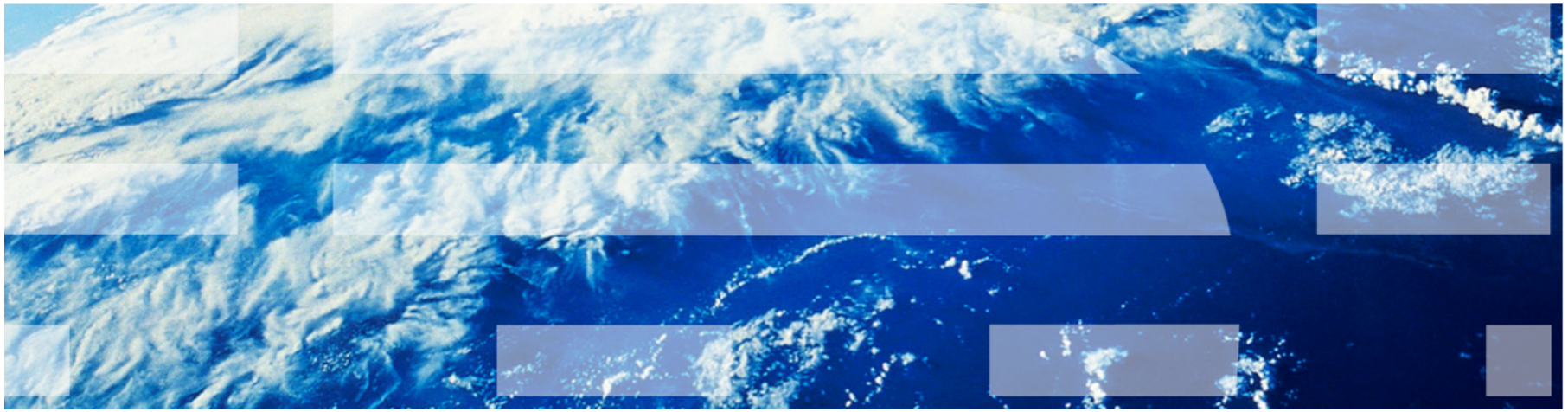
E6893 Big Data Analytics Project:

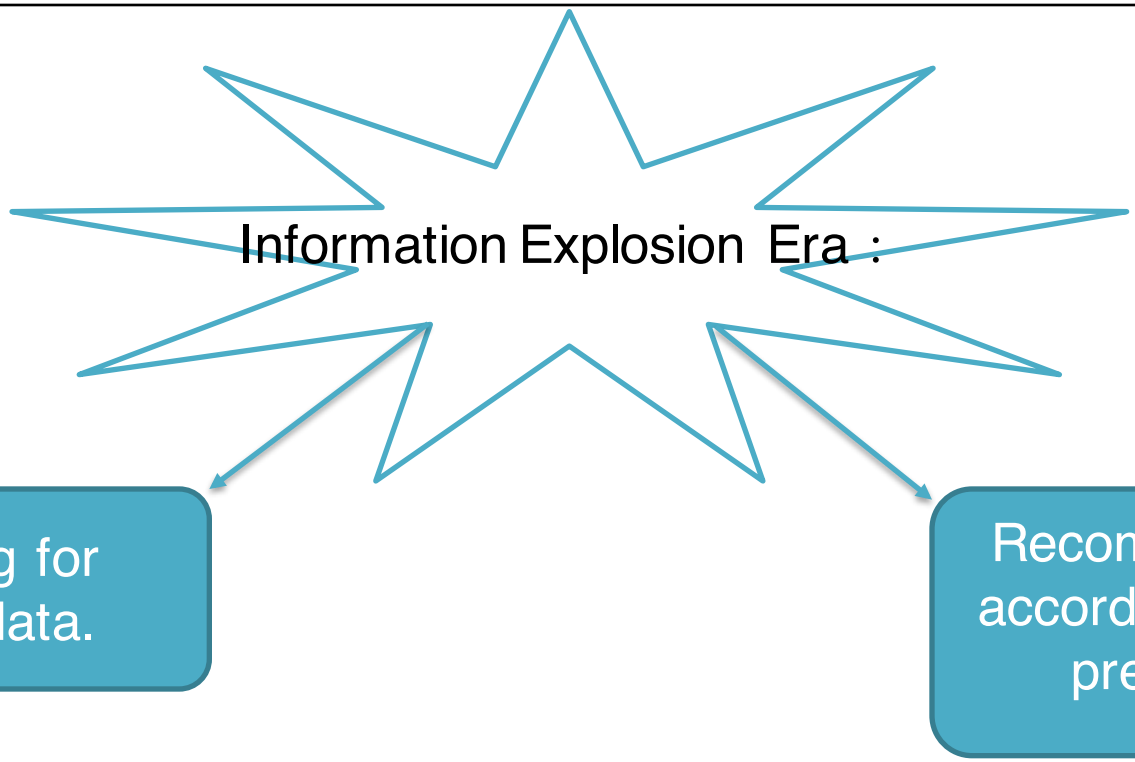
Movie clustering and recommendation based on Netflix movie rating data

Tianchun Yang, Ziyi Luo, Pengyuan Zhao



December 18th, 2015

Motivation



In this era of information explosion:
the most important part of data processing for movie site is: Clustering for existing data.
the most important part of movie site service is: Recommendation

Dataset

Academic Torrents | Browse | Upload | About | Contribute | Login | paper, author, or dataset | Search

Netflix Prize Data Set

Netflix

Home | Technical 33/0 | Comments 0 | Collections | Download 697.55MB

Type: Dataset
Tags:
Abstract:
This is the official data set used in the Netflix Prize competition. The data consists of about 100 million movie ratings, and the goal is to predict missing entries in the movie-user rating matrix.

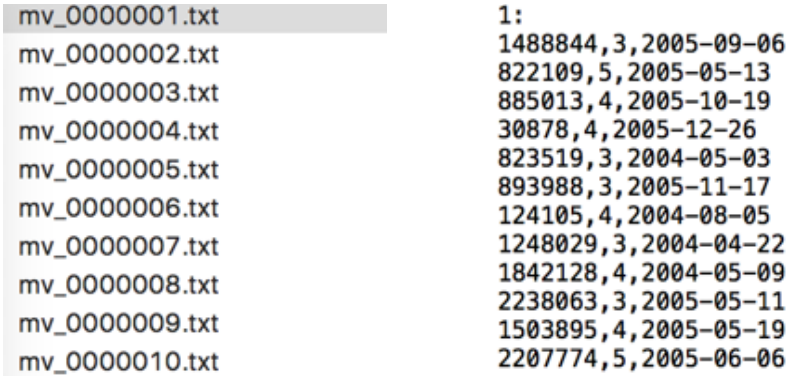
Data form:

MovieD1.txt

```
MovieID1:
CustomerID11,Rate11,Date11
CustomerID12,Rate12,Date12
...
```

MovieD2.txt

```
MovieID2:
CustomerID21,Rate21,Date21
CustomerID22,Rate22,Date22
```

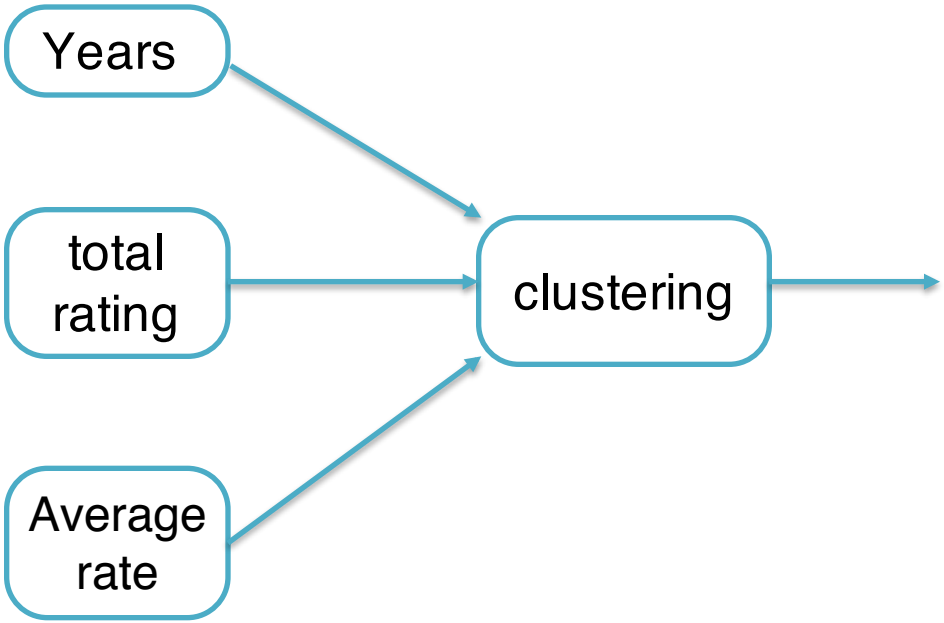


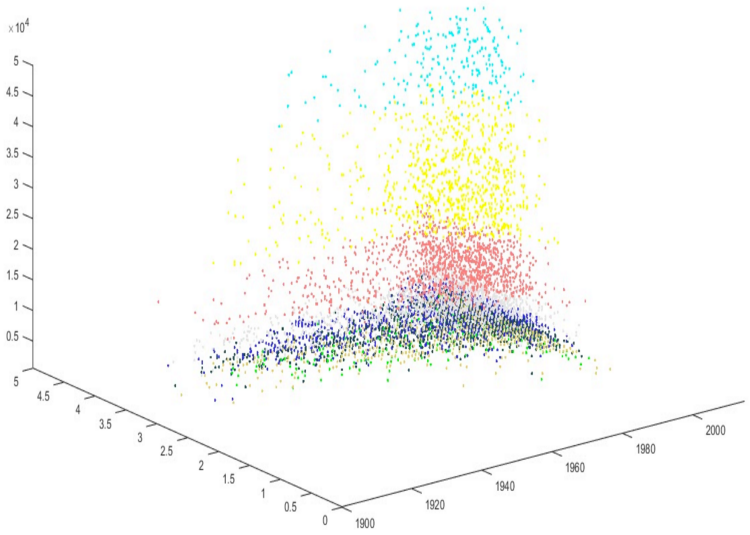
Total Data Size (After extraction): 2.51GB
Total Movies in Dataset: 17770
Total User number: 480189

Algorithms

- **Clustering algorithm:**

  K-Means Clustering, Fuzzy k-Means, Spectral Clustering, Canopy Clustering



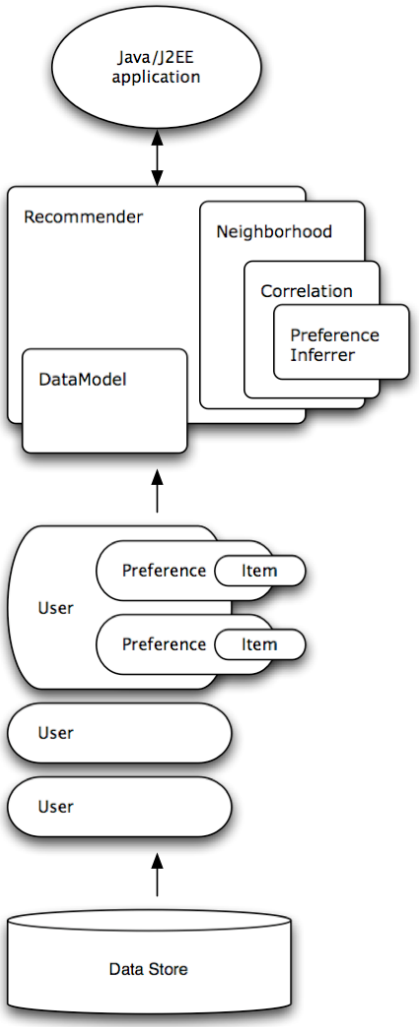


Algorithms

- **Recommendation algorithm**

  User-based Recommender:

  - Pearson Correlation Similarity
  - Nearest User Neighborhood



Algorithm and tools

Tools

- **Build Tools:**

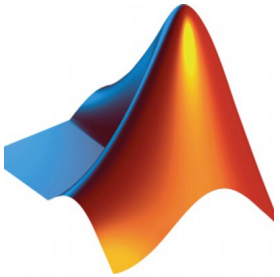
  Hadoop & Mahout & Eclipse

- **Aided analytic Tools:**

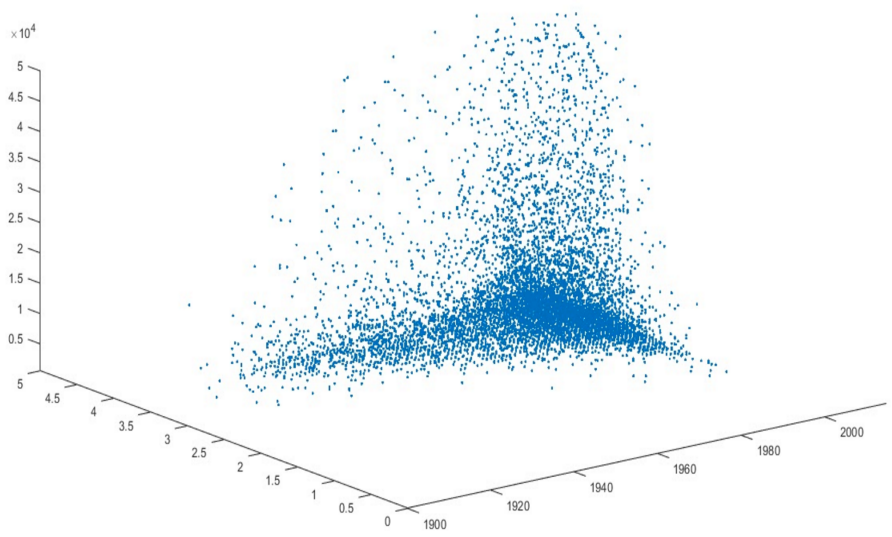
  Data preprocess & plot: Matlab, R

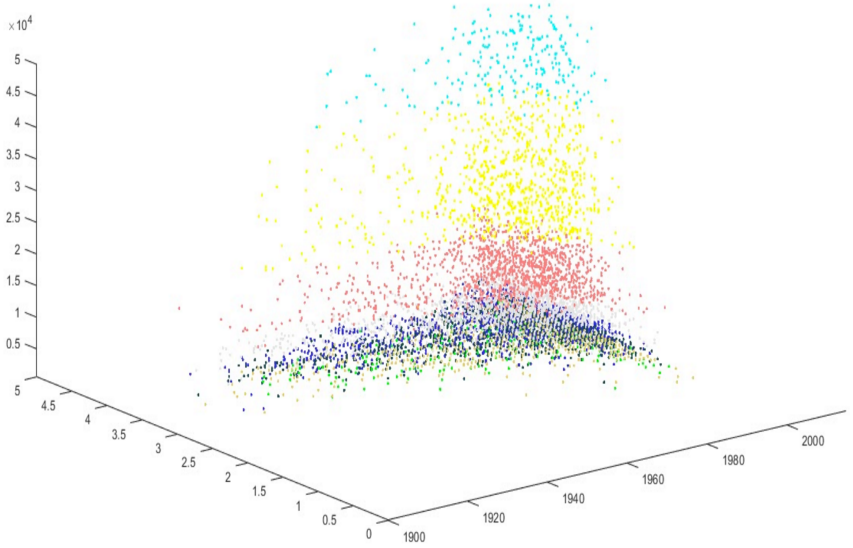
- **API:**

  OMDB Web service



Clustering



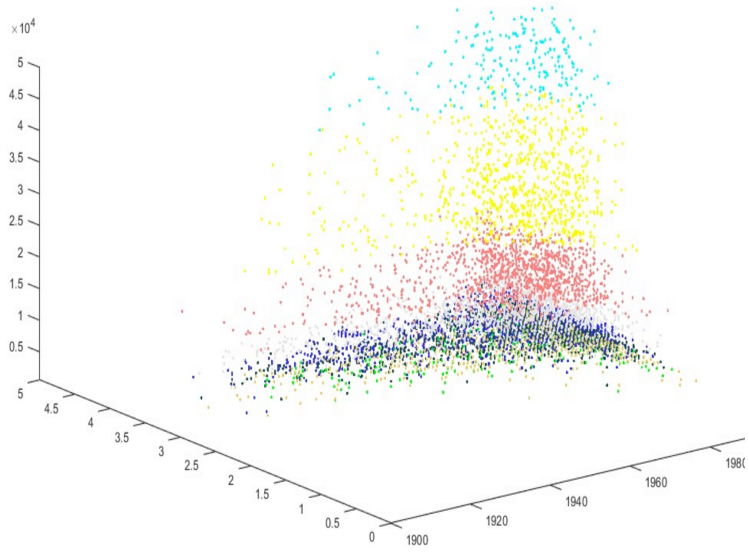


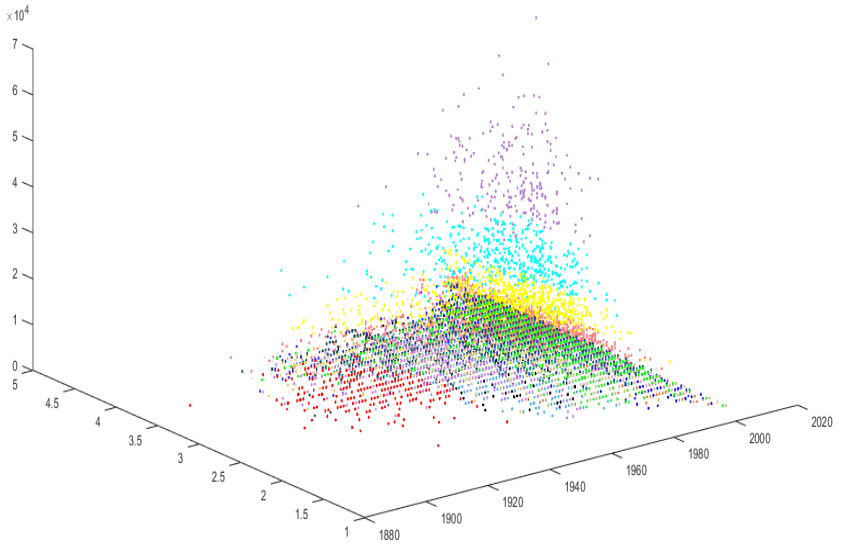
before clustering vs after clustering(K-means)

```
27,1962,3.5,273
28,2002,3.8,39752
29,2001,3.6,523
30,2003,3.8,118413
31,1999,3.1,221
32,2004,4.1,1854
```

- **Highly unbalanced input data**
- **EuclideanDistanceMeasure based**
- **Data clustered mainly based on total rate**

Clustering



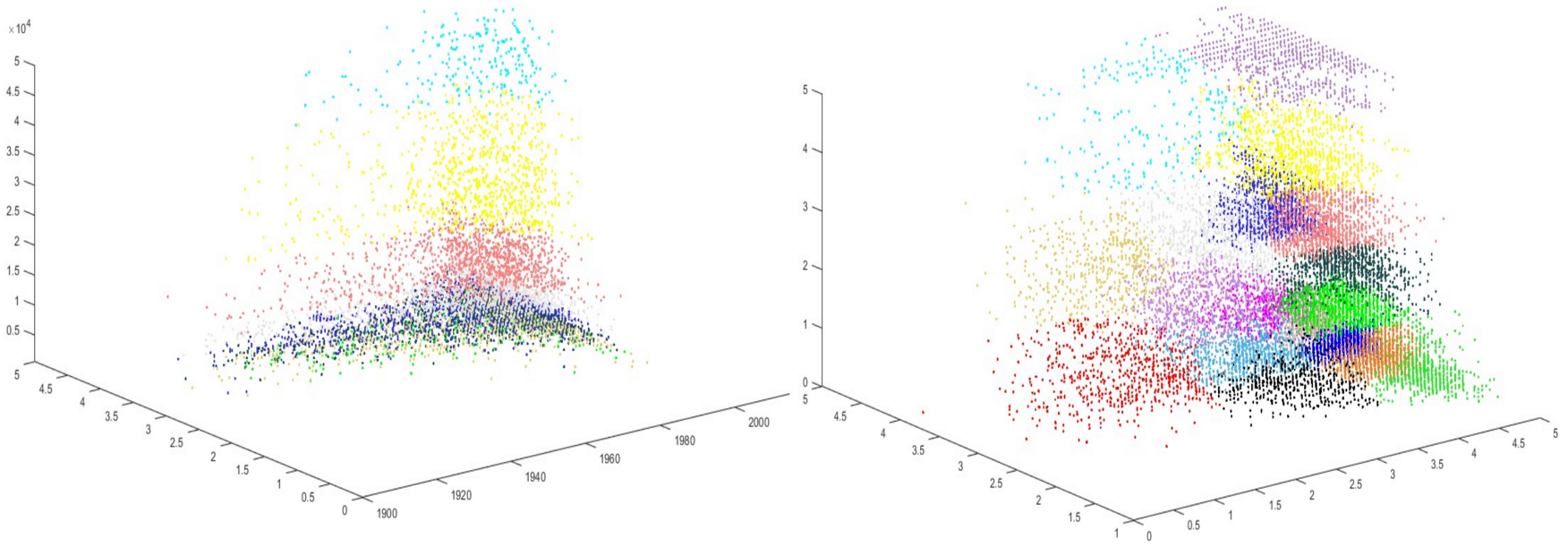


Total rate VS. popularity

- **Transfer total rate to popularity**
- **Total rate/Avg rate**
- **Data clustered mainly based on publish year**

  (But also based on popularity when the magnitude is higher than year)

Clustering

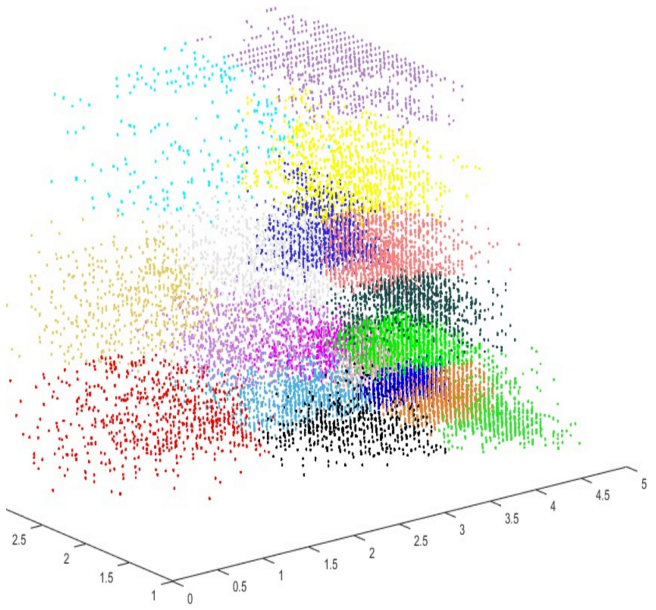


before weighted vs after weighted(K-means)

- **Weighted attributes**
- **All attributes in similar magnitude while maintain it's difference**

Clustering

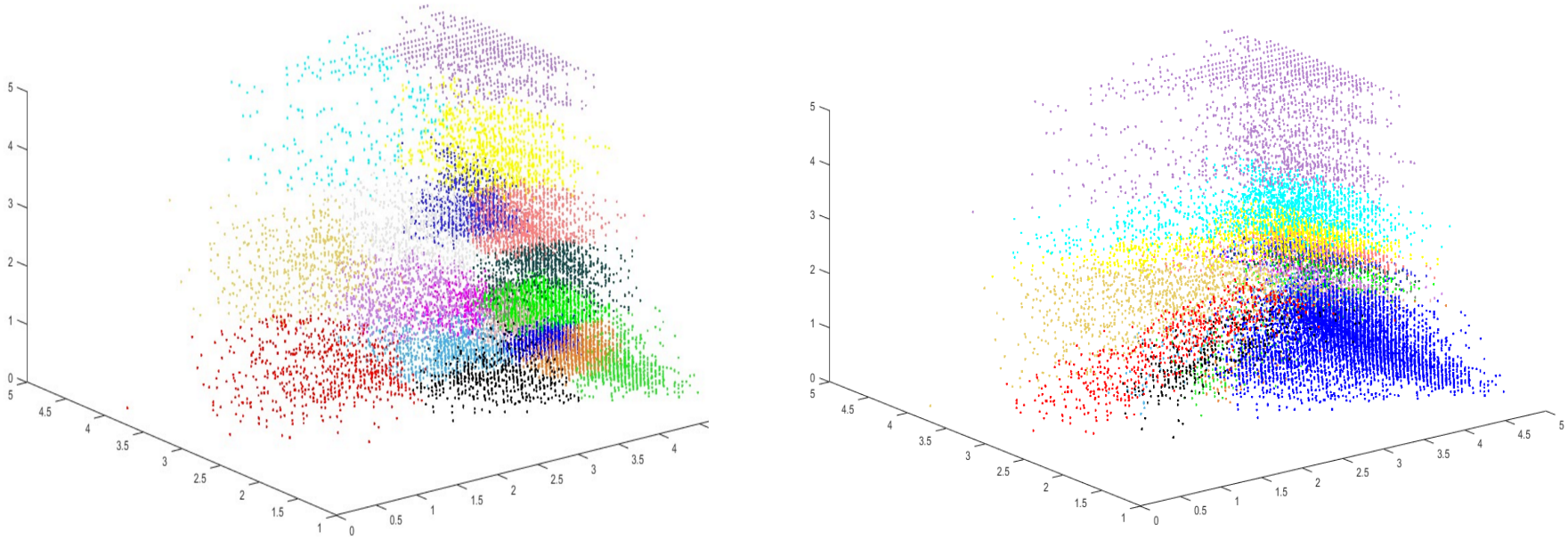
```
if(Double.parseDouble(tempTokens[i+1])<500){
    points[line][i]=0.0+Double.parseDouble(tempTokens[i+1])/500;
}else if(Double.parseDouble(tempTokens[i+1])<5000){
    points[line][i]=0.5+Double.parseDouble(tempTokens[i+1])/5000;
}else if(Double.parseDouble(tempTokens[i+1])<10000){
    points[line][i]=1.0+Double.parseDouble(tempTokens[i+1])/10000;
}else if(Double.parseDouble(tempTokens[i+1])<20000){
    points[line][i]=2.0+Double.parseDouble(tempTokens[i+1])/20000;
}else if(Double.parseDouble(tempTokens[i+1])<30000){
    points[line][i]=3.0+Double.parseDouble(tempTokens[i+1])/30000;
}else if(Double.parseDouble(tempTokens[i+1])<40000){
    points[line][i]=4.0+Double.parseDouble(tempTokens[i+1])/40000;
}else{
    points[line][i]=5.0;
```



before weighted vs after weighted(K-means)

- **Weighted attributes**
- **All attributes in similar magnitude while maintain it's difference**

Clustering



Weighted K-means vs weighted fuzzy K-means

- **Soft margin**
- **Allows empty cluster**

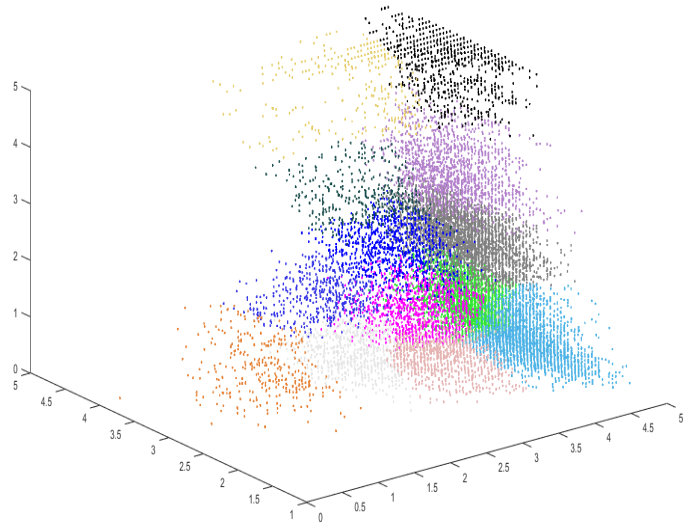
Clustering

Problems @ Documentation Declaration Console

<terminated> App4 [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (Dec 15, 2015, 12:41:50 AM)

```
Canopy id: 0 center: {0:4.149695008091578,1:3.1906448400348606,2:1.1807787138470494}
Canopy id: 1 center: {0:3.564339141870789,1:2.9094258589511743,2:1.0836860533453903}
Canopy id: 2 center: {0:3.985935687235949,1:3.503900794606301,2:3.3283965567060023}
Canopy id: 3 center: {0:2.6535870035671807,1:3.430083234244946,2:1.3943706302021386}
Canopy id: 4 center: {0:4.0260592031425375,1:3.689274691358027,2:4.489420059156381}
Canopy id: 5 center: {0:3.2026592455163874,1:3.5221768707483077,2:2.657842675736962}
Canopy id: 6 center: {0:3.9664897146648954,1:2.778102189781022,2:2.0335491970802915}
Canopy id: 7 center: {0:1.4137293320964752,1:3.33061224489796,2:0.9661673469387754}
Canopy id: 8 center: {0:2.450617283950616,1:3.971604938271605,2:4.4911827160493845}
Canopy id: 9 center: {0:1.2871900826446283,1:3.586363636363636,2:2.3341674242424246}
Canopy id: 10 center: {0:3.9454545454545453,1:4.2,2:1.283326666666667}
Canopy id: 11 center: {0:3.1909090909090905,1:3.56000000006
Canopy id: 12 center: {0:1.2272727272727273,1:4.0,2:3.46182
Canopy id: 13 center: {0:2.5,1:2.1,2:0.193}
finished
```

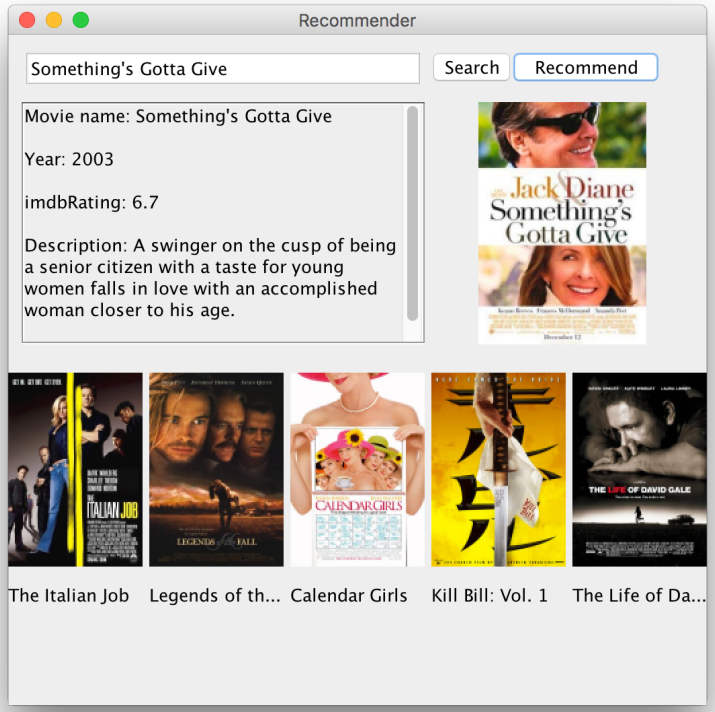
Canopy K means



- **Find reasonable cluster center first**
- **Then do K means based on above cluster center**

Two different recommender:

- **Recommendation based on the mahout recommender**
  - ➢ Pearson Correlation Similarity
  - ➢ Nearest User Neighborhood
- **Recommendation based on the aforementioned movie clustering result**
  - ➢Recalculate the distance between all other movies to the target movie in the same cluster.



Recommendation based on the mahout recommender

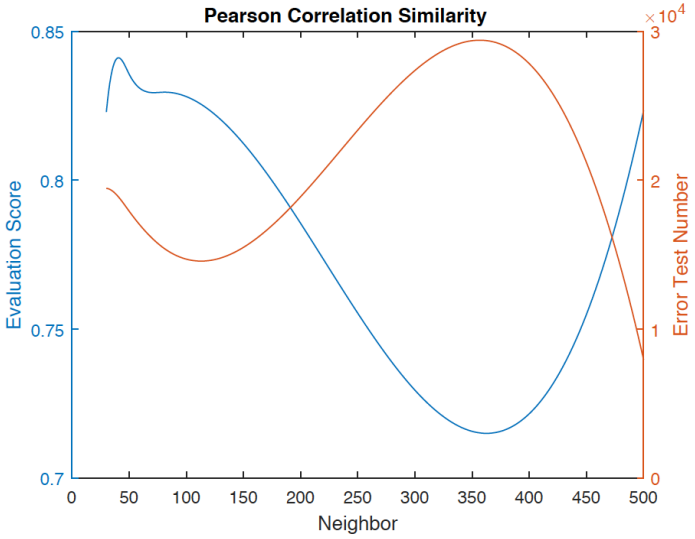
- **Evaluation of the optimal neighbor number**

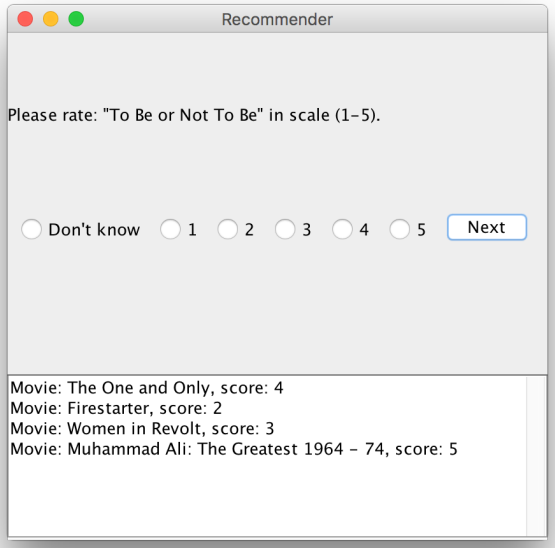
  Considering both the evaluation score and the failed test case, set 100 as the neighbor number.

- **Randomly picked up movie recommender**

  Randomly select movies for users to rate. Based on the rating result, output recommendation.

- **Drawbacks: *EXTREMELY Slow***





Recommendation based on the movie clustering result

- **Select the corresponding cluster of the target movie**

  Set the target movie as the center point. For all movies in the same cluster, calculate the distance with the target movie using the clustering algorithm once more.

- **Using OMDB API for more detailed information and friendly user interface**

  Click on the Poster, browse the imdb website for more movie information.