UCD DUBLIN

UCD School of Computer Science and Informatics
Scoil na Ríomheolaíochta agus na Faisnéisíochta UCD

COMP40320
Adaptive Personalisation

Recommender Systems

Submitted by,

Zihui Li
14201175

Introduction

The project is a case-based recommender system which is used to recommend movies for the users. The given datasets are user profile and movie profile, the user profile contains movie ratings and reviews, while the movie profile contains movie features. The system would analyse the two datasets then give recommendations for each user, and then try to test the accuracy by using the test dataset. Here we use the Precision and Recall to evaluate the whole system.

From Task 3 to Task 9, different methods were used to intended to improve the Precision separately, including adding new features, improvements on computing feature similarities, changing feature weights when computing the case similarities and implementing the content-based recommending system. However, not every method leads to an improvement. What's more, for those methods that make a progress on Precision, they have different performances on the improvement.

In this report, Task 3, 5 and 8 would be contained, with three different methods: setting movie ratings as a new feature, setting genre similarity as a new feature and Pearson Corelation function used on computing the case similarity. First two are focusing on feature similaritis while the last one cares more on the case similarity. In each part, the descriptions and hypothesis would be given first. Then the resulsts and conclusions were included. Finally, a conclusion of this whole project would be given, including the main findings, further idea for the future work and especially the limitations of the content-based method.

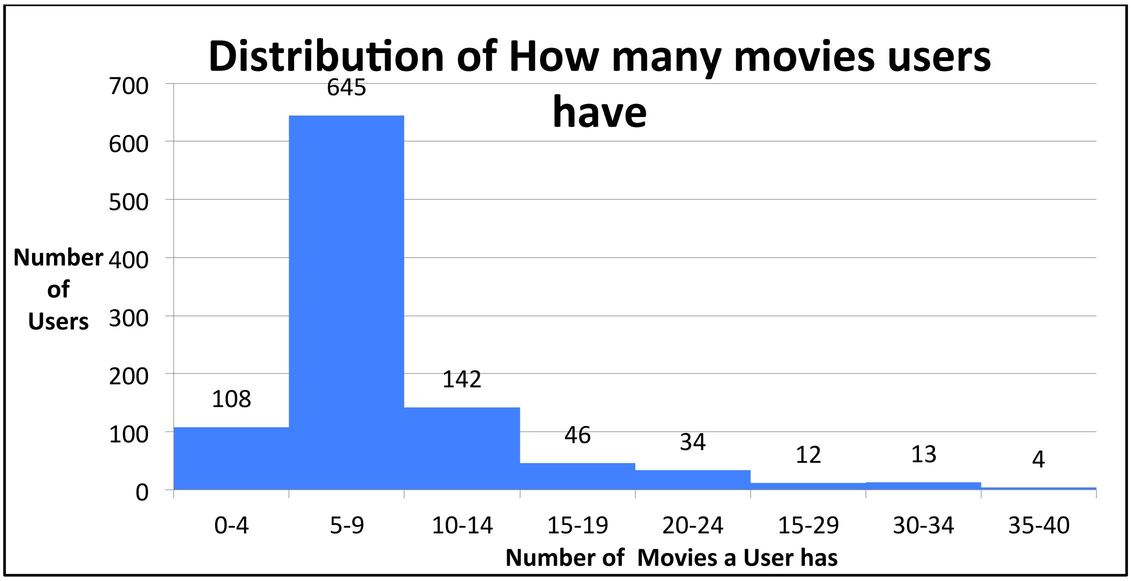
Dataset Statistics

a. Compute the number of users and movies in the dataset (use file trainData.txt).

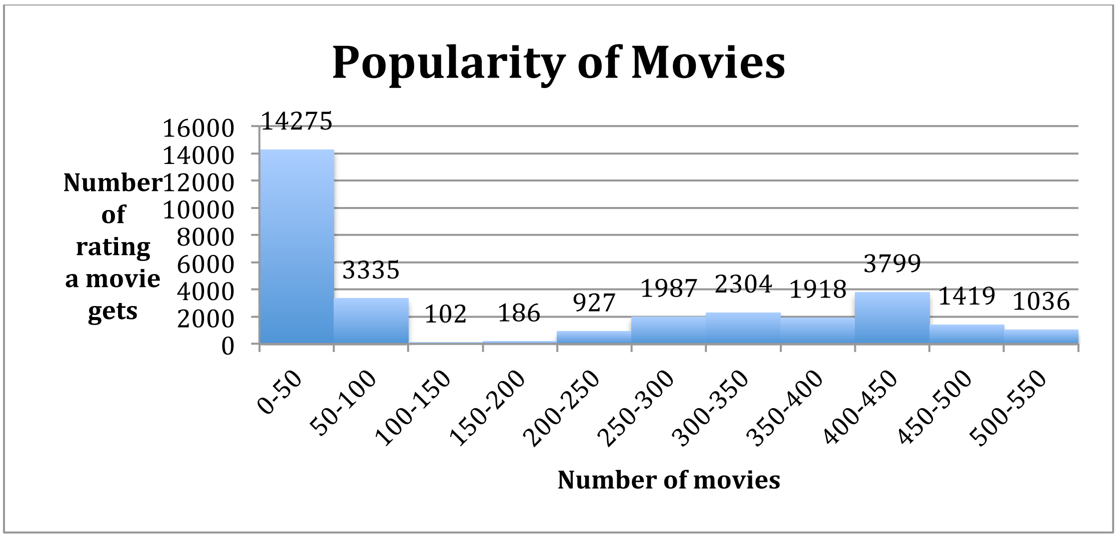
User number is 1000, movie number is 1073.

b. Plot histograms of the number of movies each user has in their profile and the popularity of each movie (use file trainData.txt).

The histogram below shows the number of movies that occurred in the profile of the users. X is the number of movies that a user has in his profile, while Y means the number of users. For example, the first bar means that, there are 108 users that have 0 to 4 movies in the profile.

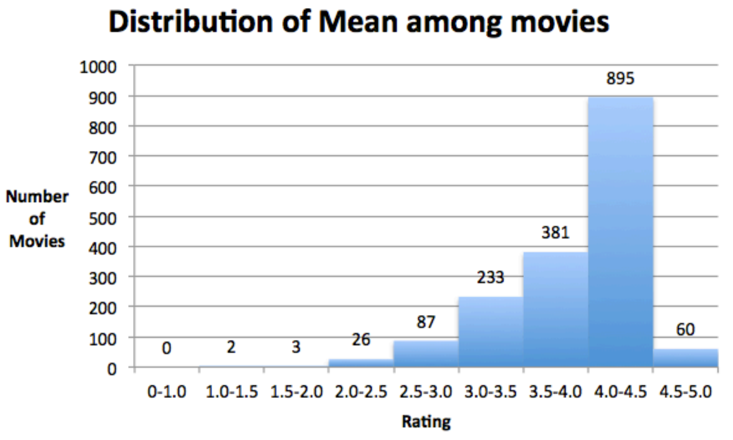


The graph given below showed the popurlarity of the movies. Y is the number of ratings that a movie gets from the profile of users. X gives the range of numbers. Let's say, the last bar means that there were 500-550 movies received 1036 ratings(sum).

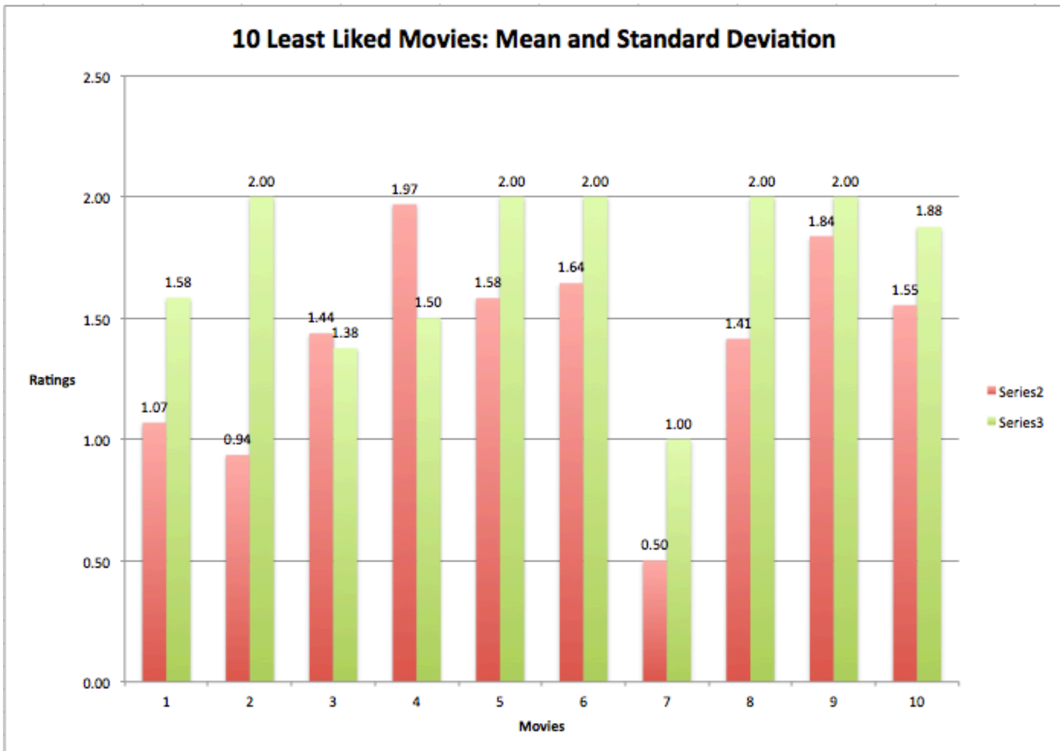


c. For each movie, compute the mean rating it has received. Plot a histogram of movie mean ratings. Identify the 10 most liked and 10 least liked movies – show (in a table or graph) the mean and standard deviation of ratings received by each of 10 most liked and 10 least liked movies (use file trainData.txt).

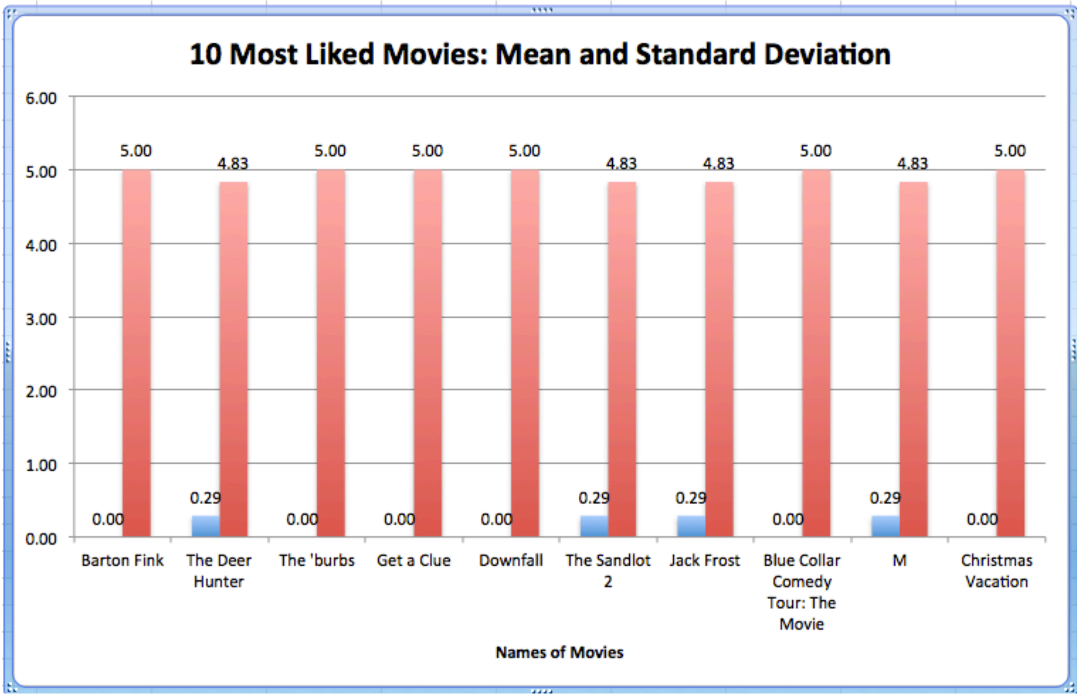
The graph given below illustrates the histogram of the movie mean ratings, with the specific numbers attached on the left. X axis is the range of rating, Y axis is the number of movies. The highest bar means that there were 895 movies that have the mean rating in the range of 4.0-4.5.



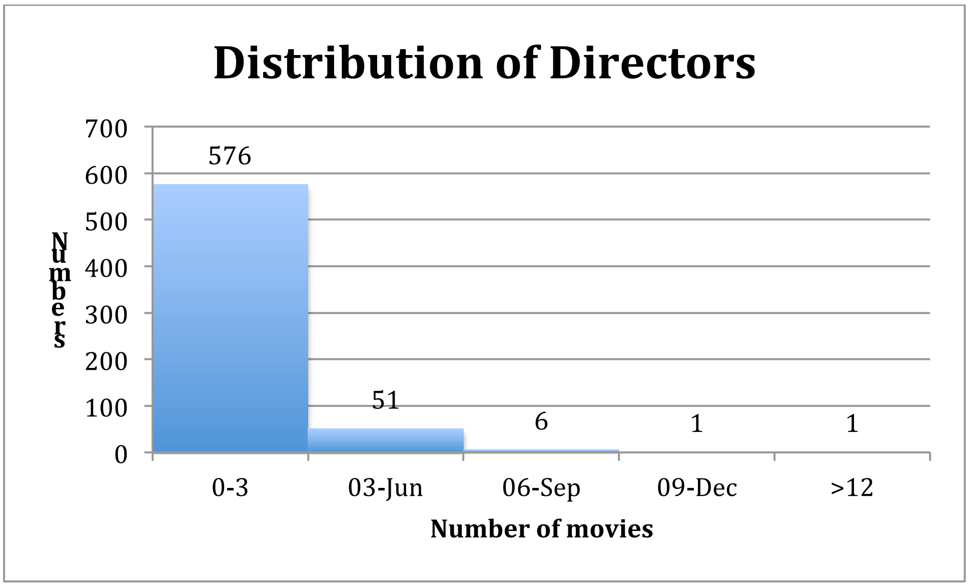
The least liked 10 movies are shown below, with a table and a bar chart(with the red bars pointing to the standard deviation while the other one pointing to the mean). While the name are given as well.



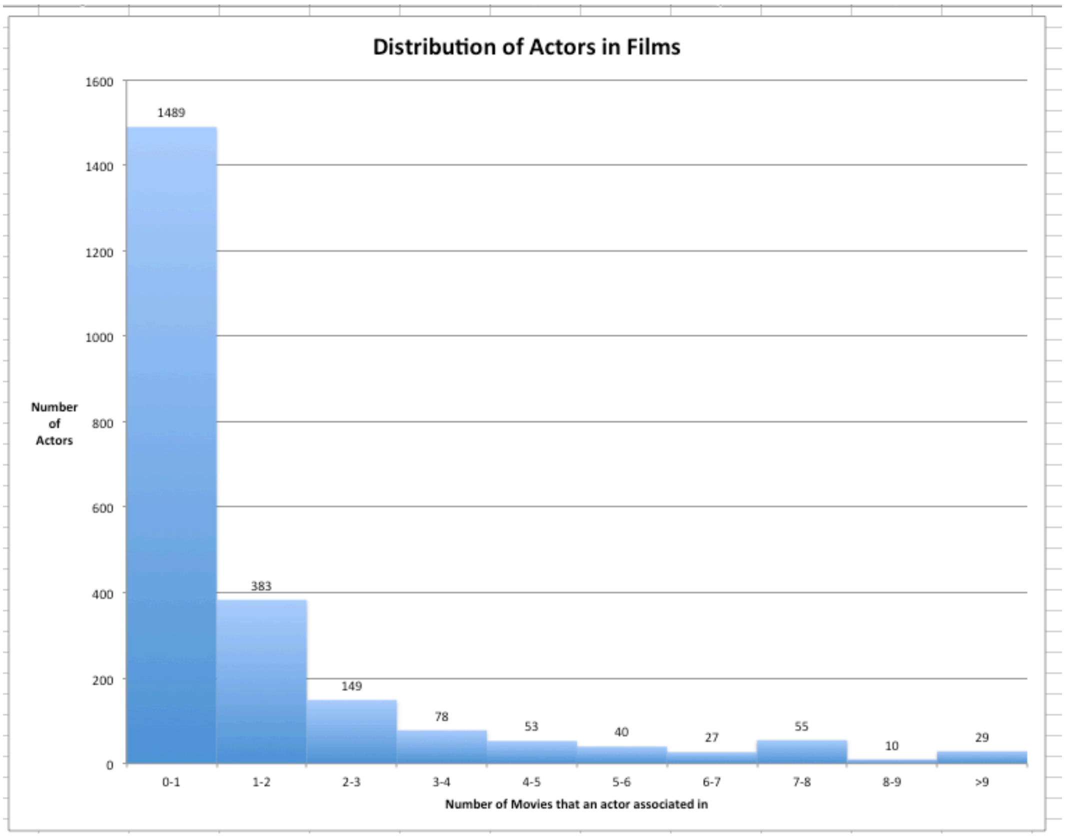
The 10 most liked movies are given below. While I think it is not a very good idea to put the standard deviate and mean into one graph, the reason is that it is a little bit hard to find out the fit range on Y axis. For the most liked movies have a high rating, usually above 4 or even equals to 5. Which means the good movies, at the same time, have a common high level of rating which result in a low standard deviate. It is obvious that here in our dataset, the standard deviate are all below 1.0. So a table might be a better way to show the data here.



d. Plot a histogram showing the number of movies each director is associated with. (Below)

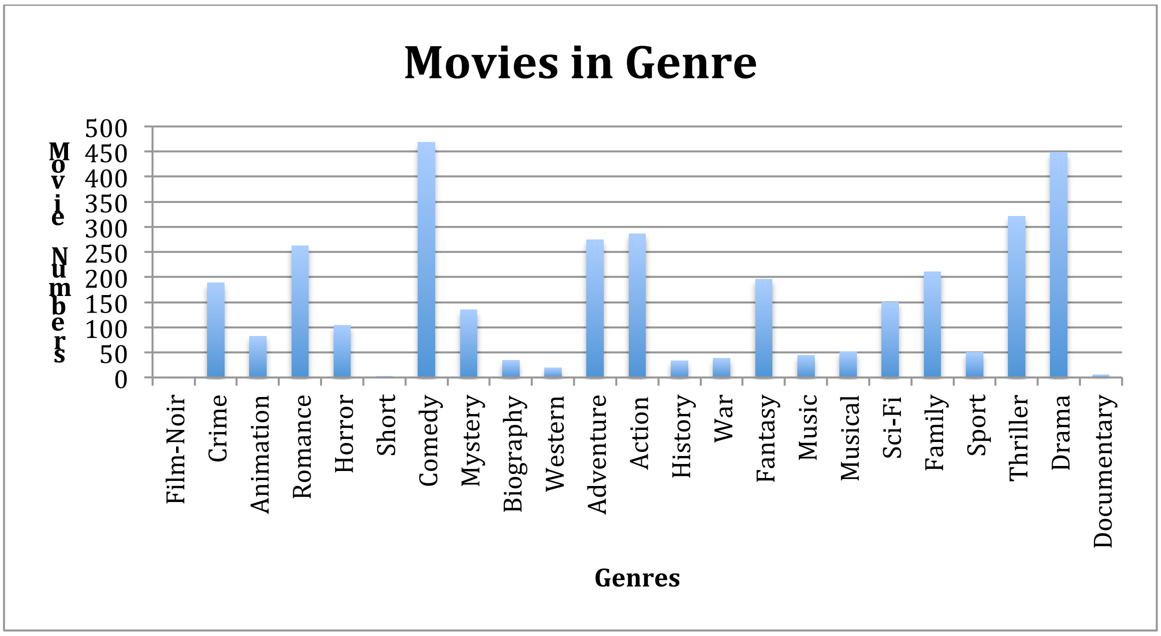


Plot a histogram showing the number of movies in which each actor appears (use file movies.txt). (Shows below)

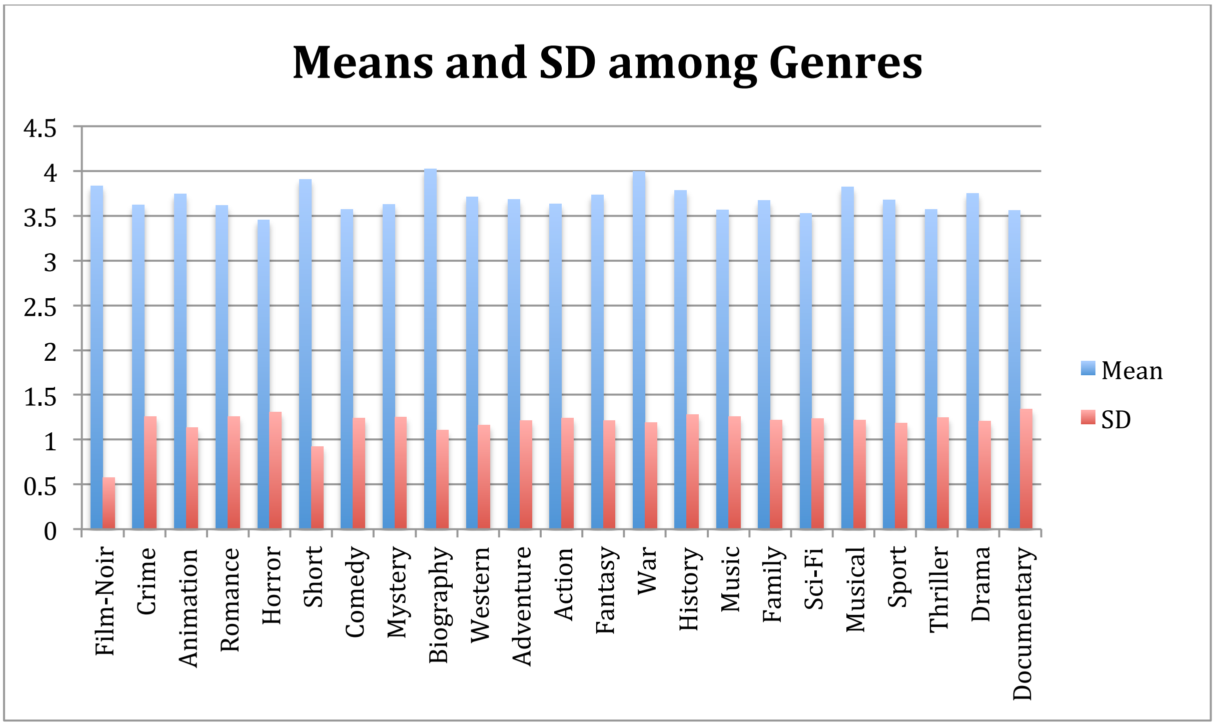


e. Plot the number of movies associated with each genre.

The graph below is the distribution of the movies in genres.



For each genre, compute and plot the mean and standard deviation of ratings assigned to all movies of that genre (use files trainData.txt and movies.txt).



Report on Task 3 – Two New Features

1.Brief Description and Hyperthesis.

Apply symmetric/asymmetric numeric feature similarity metrics as appropriate when comparing the values of these features between cases. Compare which feature in which way could make progress. The two equators are shown below:

🔷 $$\mathrm{Sim\_1}(v_C, v_T) = 1 - \frac{|v_T - v_C|}{\max(v_T, v_C)}$$

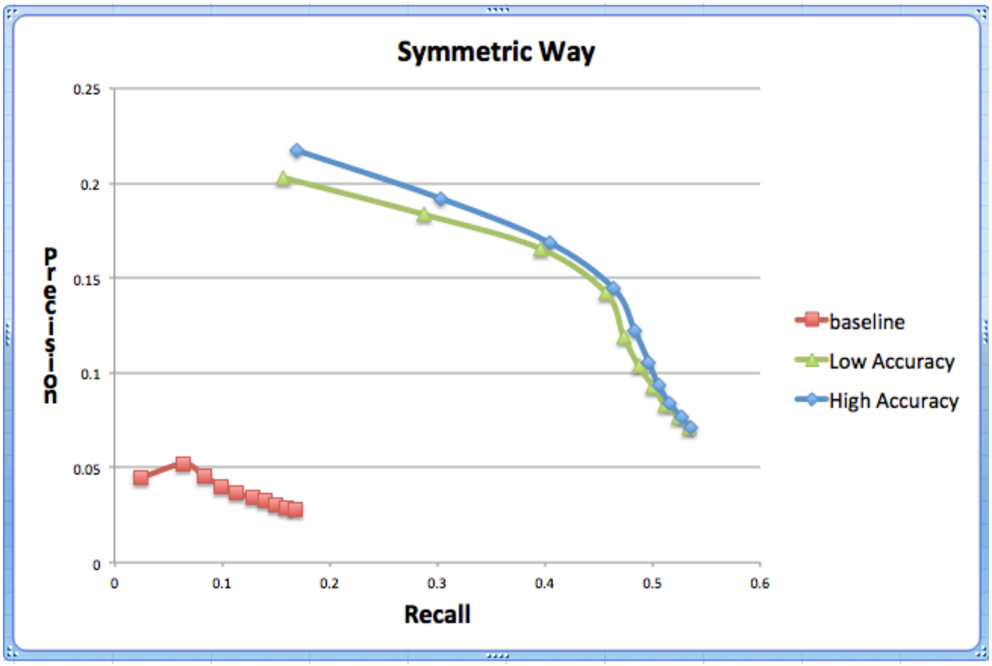
🟥 $$\mathrm{Sim\_2}(v_C, v_T) = 1 - \frac{|v_T - v_C|}{v_T + \max(0, (v_T - v_C))}$$

The hyperthesis is that they have the same performance among all the features, while both will lead to the improvement on Precision.

2. Results and Explanations

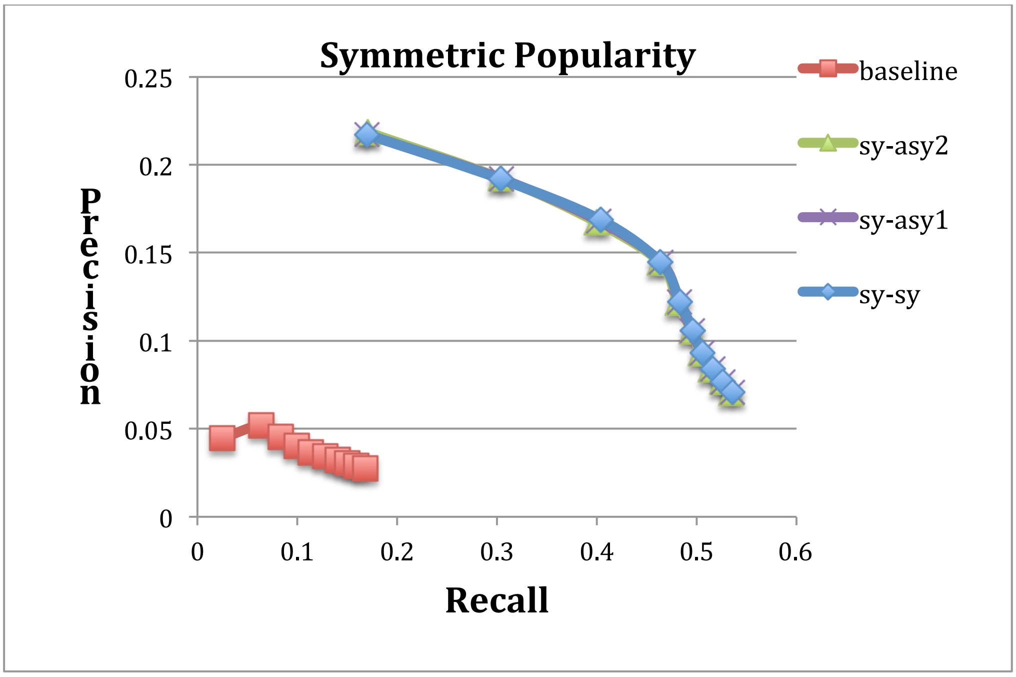
2.1 Symmetric on both popularity and mean rating

See the graphs below:
(we are using both symmetric way on popularity and mean rating)



It shows that the High Accuracy line, which is the second way, gives a little bit high precision.
The reason might be, in Java, int 5 divides by int 3 would get 1, but double 5 divides by double3 would get 1.66667. So there is a great difference before transferring and after.

2.2 Symmetric Popularity.

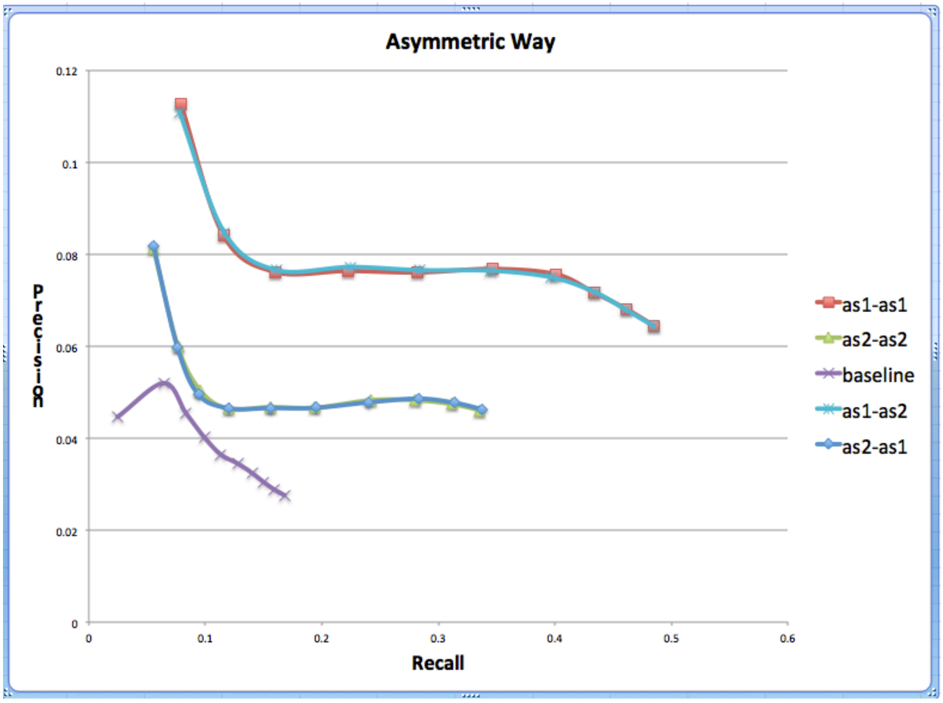


Four lines in the graph: baseline, sy-sy(symmetric mean and popularity), sy-asy1(symmetric popularity and asymmetric mean high) and sy-asy2 (symmetric popularity and asymmetric mean low).
Besides the baseline, the three seem to have a great improvement on the precision.

They only have slight difference(error on 0.001 on precision) on the data although they look the same.
According to the graph, the two ways(high and low) on asymmetric ways nearly have the same performance. So I made a hyperthesis that the two equators shown before have same performance in our case.

Let's see how we can prove it in the next section of the report.

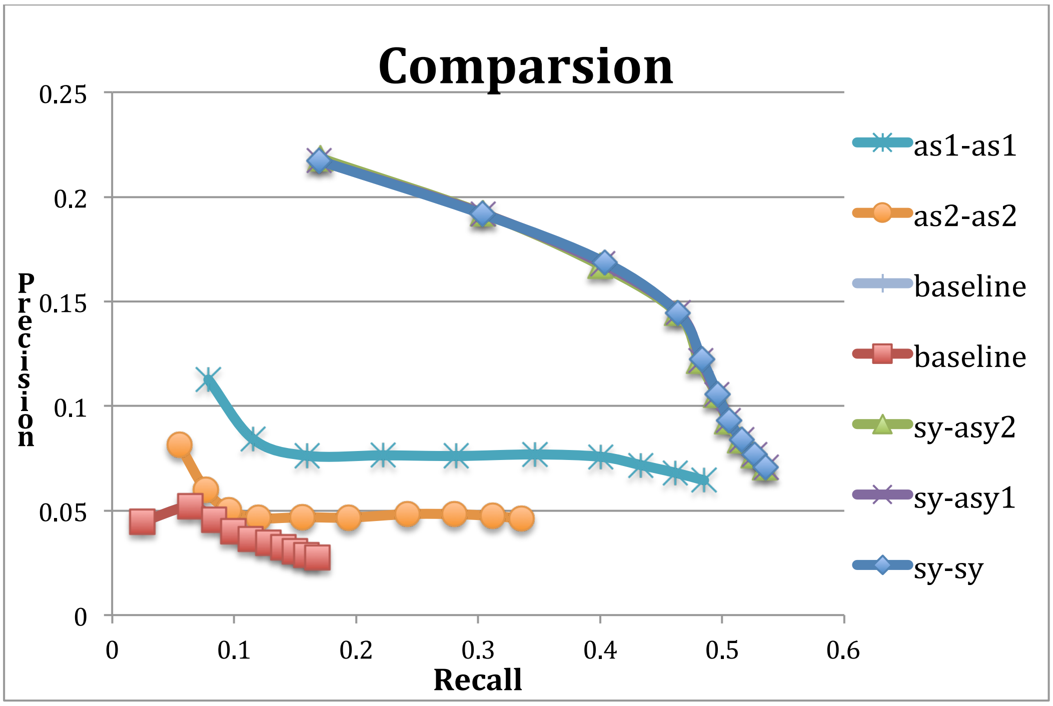
2.3 Asymmetric matrics.



Asymmetric Matric also provides better precision shown from our graph.
Five lines in the graph: base line, as1-as1(high popularity, high mean), as2-as2(low popularity, low mean), as1-as2(high popularity, low mean) and as2-as1(low popularity, high mean).
We can see that, the two lines, as1-as1 and as1-as2, nearly overlapping. Same with the as2-as1 and as2-as2. So it means, in the mean rating feature, the two matrics perfom *nearly* the same(we can still find in the data there are errors like 0.001 or 0.002 on precision). So it seems that the hypothesis we made in step 3 is right.

3 Conclusion and Problems

As the high or low asymmetric matrics on the features are nearly the same, so we eliminate parts of data, and genereate the graph:

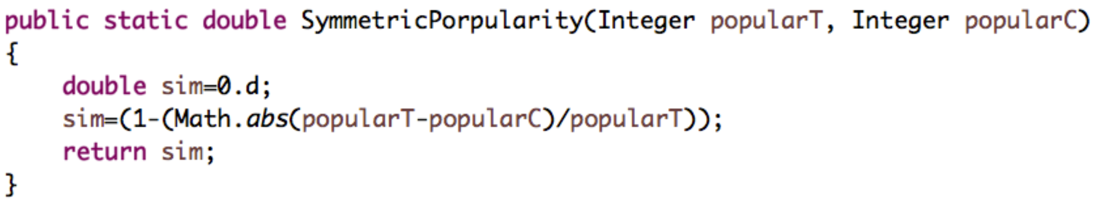


We can draw the conclusion that, when popularity is given the symmetric matric, it doesn't matter which way the mean rating is using, and that's the best case which perfoms highest precison.

When the popularity is using the high asymmetric matric, it's better than the low asymmetric one. Same, no matter which mean rating is using(low or high, or symmetric), that would not make any great changes on the chart.

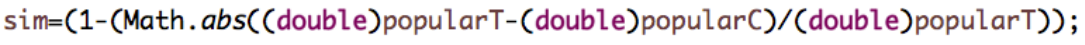
So the popularity effects more than the mean rating. If we should give them different weights, popularity should be given greater than mean rating.

When implementing the symmetric/asymmetric numeric feature similarity metrics, especially when using the popularity feature, because the data type is Integer, so different ways showing below will lead to different results:



```
public static double SymmetricPorpularity(Integer popularT, Integer popularC)
{
    double sim=0.d;
    sim=(1-(Math.abs(popularT-popularC)/popularT));
    return sim;
}
```

If the function writes like this, although the popularT and popularC are Integers, it will be automatically transferred to double when calculating. But if writes in this way:



```
sim=(1-(Math.abs((double)popularT-(double)popularC)/(double)popularT));
```

The accuracy will be rasied.

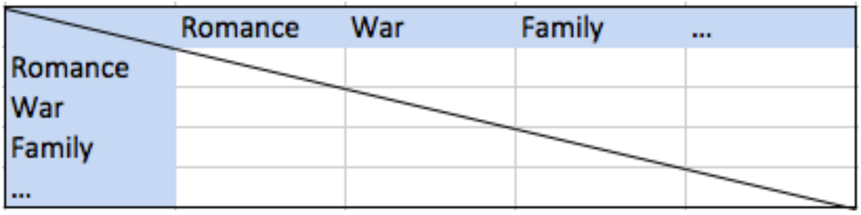
Report on Task5—Genres Similarity

1. Brief Introduction

The main idea for the genres similarity is based on the consideration of the following example:
If a movie has two genres{Romance, Tragedy} , the other has {Tragedy, War }. So there might be possibility that the system will recommend a {War} movie to a user that has {Romance} genre movies in his/her profile. Because there are some connections between Romance and War.
So if we still use the Jaccard method to compute the similarity, the situation above would be eliminated. Here we are trying to improve in this way.
The hypothesis is the Genres similarity would lead to improvement on Precision.

2. Method

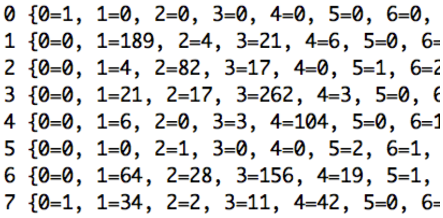
2.1.Generating a matrix of occurrence of each genre.



	Romance	War	Family	...
Romance				
War				
Family				
...				

Occurrence Matrix

First scan the movie.txt, first store each movie in a hashmap, then after processing, a matrix can be generated, see the part of the result below:



```
0 {0=1, 1=0, 2=0, 3=0, 4=0, 5=0, 6=0,
1 {0=0, 1=189, 2=4, 3=21, 4=6, 5=0, 6=
2 {0=0, 1=4, 2=82, 3=17, 4=0, 5=1, 6=2
3 {0=0, 1=21, 2=17, 3=262, 4=3, 5=0, 6
4 {0=0, 1=6, 2=0, 3=3, 4=104, 5=0, 6=1
5 {0=0, 1=0, 2=1, 3=0, 4=0, 5=2, 6=1,
6 {0=0, 1=64, 2=28, 3=156, 4=19, 5=1,
7 {0=1, 1=34, 2=2, 3=11, 4=42, 5=0, 6=
```

Matrix (part)

Here, number of 0,1,2… represent the different genres, each line is a hashmap with the key being a genre and a value being the times it occurs. So we can get the matrix.

2.2. Compute Similarities.
For each **genre pair**, sim(a,b):

$$\text{Sim(a,b)} = \frac{\text{Elem(a,b)}}{\text{Total number of Movies}}$$

Note: Elem(a,b) is the element in the row a, column b in the matrix.

For the computing **similarity of cases**:

$$\text{Similarity(c1 , c2)} = \frac{\sum_{i \in c1, j \in c2} \text{sim(i,j)}}{|c1| \times |c2|}$$

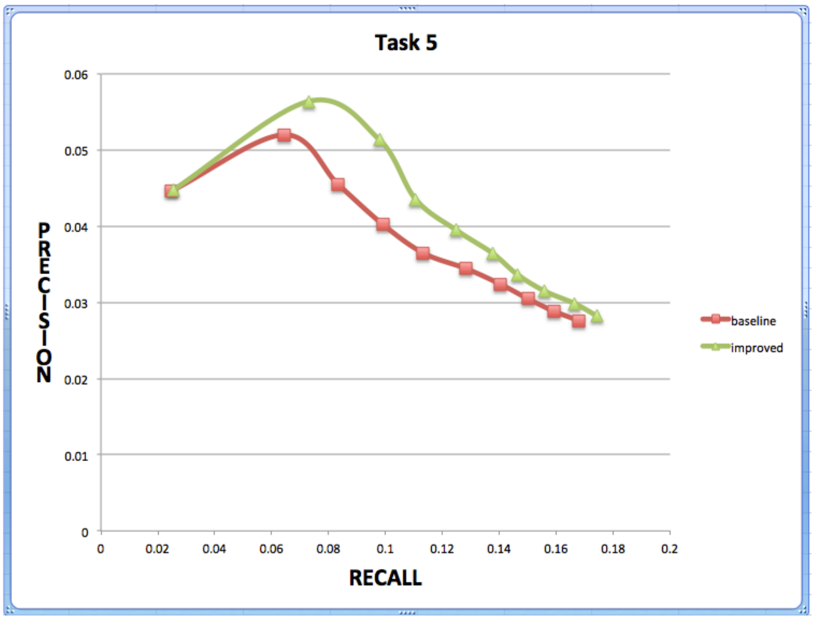
Note: |c1| means the size of set c1.

E.p. Compute the two cases: c1{a,b,c} and c2{d,e}:

$$\text{Similarity(c1,c2)=}\frac{\text{sim(a,d)+sim(a,e)+sim(b,d)+sim(b,e)+sim(c,d)+sim(c,e)}}{3\times 2}$$

3. Result and Explanation.

The result and the comparison with the baseline (Max, Jaccard from the original version of the program) could be seen below:



Comparison

It shows that the improved similarity algorithm performs better than the baseline, even though the first point is a little lower. And the other nine points have higher precision and recall at the same time. The equators are:

$$\text{Precision} = \frac{|T \cap R|}{|R|}$$

$$\text{Recall} = \frac{|T \cap R|}{|T|}$$

One possible situation is that if size of T and R we recommend keep same as before improved, size of T^R raised after we improved, thus giving a higher precision and recall at the same time.

Our hypothesis is right.

4. Problems

For the way of computing similarity between each genre pair, we can also define all the elements with same row and column numbers to be 1. Because a genre has totally similarity of 1 with itself. But here, if we are computing two cases like c1{a,b} and c2{a,b}, the equator should be:

$$\text{Similarity(c1,c2)=}\frac{\text{sim(a,a)+sim(a,b)+sim(b,a)+sim(b,b)}}{2\times 2}$$

Where sim(a,a) and sim(b,b) equals to 1 respectively, however, sim(b,a) and sim(a,b) might less than 1, that means the similarity(c1,c2) would be less than 1. But in our understanding, c1 and c2 have the same genres which means the similarity must be 1.

Here the problem is that, we are talking about the global similarity, which means, only if two cases have the same 23 genres, we give the similarity of 1. It is easy to understand. Lets say, the similarity of {a}and {a} should be less than the similarity of {a,b} and {a,b}, because the later have two in common.
But I didn't figure out a better way to compute a reasonable similarity after I got the matrix of occurrence, although the precision in the result seems improved.

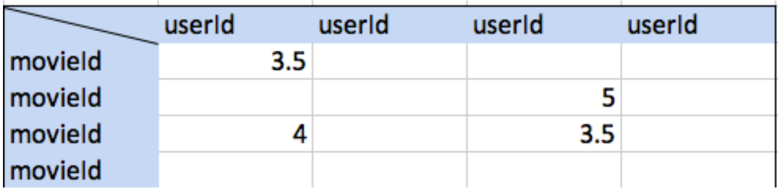
Report on Task 8 --- Pearson Correlation

1. Brief Introduction and Hypothesis

In Task2, we implemented changes based on rating, the result shows that it has a high improvement in the Precision. So rating is a weighted feature among others. So in Task8, I implemented Pearson Correlation on ratings.

2. Method

2.1. User-Movie Rating Matrix
First we generate an user-item(movie) Matrix with the ratings.



	userId	userId	userId	userId
movieId	3.5			
movieId			5	
movieId	4		3.5	
movieId				

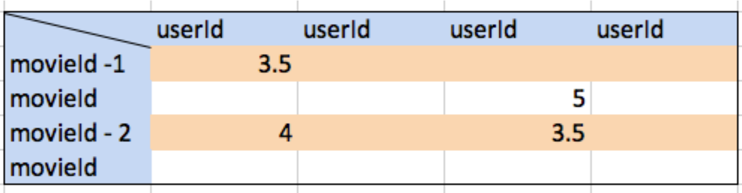
User-Movie Rating Matrix

Row index is the movie Id, and the Column index is the user Id. It is a sparse matrix, because we have 1000 movies in our dataset and for one user, he might not have too many ratings in his profile.
The data structure we are using to store this matrix is the Matrix class. It is a HashMap actually:

Map<MovieId,Map<UserId,Rating>>

The function getElement(movieId, userId) would return the rating of the movie of this user.

2.2. Pearson Correlation
In the FeatureSimilarityNew class, a new method named PearsonRating is defined to compute the Pearson Correlation.
When given two cases, let's say movie1 and movie 2, first we get the rows of the Matrix. An example is given below:



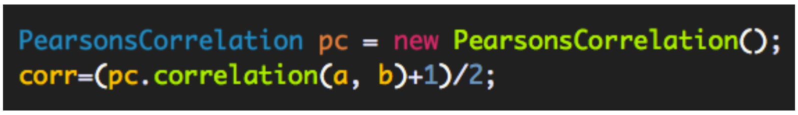
	userId	userId	userId	userId
movieId -1	3.5			
movieId			5	
movieId - 2	4		3.5	
movieId				

Rows Chosen

Then we get two vectors: [3.5, 0, 0, 0] and [4, 0, 3.5, 0]. The function to calculate the Pearson is :

$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

We can use : org.apache.commons.math3.stat.correlation.PearsonsCorrelation
To help us to calculate the Pearson Correlation. The brief implement is given below:



```
PearsonsCorrelation pc = new PearsonsCorrelation();
corr=(pc.correlation(a, b)+1)/2;
```

where: corr is a double number of the Pearson Correlation,
a and b are Array of double.

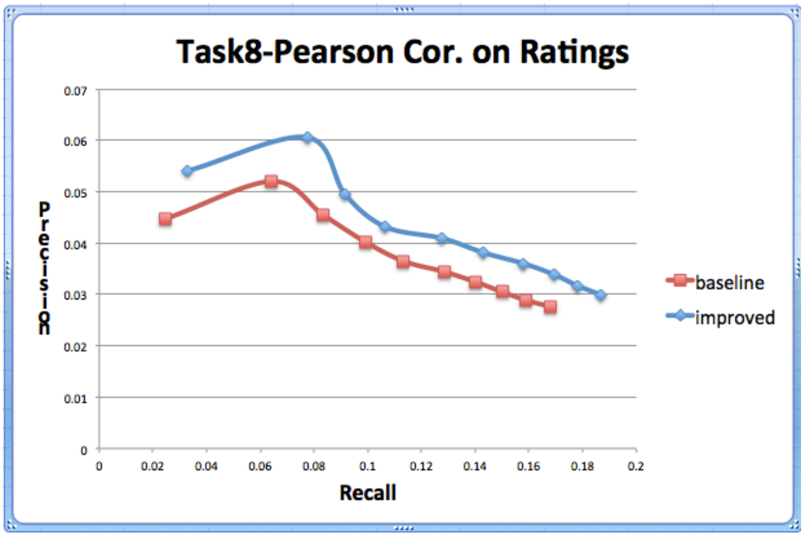
Since the Pearson Correlation is between 1 and -1, we cannot have a negative value in calculating similarity, so normalization is needed(Line 2 in the screenshot).

2.3. CaseSimilarity

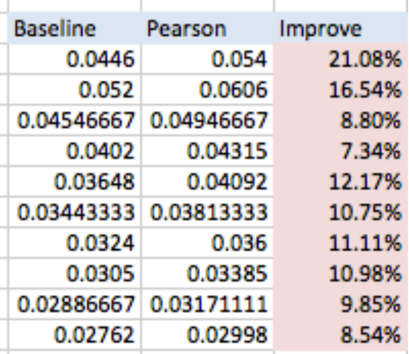
After we get the Pearson Correlation, we have to add it to the getSimilarity function. Then a new RATING_WEIGHT is set up, with the same value of 1 as other feature weights.

Rating weight is also need to be added in the below. Pay attention to the parameters passed to the PearsonRating function, they are the movie rating list, as we discussed before, the vector of ratings.

3. Result and Conclusion



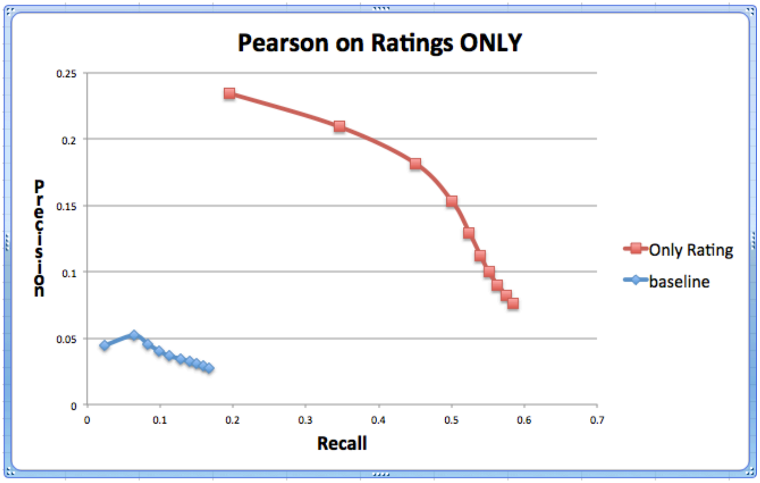
With the implement of the Pearson Correlation on the ratings, we can see that there is an improvement on both Precision and Recall. The data could give the details:



Baseline	Pearson	Improve
0.0446	0.054	21.08%
0.052	0.0606	16.54%
0.04546667	0.04946667	8.80%
0.0402	0.04315	7.34%
0.03648	0.04092	12.17%
0.03443333	0.03813333	10.75%
0.0324	0.036	11.11%
0.0305	0.03385	10.98%
0.02886667	0.03171111	9.85%
0.02762	0.02998	8.54%

Improvements on Precision

For the first point, the Pearson way improved more than 20% on the Precision, the second point also gives a big progress. So the first-two points are more sensitive to the rating feature. And the rating feature is more important than the other features. The graph below gives another situation: ONLY compute the similarity by the rating feature, which means the other weight of features are given the value of 0.



Only with Rating

The result gives even more improvement than the former one with a nearly 0.25 Precision of the first point.

Our hypothesis is right.

So the conclusion is that the feature rating is more important than the other features. While when designing the recommender system, it should weight more than the others because people might care more about the ratings than directors or actors.

Conclusions

1. Key Findings

Among all the features: genres, actors, directors, popularity and mean rating, the mean ratings is the most one that making improvements. That is to say, most people care more about the mean rating when compared to other features, like actors and directors.

For the case similarity, some methods like Pearson and Cosine, might improve the Precision, also they have the same trend with our baseline.

2. Limitations on content-base recommending system

On Task9, the matrix of document-term was generated to compute the documents similarity. The documents are the reviews of each movie from all the users that have this movie in the profile.

When generating the matrix, the simple way is to count how many times that a certain term appears. However, after ignoring some stop words, there is a risk that an opposite meaning would be get. For example, “won’t be fancy”, if ignoring “ won’t ” , we would get “fancy” only, indicating that the user like the movie, but actually he might not love it.

At the language level, one limitation is that, the synonyms would be taken into consideration, especially in multiple languages. The words: “good”, “fine” in English, and the words “bon”,”bien” in French have the same meaning. However, in our system, they would be count separately and took lots of room to store them as well.

3. Future work

The system is to recommending the existing movies to users. But if there is a new movie and no ratings or reviews at all, it could be possible to make predictions on ratings then recommended it to certain users. The answer is that we can calculate the movie case similarities based on the existing features (directors, actors, etc.).

Group recommending is another application as well. That is, how to recommend movies for a group of users. First, the user-based recommending system can be used to do the classification job. Then the case-based one(this system) can figure out the movies that could be recommended.