Collaborative Filtering-based Recommendation System for Amazon Movies

Diru Jia, Wenqi Kou, GQ Gaoqu
Industrial Engineering and Operations Research, UC Berkeley

Abstract: *Movies is an increasingly popular category on Amazon. Different from regular products, people's preferences of movies contain valuable information and features. By studying the reviews of movies, it's easy to get customers' tastes and preferences of movies as well as measuring similarity between movies. In this paper we tried a simple KMeans clustering model and also used an item-based collaborative filtering model to make recommendations.*

1. Introduction

E-commerce platforms have a large variety of commodities, which makes it difficult for users to choose due to information explosion. In addition, e-commerce often faces the problem of high advertising cost but low conversion rate. Thus, E-commerce is an excellent platform for personalized recommendations.

At present, almost all mainstream shopping APPs have functions such as "relevant items" and "guess what you like". Instead of relying on expert rules, the recommendation system can achieve more accurate and efficient personalized recommendation based on a large amount of commodity data and user behavior in the scene. Specifically, the recommendation system processes diversified information such as customers' purchasing power, brand loyalty, consumption frequency, consumption interest, etc., to form user portraits, and conduct large-scale personalized recommendations. As a result, with the rise of click through rate, conversion rate and advertising ROI, the customer scale and GMV continue to increase.

Based on the Amazon Review Data, we are going to build a recommendation system, in order to save advertising cost and increase GMV by recommending customers the products they are interested in and more likely to purchase.

2. Data Preprocessing

2.1. Data Description

Our project used the Amazon review dataset released in 2014. This dataset contains two types of data, one is product review data, including ratings, review text, etc., and the other is product information data, including product description, category, brand, price, etc. The updated dataset in 2018 contains all Amazon review data of all product categories from May 1996 to October 2018, with a total of 233.1 million entries.

Due to computing power and storage constraints, we decided to narrow down the scope of our project and we selected the products data in the 'Movies and TV' category for the following analysis and modeling experiments, which consists of 203,766 entries of product information data and 4,607,047 entries of product review data.

2.2. Data Cleaning

As introduced above, we have two types of data, product review data and product information data. Product review data has four columns: Reviewer Id, Movie Id, Rating and Review time, which have no null value and format issue. However, for product information data, 6 out of 19 columns have more than 95% null values. We directly deleted those columns. The data also contains image and text type columns such as product image links, product titles and detailed descriptions. Since we are not going to implement complex NLP and CV models, we also excluded these 3 columns. The remaining columns contain the information of product category, subcategory, brand, price, ranking, etc. The processing methods for these columns are described in detail below.

2.2.1. Movie Type

Though we only downloaded the 'Movies & TV' Category data, there are several products from other categories like Sports & Outdoors in the dataset. Thus, we only selected Movies & TV in the column 'main category' and there are 203464 entries left after selection.There are also several sub-category under the main category. To further narrow down our scope, we selected 18 keywords of movie types, like 'Science Fiction' and 'Animation', and extracted movies whose subcategories contain these keywords. The number of movies under each movie type are shown in Figure 1.

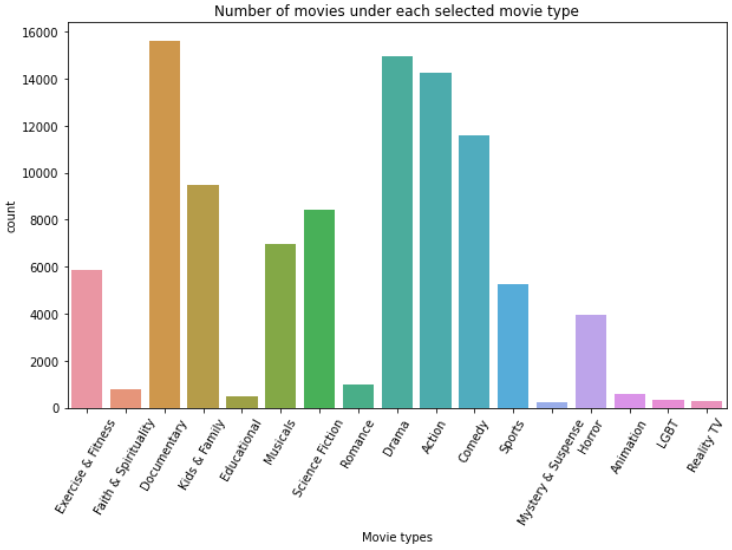

Figure 1. Number of movies under each movie type

2.2.2. Movie Rank & Price

The data type of both ranking and price information is string. Specifically, the value of column 'rank' is in the format of '370,026 in Movies & TV', and the value of column 'price' is in the format of '$74.95'. In this case, we used Regex to extract the target substring and convert it into integer.

Besides, we found 1% of the ranking data and 40% of the price data are missing. Thus, we filled in the missing data of the column ‘rank’ by the last value (ffill), and the missing data of the column ‘price’ by the mean value.

2.2.3. Movie Brand

Movie Brand actually refers to the movie director or movie actors/actresses. However, there are several meaningless symbols and words in this column, like ‘*’, ‘various’, etc. Thus, we used Regex to extract the names with both capitalized first name and last name.

2.3. Exploratory Data Analysis

2.3.1. Movie Reviews

For movie review data, each record represents a user's rating for a movie, so the granularity of this data is by user by movie. There are more than 2 million individual customers and a total of 200,941 movies in the dataset.

We calculated the average rating and the total number of reviews for each movie as shown in figure 2. Most movies have an average rating of greater than 3 and are mostly concentrated between 4 and 5. It can be seen that most of the users give ratings conservatively and will not give too low scores. Also, we found that more than 95% of the movies have less than 300 reviews and 90% of the movies have less than 40 reviews. Thus, we can conclude that most of the movie products on Amazon are long-tail products.

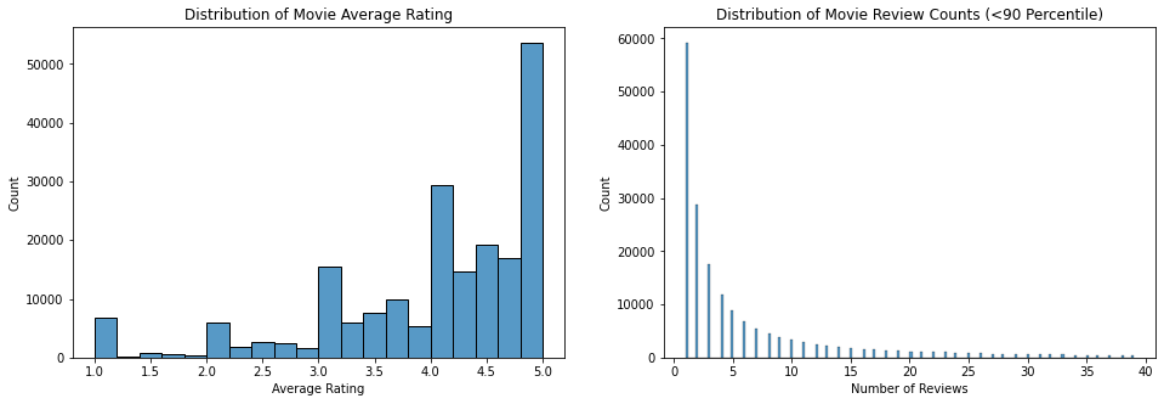

Figure 2. Distribution of average ratings and review counts of movies

We also calculated the average rating and the total number of reviews given by each customer. There is a customer that has written more than 2,500 reviews. But 90% of the customers only gave less than 3 reviews. Thus, we think it is more worthwhile to dig into information on movie level rather than user activity, considering that we do not have other user related data on hand.

2.3.2. *Movie Information*

We explored the two numerical features, ranking and price, in the movie data. There is no explicit explanation of the meaning of ranking, but we speculated that it represents the ordering of items within amazon. We found that there are duplicate rankings, that is, it is possible that two movies are on the same ranking.

For price, we found that the prices of more than 99% of the movies are lower than $100. The right plot in figure 3 shows the distribution of movie prices that is lower than $100. We can see that most of the movies have the price lower than $40 and around $20.

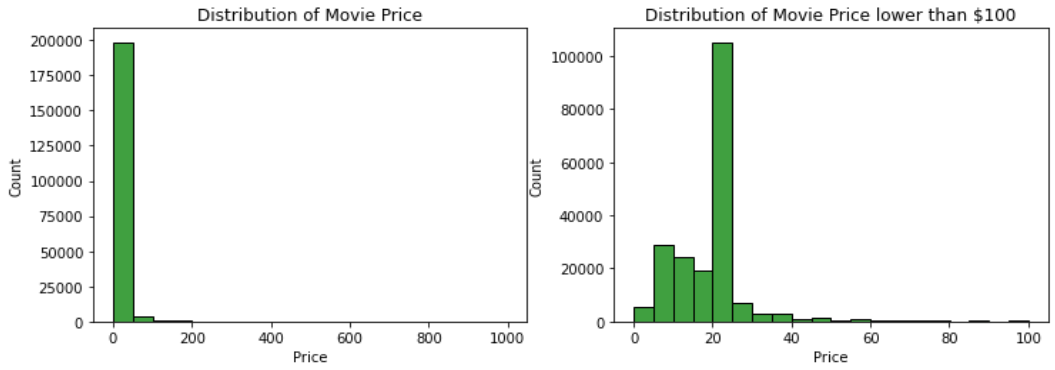

Figure 3. Distribution of movie prices

2.3.3. *Combine Two Data Sources*

In order to investigate the relationship between review activity and movie info, we combined the two data tables above.

First of all, we explored the average rating distribution and review counts distribution within each movie type. From figure 4, we can see that the ratings of Documentary, Kids & Family, Musicals and Faith & Spirituality movies concentrate mainly on score 5 and the ratings of Science Fiction, Romance, Drama and Action movies are distributed in all score segments. It is worth noting that the rating of horror movies are relatively lower compared to other movie types. And the ratings of LGBT movies are almost evenly distributed.

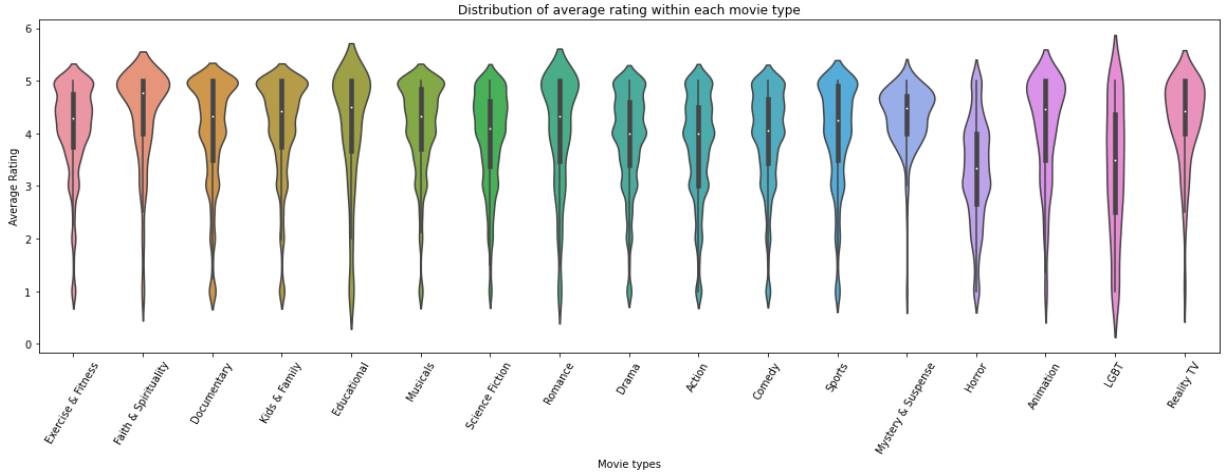

Figure 4. Distribution of average rating within each movie type

We further investigate the relationships among average rating, review counts and movie attributes. We can see from figure 5 that the average rating of a movie is negatively related to its review counts. Since most of the movies are reviewed less than 40 times, their average ratings are probably incomplete and biased to reflect the movie quality. Moreover, there is a negative correlation between review counts and rankings, that is, the more a movie is reviewed, the higher its ranking is (lower number). Thus, we can conclude that the ranking of a movie probably represents its popularity.

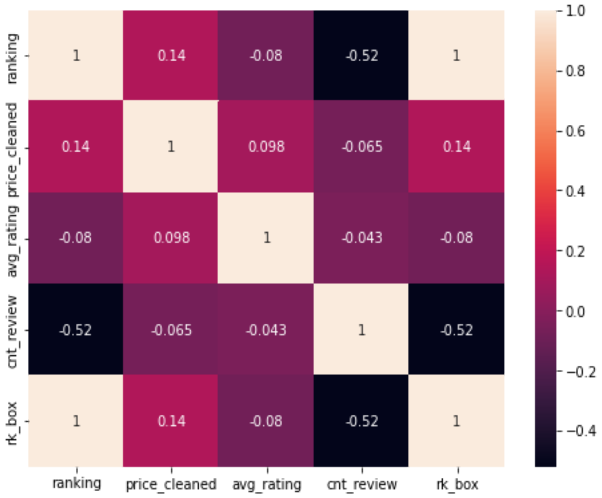

Figure 5. Heatmap of review and movie features

3. Modeling

3.1. Overview

Our recommendation system was built in two ways. The first model was accomplished by an unsupervised clustering model, while the second revised model was implemented by collaborative filtering method. We used the KMeans model to cluster customers' movie rating matrix, and then voted the clustering labels based on customers' historical browsing data and recommended the whole corresponding cluster of movies. Although this recommendation method has a high detection rate, too many movies were recommended at one time, the precision is only 0.92%. Therefore, we used item-based collaborative filtering to make small-scale movie recommendations to customers.

3.2. Data Sampling

Even after the data cleaning, we still faced a large sparse dataset. However, due the amount of data is too large to be processed by Google Colab, we decided to select part of the data for modeling and analysis. In the original dataset, there are 4,607,047 reviews, 2,088,620 reviewers and 200,941 movies. Our subsequent modeling and analysis would only focus on those movies that have been watched more than 500 times and those customers who have seen at least 10 or more movies. As a result, there are 167,833 reviews, 7,614 customers and 1,278 movies in the remaining selected dataset.

3.3. Data Splitting

Before modeling, the dataset was divided into a training set and a test set. In order to simulate the use of historical data to predict the future in the real forecasting scenarios, we should keep those events that occur after the events used to fit the model as the test set. Therefore, we sorted the customer ratings' table by the timestamp of the reviews, and then splitted the training set and test set in that chronological order.

3.4. Baseline

In the `movie_info` table, column `also_buy` represents other movies purchased by users who purchased that movie. Our baseline model assumes that customers who have purchased this movie will also like other movies purchased by other people who have purchased this movie. Then, the baseline model would find the `also_buy` movies and recommend them based on the historical customers' reviewing data. This baseline model is naive but we would compare the recommended results of the other two models with the baseline to prove whether our model is effective or not.

3.5. Movie Clustering

Clustering is to group data objects into multiple clusters. Its goal is to have high similarity among objects in the same cluster, while objects in different clusters differ greatly. In this case, we assumed that customers would like similar movies, so we used the KMeans model to cluster the movies and then recommended similar movies to customers. We first clustered the movie with 4 clusters on `price`, `avg_rating`, `cnt_review` and `rk_box` attributes from the cleaned `movie_info` table.

At the same time, we reconstructed the rows and columns of the customers' movie rating table, and obtained a table with rows of `movie id`, columns of `customer id` and corresponding values of the customer's rating of the movie. Our second clustering method would cluster the movies according to customers' ratings of the movies. In this case, the granularity of the data is the movie, and each customers' rating to the movie is taken as a feature.

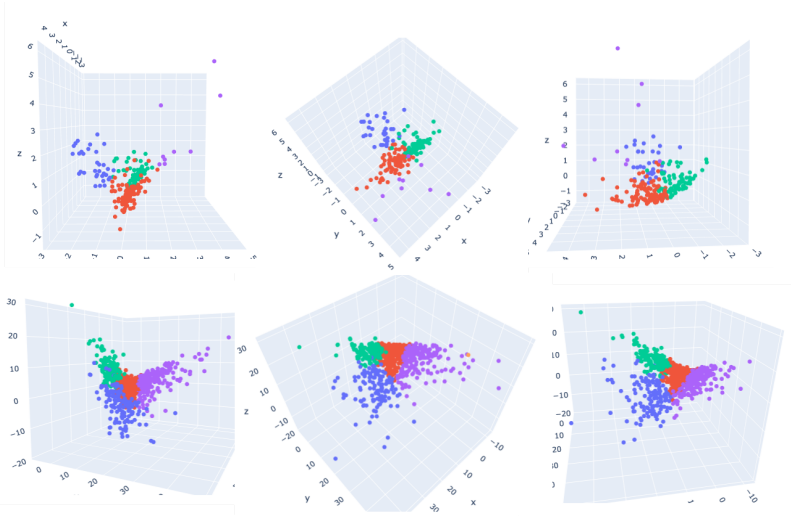

Figure 6. 3D movie clustering result (method 1 - top, method 2 - bottom)

We used PCA dimensionality reduction to extract the three most representative dimensions of the two matrices respectively, and then we visualized and compared the results of the two clustering approaches in the 3D plots (Figure 6). We found that the second clustering method performed better. As a result, we would vote the clustering labels based on customers' historical browsing data and recommend the whole corresponding cluster of movies based on the second clustering result.

3.6. Item Collaborative Filtering

Collaborative Filtering (CF) is rapidly becoming a popular technology in information filtering systems. We chose to use item based CF which assumes that similar items are likely to be liked by the same user. In other words, items liked by the same user are similar items. For example, movie 1 and movie 2, both of which are liked by customer A, are similar. Movies 5 and 6, however, are not liked by the same customers at the same time, so they are not similar.

Customer	Movies
A	[a, b, c]
B	[a, b]
C	[a, b, d]
D	[a, c]
E	[b, c]

a	b	c	d
4	4	3	1

	a	b	c	d
a		3	2	1
b	3		2	1
c	2	2		0
d	1	1	0	

Figure 7. Example for counting the number

In order to build this model, we first counted the number of times that each movie was liked by customers and the number of times that each movie was liked by multiple customers

at the same time by using the hash dictionary. For example, suppose there are five customers A, B, C, D, and E (Figure 7). Customer A likes movies a, b, and c, and customer B likes movies a, b, and so on. Movie a and movie b are favored by customer A, B and C at the same time, so the co-occurrence times of movie a and movie b are 3. This statistical method could be used to quickly construct the co-occurrence matrix. The next step was to calculate the similarity among each movie. The formula is shown below.

$$W_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$

N(u) represents the number of customers who like the movie u, N(v) represents the number of customers who like the movie v, and the intersection of the two represents the number of customers who like both movie u and v, that is the value in co-occurrence matrix. For example, the similarity between movie a and movie b can be calculated as below.

$$W_{ab} = \frac{|N(a) \cap N(b)|}{\sqrt{|N(a)||N(b)|}} = \frac{3}{\sqrt{4 \times 4}} = \frac{3}{4}$$

The more people like the two movies at the same time, the more similar the two movies are. Therefore, based on the similarity matrix calculated above, we could sort the similarity and recommend top-n movies to customers.

4. Results Analysis

4.1. Metrics Definition

To measure the performance of models, we used multiple metrics including precision, recall and detection rate. We first used the classic metrics, precision and recall. Here we counted the number of 'hit items', which means the items that are recommended to certain customers and were actually bought by the customers. Recall indicates how well our recommendation is while precision indicates how efficient our model is.

$$Recall = \sum_{all\ customers} \frac{Number\ of\ hit\ items}{Number\ of\ items\ that\ customer\ i\ bought}$$

$$Precision = \sum_{all\ customers} \frac{Number\ of\ hit\ items}{Number\ of\ recommended\ items\ for\ customer\ i}$$

We also have a new metric called detection rate, which measures the model on the customer aspect. If there exists one correct recommendation, then we regard the customer as a hit customer. Calculating the detection rate can help us better understand the overall performance of the model.

$$Detection\ Rate = \frac{Number\ of\ hit\ customers}{Number\ of\ recommended\ customers}$$

On the items aspect, we may address recall more than precision, for a correct recommendation can bring revenue. But the precision shouldn't be extremely high either. If our recommendation system offers so many non-relevant recommendations, customers will lose faith in our recommendation. So the number of recommendations is important and tricky to set.

4.2. Model Performance

We first sorted the review data by time, to get the history reviews of each customer. Then the data was split into a train and a test set. After KMeans clustering and collaborative filtering on the train data, we can get the similarity matrix among all movies in the certain cluster. In the predicting part, the model will do the recommendation based on the historical data of each customer. Knowing what the customers have bought before, the baseline will recommend all movies in the cluster while the CF-based model will recommend the first 10 movies with the highest similarity with the movies the user already bought. We also set the baseline model as a recommendation model based on the list of 'also_buy' of one movie. The performance of baseline model, model based on KMeans clustering and model based on Item-based Collaborative Filtering is shown below.

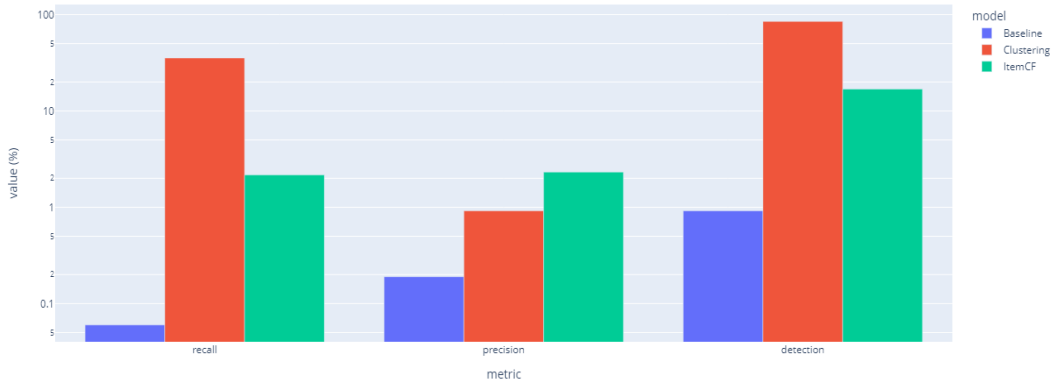

Figure 8. Performance on different models

The recommendation model based on KMeans clustering has really high recall and detection. But it's recommending so many movies to one customer, which is intolerant for user experience. So we used Item-based Collaborative Filtering first to study the similarity between movies. Then make recommendations based on the similarity. The precision is obviously higher.

4.3. Peer Review Implementation

5. Impact & Future Works

5.1. Potential societal impacts and/or ethical concerns

As a recommendation system, a classic issue is the customer's privacy. Using customers' historical data to make recommendations can lead to privacy issues. So we can also deploy it as an optional function, or a searching function. Giving the initiative back to the customers can effectively avoid privacy issues.

Additionally, malicious activities should be detected and interfered early. For example, frequent purchase of R level movies that includes sensitive or violent content should be awared. If the purchase history is already mainly about negative content, then the recommendation should definitely be stopped.

5.2. Discussion

Given that the amount of customers is huge, we here used item-based CF. But we can also leave more movies and delete more customer data in the data processing to see the performance of user-based CF.

Also, In the calculation of similarity, we were using a metric that is similar to cosine similarity metric, we can also try the euclidean distance as the metric. No matter if it's item-based or user-based, euclidean distance will be more reasonable for the calculation of similarity.

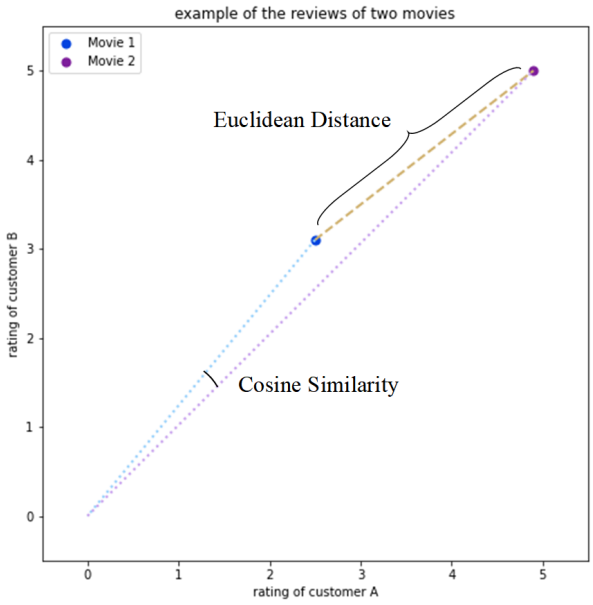

Figure 9. Two metrics

From the figure above we can see that cosine similarity indicates the difference of customer reviews. Movie 1 gets the same reviews from customers, and so is movie 2. In cosine similarity metric, the two movies are similar. However, the distance between two points is big, which indicates that they have different ratings on customers.

Actually, we can use the cosine similarity to measure the quality of customer-based clustering. If the clustering of customers is accurate, then those customers should have similar taste, so the cosine similarity on movies shouldn't be so diverse.

Last but not least, we can also try using a rating matrix to do the CF instead of just using purchase counts. Different from purchase counts of certain products, purchase of a movie doesn't necessarily mean a preference to the movie. Each purchase contains preference information - the rating. The purchase can be both negative and positive. So it'll be more reasonable to use the rating matrix to do the CF and calculate the similarity.