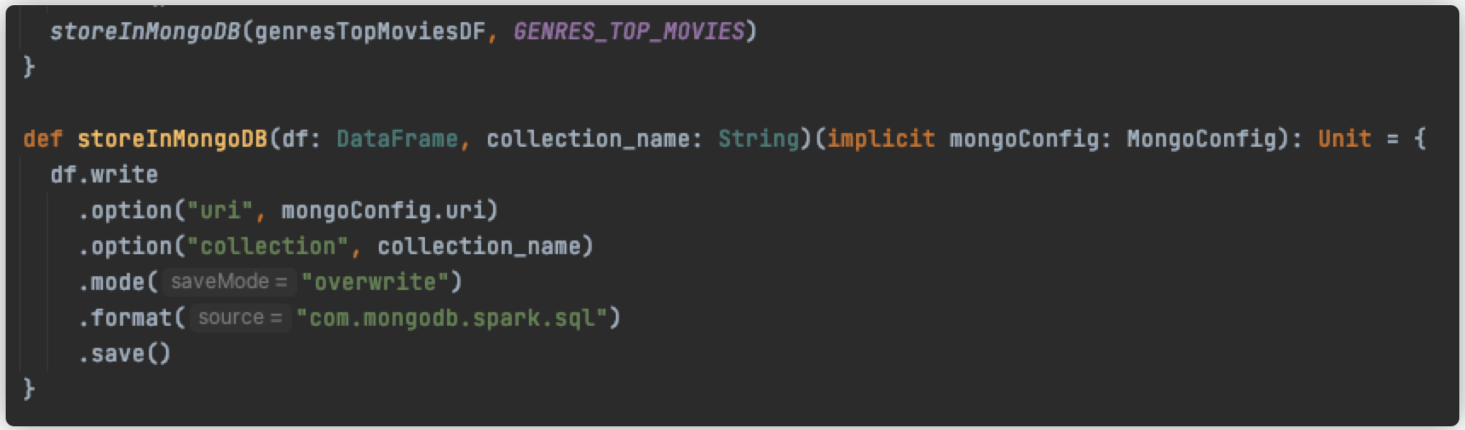
Repository

- Scala:
- Recommendation Part
- User Interface Use Play
- MongDB for dataset using SparkSQL
- Repo: https://github.com/tracy626/CSYE7200_FinalProj_Team3

Implementation

Statistics

- Most rated movies
- Movies of highest average score
- Movies of highest score in genre



```
    storeInMongoDB(genresTopMoviesDF, GENRES_TOP_MOVIES)
  }

  def storeInMongoDB(df: DataFrame, collection_name: String)(implicit mongoConfig: MongoConfig): Unit = {
    df.write
      .option("uri", mongoConfig.uri)
      .option("collection", collection_name)
      .mode( saveMode = "overwrite")
      .format( source = "com.mongodb.spark.sql")
      .save()
  }
```

Implementation

ALS

- The Cartesian product of UserID and MovieID produces a empty tuple of (uid, mid)
- The tuple of (uid, mid) predicted by the model.
- Sort the prediction results by prediction score.
- Return the K movies with the highest scores as the recommendation of the current user.

- Parameters (rank, iterations, lambda)

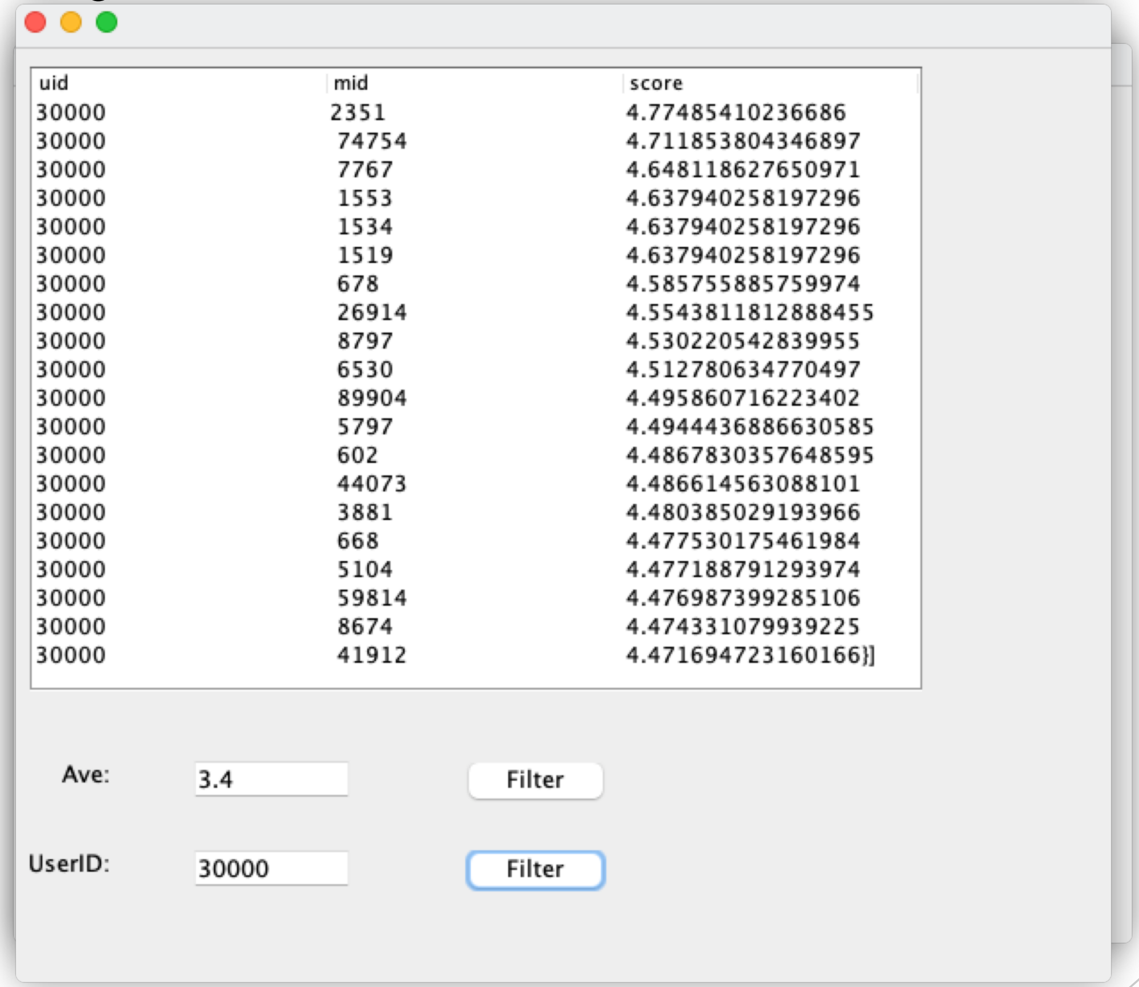
We need to evaluate the model. The usual approach is to calculate the root mean square error (RMSE) and examine the error between the predicted score and the actual score.

- With RMSE, we can select the group with the smallest RMSE as the optimal choice for our model by adjusting the parameter values multiple times.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(observed_t - predicted_t)^2}$$

GUI

- Play
- Swing



uid	mid	score
30000	2351	4.77485410236686
30000	74754	4.711853804346897
30000	7767	4.648118627650971
30000	1553	4.637940258197296
30000	1534	4.637940258197296
30000	1519	4.637940258197296
30000	678	4.585755885759974
30000	26914	4.5543811812888455
30000	8797	4.530220542839955
30000	6530	4.512780634770497
30000	89904	4.495860716223402
30000	5797	4.4944436886630585
30000	602	4.4867830357648595
30000	44073	4.486614563088101
30000	3881	4.480385029193966
30000	668	4.477530175461984
30000	5104	4.477188791293974
30000	59814	4.476987399285106
30000	8674	4.474331079939225
30000	41912	4.471694723160166}]

Ave: 3.4

UserID: 30000

Acceptance criteria

- User is able to:
  - Rate movie and get feedback about similar movies recommendation according to rating history (predict user's preference)
  - Ask for recommendation of new movies and get list by rank
  - Get trending movie recommendation
  - Get User-Based Recommend result in less than 3 seconds
  - Get Statistics Recommend result in less than 1 second
  - …

Goals of the Project

- Clean and process raw dataset
- Analyze movies rating with other features
- Input recommendation and analysis results to database
- Create UI for recommendation system
- Create reactive page for user to filter recommendation results