Video Recommender System

Shikhar Sanghvi (105054667), Priyanshi Shah (105050988), Rui Xu (105012245)

Abstract: Recommender systems are an important part of information and e-commerce ecosystem. They represent a powerful method for enabling users to filter through large information and product spaces. Nearly past years of research on collaborative filtering have led to a varied set of algorithms and a classic collection of tools and software for evaluating their performance. Research in the field is moving in the direction of a richer understanding of how recommender technology may be embedded in specific domains. The variety of personalities exhibited by different recommender algorithms show that recommendation is not a one-size-fits-all problem. Domain specific tasks, information needs, and item domains represent unique problems for recommenders, and design and evaluation of recommenders needs to be done based on the user tasks to be supported. Effective deployments must begin with proper analysis of prospective users and their goals. Based on this analysis, system developer has a host of options for the choice of algorithm and for its embedding in the user experience. This report describes what we did on the course project, including the details of how our video recommender system works and the techniques we used. The target of this project is to develop a video recommender system. The basic functions include recommendation of videos for particular user. Some other improvements are also provided, like testing and evaluation on different datasets. Also, the system is combined with some advanced algorithms, such as collaborative filtering, correlation coefficient, hold out test.

***Index Terms*— Collaborative filtering, Mahout, Movie, Recommendation System, User-based**

1. Acknowledgement

First and foremost, we would like to thank God almighty for giving us the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. We are blessed with his and our gaurdian blessings. In our journey towards this course, we have a teacher, a friend, an inspiration, a role model and a pillar of support in my Guide, Dr. Kobti, Instructor of 03-60-570-01 course. He has been there providing his invaluable guidance, inspiration and suggestions in our quest for knowledge. He has given us all the freedom to pursue our experiments, while silently and non-obtrusively ensuring that we stay on course and do not deviate from the core of our project. Without his able guidance, this project would not have been possible, and we shall eternally be grateful to him for his assistance. We have great pleasure in acknowledging our gratitude to our colleagues and fellow research scholars at University of Windsor. We take pride in acknowledging ourselves for such an incredible experience towards making this project. I would also like to thank all of our team members, Shikhar Sanghvi for implementing the technical part (coding), Priyanshi Shah for implementing the algorithms and Rui Xu for problem solving and working on datasets. All of us contributed in evaluation and testing part.

2. Introduction

People always want to see the videos that they like. So, Video Recommendation systems recommend the videos to the users by quick filtering of the information on the video website. Lots of companies have a recommendation system. For instance, Netflix and YouTube offer this service.

To define the problem statement, providing related content out of a relevant and irrelevant collection of videos to users of online service providers. In our project, we are trying to recommend videos to users, based on user-videos ratings by using collaborative filtering.

For example, User 1 watched movie 1 and movie 2 and give the ratings of these movies. The recommendation system is trying to recommend movie 3 for user 1 and predict the rating he will assign to the movie 3.

Apache Mahout is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focus mainly in the areas of collaborative filtering, recommendation and machine learning algorithm. Many of the implementations use the Apache Hadoop platform. Then, we use user-based collaborative filtering in our recommender system. As for the dataset, we use MovieLens 20M dataset. It is the dataset includes 20 million ratings to 27,000 movies by 138,000 users. Users of MovieLens were selected randomly and all users rated at least 20 movies. Each movie or user has a unique id. And the format of the

.

data is User id | Video id | Rating. The total rating possible is 138,000*27,000, which is about 3 billion. At the end of the project, we use "hold-out" test to calculate the evaluation result.

Our motivation was to explore more on recommender, so we try to survey on different methods of recommender and learn about the algorithms of the recommendation system. After mastering the algorithm, we work on creating a better recommender system to recommend videos to particular users with better evaluation results then existing algorithms.

3. Recommendation filtering techniques

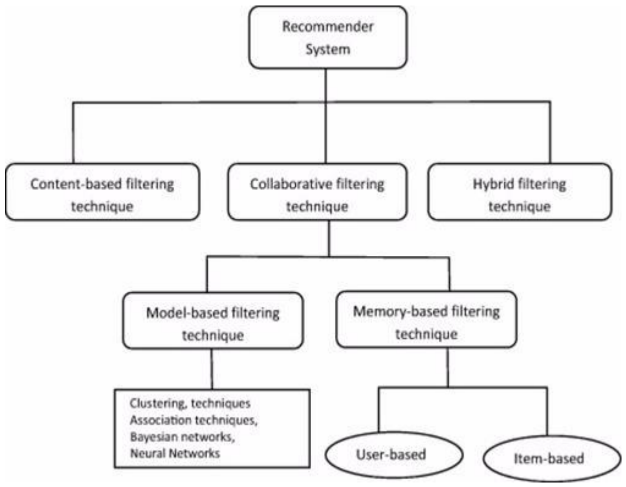

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Fig. 1. Recommendation Techniques

3.1 Content Based Filtering

Content based filtering method are based on a description of the item and a profile of the user's preferences.

The content-based filter analyzes set of descriptors feed to it as input for a particular item which is previously rated by the user. This filter then constructs a model of users' interest which generates recommendations. [4]

In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past. In particular, various user items are compared with items previously rated by the another user and the best-matching items are recommended. This approach has its base in information retrieval and information filtering research.

3.1.1. Pros and Cons of content-based filtering techniques

CB filtering techniques overcome the challenges of CF. They have the ability to recommend items even if there are not any ratings provided by user. So even if the database does not contain any user preferences, recommendation accuracy is not affected. Also, if the user taste change, it has the capacity to balance its recommendations in a short time. They can manage different situations of different users do not share the same items, but only identical items according to their features. Users can get recommendations without sharing their information, and this ensures privacy [11]. CBF technique can also explain on how recommendations are generated to users. However, the techniques suffer from various problems as discussed in the literature [6]. Content based filtering techniques are dependent on items' metadata. That is, they require rich details of items and very well-organized user profile before recommendation can be made to users. This is also called limited content analysis. So, the effectiveness of CBF depends on the availability of descriptive data. Content overspecialization [12] is another serious problem of CBF technique. Users are restricted to getting recommendations similar to items already defined in their profiles.

3.2 Collaborative filtering technique

The term "collaborative filtering (CF)" was coined by Goldberg et al., in 1992 who proposed that the information filtering process becomes more effective when humans are concerned [1][2]. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. [3]

This technique works by building a database that is user-item matrix of preferences for items by users. It then compares users with relevant interest and preferences by calculating similarities between their profiles to make recommendations.

Such users build a group called neighborhood. A user gets recommendations to those items that he has not watched before but that were already rated by the similar users in his neighborhood.

Recommendations that are produced by CF can be of either prediction or recommendation.

Rij, which is prediction is a numerical value expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most as shown in Figure 2. The technique of collaborative filtering can be divided into two categories: memory-based and model-based.

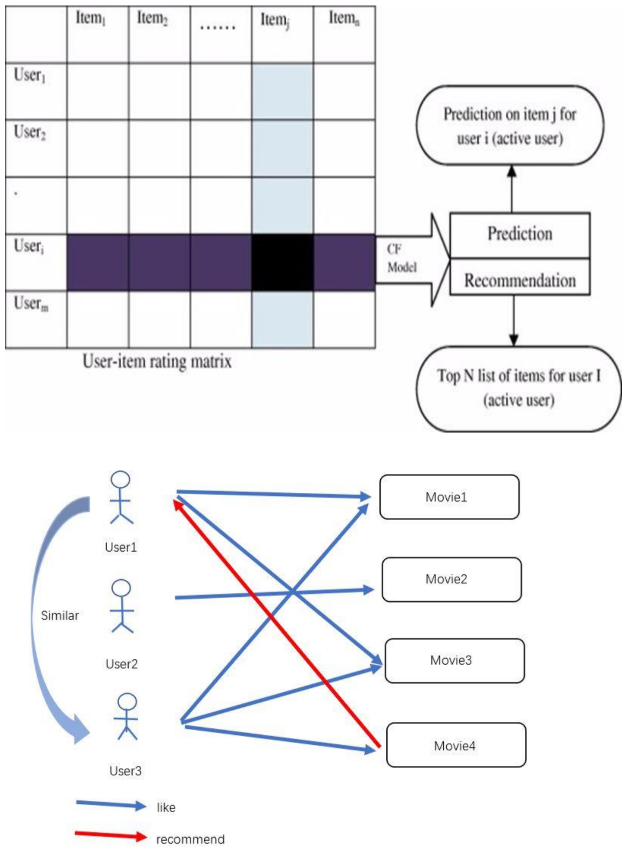


Fig. 2. Collaborative Filtering Process

3.2.1 Pros and Cons of collaborative filtering techniques

Collaborative Filtering has some major advantages over CBF in that it can perform in domains where there is not much content connected with items and where content is hard for a computer system to review (such as opinions and ideal). Also, Collaborative Filtering technique has the ability to provide serendipitous recommendations, which means that it can recommend items that are similar to the users even without the content being in the user's profile. Despite the success of Collaborative Filtering techniques, their widespread use has disclosed some problems.

3.3 Hybrid Filtering system

Hybrid Filtering system is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. Hybrid approaches can be implemented in different ways: by making content-based technique and collaborative-based predictions separately and then combining them; by adding content-based abilities to a collaborative-based approach; or by specifying the approaches into one model.

Hybrid approach wherein it utilizes an ensemble of content based and collaborative filtering approaches, which is called rules, to come up with a set of recommended videos. [5]

4. Phase of recommendation process

4.1 Information collection phase

A recommendation agent cannot function accurately until the user profile has been well developed. The system needs to know as much as possible from the user in order to provide accurate recommendation right from the dataset. Recommender systems rely on different types of input such as the most convenient high-quality explicit feedback, which includes explicit input by users regarding their interest in item or implicit feedback by inferring user preferences indirectly through observing user behavior [8]. Hybrid feedback can also be obtained through the combination of both feedbacks. In E-learning platform, a user profile is a collection of personal information associated with a specific user. This information includes cognitive skills, intellectual abilities, learning styles, interest, preferences and interaction with the system.. Thus, a user profile describes a simple user model. The success of any recommendation system depends largely on its ability to represent user's current interests. Accurate models are indispensable for obtaining relevant and accurate recommendations from any prediction techniques.

4.1.1. Explicit feedback

The system normally prompts the user through the system interface to provide ratings for items in order to construct and improve his model. The accuracy of recommendation depends on the quantity of ratings provided by the user. The only shortcoming of this method is, it requires effort from the users and also, users are not always ready to supply enough information. Despite the fact that explicit feedback requires more effort from user, it is still seen as providing more reliable data, since it does not involve extracting preferences from actions, and it also provides transparency into the recommendation process that results in a slightly higher perceived recommendation quality and more confidence in the recommendations [9].

4.1.2 Implicit feedback

The system automatically infers the user's preferences by monitoring the different actions of users such as the history of purchases, navigation history, and time spent on some web pages, links followed by the user, content of e-mail and button clicks among others. Implicit feedback reduces the burden on users by inferring their user's preferences from their behavior with the system. The method though does not require effort from the user, but it is less accurate. Also, it has also been argued that implicit preference data might in actuality be more objective, as there is no bias arising from users responding in a socially desirable way [9] and there are no self-image issues or any need for maintaining an image for others [10].

4.1.3 Hybrid feedback

The strengths of both implicit and explicit feedback can be combined in a hybrid system in order to minimize their weaknesses and get a best performing system. This can be achieved by using an implicit data as a check on explicit rating or allowing user to give explicit feedback only when he chooses to express explicit interest.

4.2 Learning phase

It applies a learning algorithm to filter and exploit the user's features from the feedback collected from the information collection phase.

4.3 Prediction/recommendation phase

It recommends or predicts what kind of items the user may prefer. This can be done either of the ways, based on the dataset collected in information collection phase which could be memory based or model based or through the system's observed activities of the user. Fig. 2 highlights the recommendation phases.

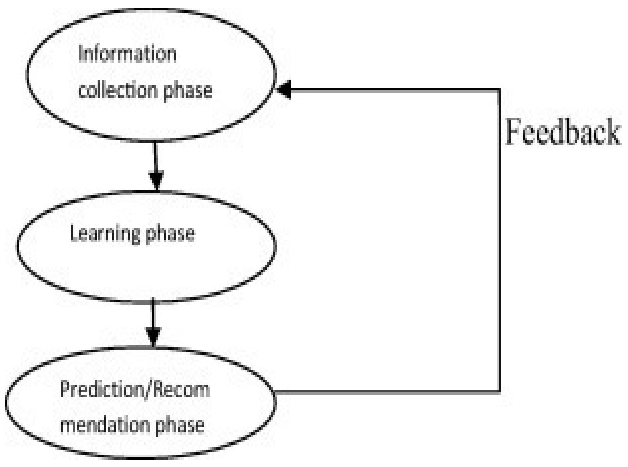

Information collection phase
Feedback
Learning phase
Prediction/Recom mendation phase

Fig. 3. Recommendation phase

5. Implementation of recommender system

5.1 Software Configuration

- Eclipse (Mars)
- Mahout 0.10.0 version
- Maven
- Applet Server
- Excel sheet (For dataset)

5.2 Prerequisites

Create a java project in your favorite IDE (We have use Mars version here) and make sure mahout is on the class path. The easiest way to accomplish this is by importing it via maven as described below.

Getting mahout: Download the latest version of mahout. Here is link http://www.apache.org/dyn/closer.cgi/mahout/ Or checkout the latest code on the below given link: http://mahout.apache.org/developers/version-control.html
Alternatively, Add Mahout 0.10.0 to a maven project
(We have chosen alternative path in our project)
Mahout is also available via a maven repository under the group that has a id name called apache mahout. If you would like to import the latest release of mahout into a java project, add the following dependency in your pom.xml:

```
<dependency>
   <groupId>org.apache.mahout</groupId>
   <artifactId>mahout-mr</artifactId>
   <version>0.10.0</version>
</dependency>
```

If you are building a recommender system for the first time, please also refer to a list of Dos and Don'ts that might be helpful.

5.3 Do and Don't list

- Don't start with a distributed, Hadoop-based recommender, take on that complexity only if necessary. Start with non-distributed recommenders. It is simpler, has fewer requirements, and is more flexible.
- As a crude rule of thumb, a system with up to 100M user-item associations (ratings, preferences) should "fit" onto one modern server machine with 4GB of heap available and run acceptably as a real-time recommender. The system is invariably memory-bound since keeping data in memory is essential to performance.
- Beyond this point it gets expensive to deploy a machine with enough RAM, so, designing for a distributed makes sense when nearing this scale. However most applications don't "really" have 100M associations to process. Data can be sampled; noisy and old data can often be aggressively pruned without significant impact on the result.
- The next question is whether or not your system has preference values, or ratings. Do users and items merely have an association or not, such as the existence or lack of a click? or is behavior translated into some scalar value representing the user's degree of preference for the item.
- If you have ratings, then a good place to start is a Generic Item Based Recommender, plus a Pearson Correlation Similarity metric. If you don't have ratings, then a good place to start is Generic Boolean Preferred Item Based Recommender and Log Likelihood Similarity.
- If you want to do content-based item-item similarity, you need to implement your own Item Similarity.
- If your data can be simply exported to a CSV file, use File Data Model and push new files periodically. If your data is in a database, use MySQL JDBC Data Model (or its "Boolean Pref" counterpart if appropriate, or its PostgreSQL counterpart, etc.) and put on top a Reload

from JDBC Data Model.

- This should give a reasonable starter system which responds fast. The nature of the system is that new data comes in from the file or database only periodically – perhaps on the order of minutes.

5.4 Dataset

In our project we have use three datasets. Two for training purpose and one actual dataset. Mahout's recommenders expect interactions between users and items as input. The easiest way to supply such data to Mahout is in the form of a text file, where every line has the format userID, itemID, value. Here userID and itemID refer to a particular user and a particular item, and value denotes the strength of the interaction (e.g. the rating given to a movie).

Training dataset:

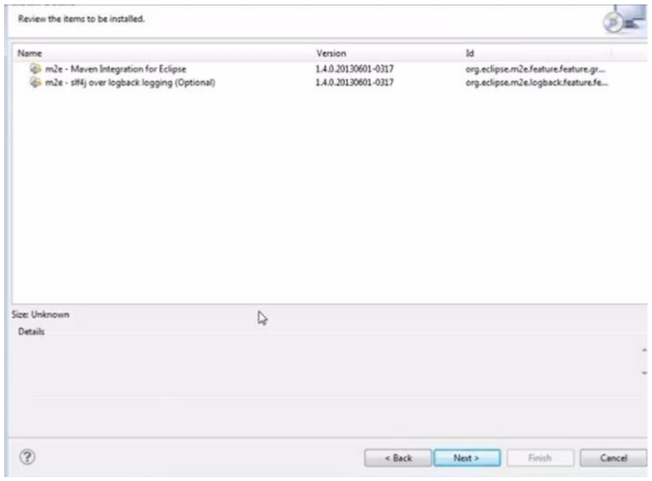
1) In this example, we'll use some made up data for simplicity. Create a file called "dataset.csv" and copy the following example interactions into the file from the link.
http://www.occamslab.com/petricek/data/ratings.dat
2) We use MovieLens 10M dataset as our secondary training set.
https://grouplens.org/datasets/movielens/10m/

Actual dataset:

In MovieLens 20M dataset , we have use rating excel sheet as our dataset as it was only require for our recommender system.
https://grouplens.org/datasets/movielens/20m/

5.5 Environment Setup

Step 1 - Download the latest version of maven(here we name as M2e)



Review the items to be installed.

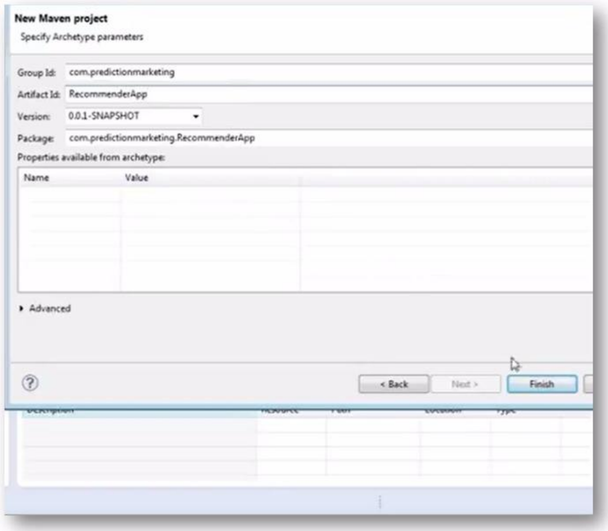
Name	Version	Id
m2e - Maven Integration for Eclipse	1.4.0.20130601-0317	org.eclipse.m2e.feature.feature.gr...
m2e - slf4j over logback logging (Optional)	1.4.0.20130601-0317	org.eclipse.m2e.logback.feature.fe...

Size: Unknown
Details

Fig.4.

Step 2 - Create Maven Project in Eclipse IDE. Follow the option File > New -> Project and finally select Maven Project wizard from the wizard list. Now name project as RecommenderApp using the wizard window as follows.

Step 3 - Add Mahout 0.10.0 to a maven project.



New Maven project
Specify Archetype parameters

Group Id: com.predictionmarketing
Artifact Id: RecommenderApp
Version: 0.0.1-SNAPSHOT
Package: com.predictionmarketing.RecommenderApp
Properties available from archetype:

Name	Value

Fig. 5.

Create Source file:

Step 4 - Let us now create actual source files under the RecommenderApp project. First we need to create a package as show on the figure. To do this, right-click on src in package explorer section and follow the option : New -> Package.
Next we will create App.java, Evaluate Recommender.java under the package

Step 5 - Add Required Libraries

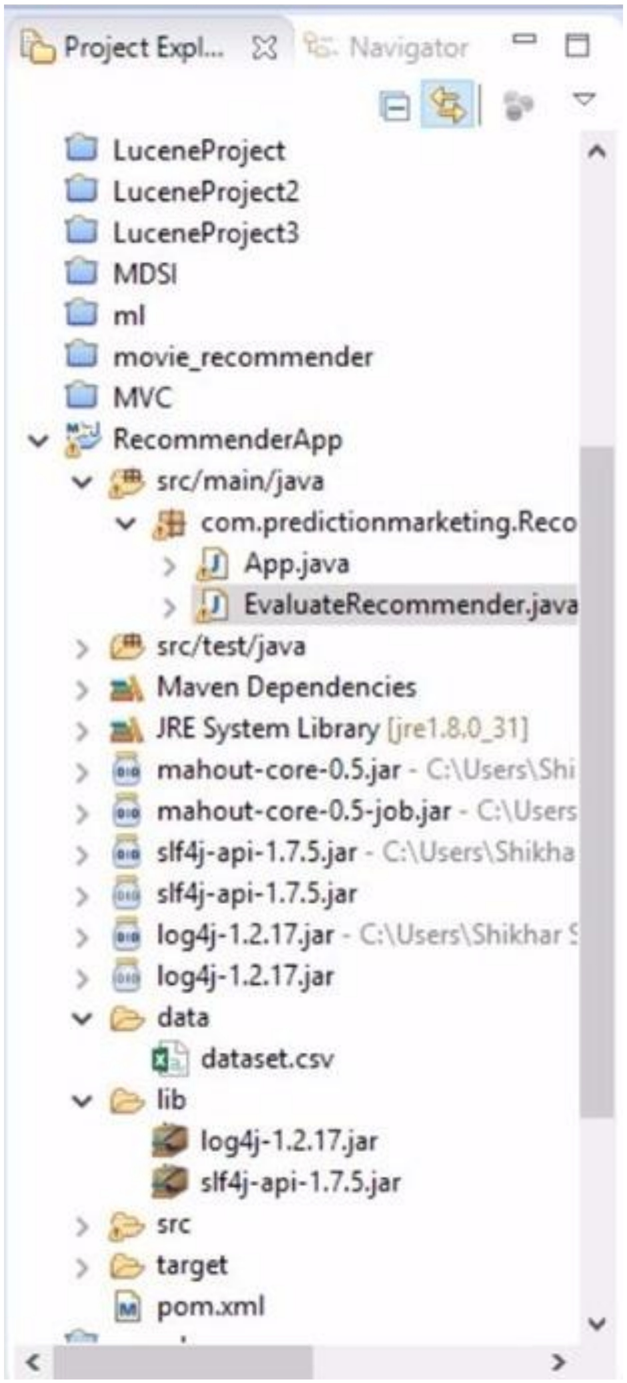


Fig. 6.

6. Creating the recommender based on user-based

In this project, we want to create a user-based recommender. The idea behind this approach is that when we want to compute recommendations for a particular users, we look for other users with a similar taste and pick the recommendations from their items:

Step 1 - Create a class called SampleRecommender with a main method.
The first thing we have to do is load the data from the file. Mahout's recommenders use an interface called DataModel to handle interaction data. You can load our made up interactions like this:

```
DataModel model = new FileDataModel(new
File("/path/to/dataset.csv"));
```

Step 2 - For finding similar users, we have to compare their interactions. There are various methods for doing this. One popular method is to compute the correlation coefficient between their interactions. In Mahout, you use this method as follows:

```
UserSimilarity similarity = new
PearsonCorrelationSimilarity(model);
```

Step 3 - The next thing we have to do is to define which similar users we want to leverage for the recommender. For the sake of simplicity, we'll use all that have a similarity greater than 0.1. This is implemented via a ThresholdUserNeighborhood

```
UserNeighborhood neighborhood = new
ThresholdUserNeighborhood(0.1, similarity, model);
```

Step 4 - Now we have all the pieces to create our recommender:

```
UserBasedRecommender recommender = new
GenericUserBasedRecommender(model, neighborhood,
similarity);
```

Step 5 - We can easily ask the recommender for recommendations now. If we wanted to get three items recommended for the user with userID 2, we would do it like this:

```
List recommendations = recommender.recommend(2, 3);
for (RecommendedItem recommendation :
recommendations)
 {
  System.out.println(recommendation);
}
```

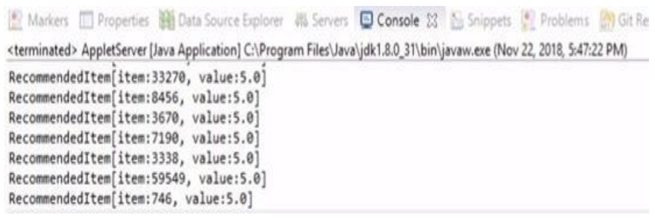
7. Prototype of Algorithm

7.1 Memory based techniques

The items that were already rated by the user before play a relevant role in searching for a neighbor that shares appreciation with him [14], [15]. Once a neighbor of a user is found, different algorithms can be used to combine the preferences of neighbors to generate recommendations. Due to the effectiveness of these techniques, they have achieved widespread success in real life applications. Memory-based CF can be achieved in two ways through user-based and item-based techniques. User based collaborative filtering technique calculates similarity between users by comparing their ratings on the same item, and it then computes the predicted rating for an item by the active user as a weighted average of the ratings of the item by users similar to the active user where weights are the similarities of these users with the target item. Item-based filtering techniques compute predictions using the similarity between items and not the similarity between users. It builds a model of item similarities by retrieving all items rated by an active user from the user-item matrix, it determines how similar the retrieved items are to the target item, then it selects the k most similar items and their corresponding similarities are also determined. Prediction is made by taking a weighted average of the active users rating on the similar items k. Several types of similarity measures are used to compute similarity between item/user. The two most popular similarity measures are correlation-based and cosine-based. Pearson correlation coefficient is used to measure the extent to which two constants linearly relate with each other and is defined as [16], [17]

8. Discussion

After we run the demonstration, we got the result that includes user ID, the movie id which were system recommended, and the predicted rating.



```
<terminated> AppletServer [Java Application] C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe (Nov 22, 2018, 5:47:22 PM)
RecommendedItem[item:33270, value:5.0]
RecommendedItem[item:8456, value:5.0]
RecommendedItem[item:3670, value:5.0]
RecommendedItem[item:7190, value:5.0]
RecommendedItem[item:3338, value:5.0]
RecommendedItem[item:59549, value:5.0]
RecommendedItem[item:746, value:5.0]
```

This result shows 7 recommended results for User 1. The item is the recommended movie ID and value is the predicted value of the rating. In other word, value is the degree of user preference estimated by the system.

In this system, user 6 requests three recommendations and gets three. Recommendation system engine recommends video 33270, video 8456, video 3670, video 7190, video 3338, video 59549 and video 746 to user 1. The reason for this is that the system calculates Pearson correlation index to obtain all users similar to user 1, and selects all users whose Pearson correlation index is higher than the set value to form a user group through filtering. By calculating the preference coefficient, the seven videos with the best corresponding preference index are selected from the user group. Then the system recommends them to user 1.

There are two reasons why the system does not choose other video for users. One is that the video is not viewed by users similar to user 1, and the other is that the preference value of user 1 calculated by the system is not listed in the top 7.

8.1 Evaluation

Then, we run the evaluation. The result of evaluation was used to judge the quality of our recommendation system.

The method that we use is “hold-out” test. In “hold-out” test, we divided the initial dataset into training dataset and test dataset. The training set is used for model training and the test set is used for performance evaluation. The test process is invisible, and it is impossible to directly see which part of the system is selected as test data and which part is used as training data.

The comparison methods include the calculation of average difference and the calculation of the sum of squares. In this project, we chose the average difference. Recommend products to test set users through training set simulation, calculate preference index, and then make the difference between the calculated recommendation index and the real rating of users in the original test set to obtain the overall average value.

The smaller the result is, the better the recommendation system will be.

To test our recommender, we created a class called EvaluateRecommender with a main method and added an class named MyRecommenderBuilder that implements the RecommenderBuilder interface. We implemented the buildRecommender method and made it setup our recommender.

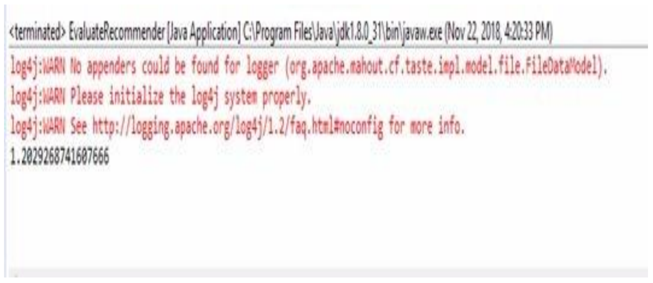
Then we created the code for the test. We were planning to check how much the recommender misses the real interaction strength on average. We employ an AverageAbsoluteDifferenceRecommenderEvaluator for this.

The following code shows the evaluation of our recommendation system:

```
UserSimilarity similarity = new
PearsonCorrelationSimilarity(dataModel);

UserNeighborhood neighborhood = new
ThresholdUserNeighborhood(0.1, similarity, dataModel);
return new GenericUserBasedRecommender(dataModel,
neighborhood, similarity);

DataModel model = new FileDataModel(new
File("/path/to/dataset.csv"));

RecommenderEvaluator evaluator = new
AverageAbsoluteDifferenceRecommenderEvaluator();

RecommenderBuilder builder = new
MyRecommenderBuilder();

double result = evaluator.evaluate(builder, null, model, 0.9,
1.0);
System.out.println(result);
```

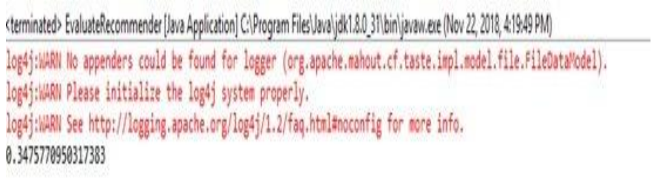
We run this test multiple times, and we got different results, because the splitting into trainingset and testset is done randomly.

For training dataset, we got the result.



```
<terminated> EvaluateRecommender [Java Application] C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe (Nov 22, 2018, 4:20:33 PM)
log4j:WARN No appenders could be found for logger (org.apache.mahout.cf.taste.impl.model.file.FileDataModel).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
1.2029268741607666
```

For actual dataset, we got the result:



```
<terminated> EvaluateRecommender [Java Application] C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe (Nov 22, 2018, 4:19:49 PM)
log4j:WARN No appenders could be found for logger (org.apache.mahout.cf.taste.impl.model.file.FileDataModel).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
0.3475770950317383
```

This means that second one is better than the first one. In the second evaluation, we used more data, which means our accuracy will be higher.
Results show that even if the calculated result is randomly, the actual data set of recommendation system always is better than the training dataset.

9. Conclusion

In this project, we implemented a recommendation system. In this recommendation system, we can recommend a video suitable for a particular user. We also predict users' preference for video recommendations. Achieved our initial goal of creating a video recommendation system. We have mastered the calculation method of Pearson correlation coefficient and applied the "hold out" test method to calculate evaluation.

Unfortunately, in this project, we had two failed attempts. One is a failed attempt at dataset and the other is a failed attempt at IBM cloud permission.

Initially, we have built a relatively complete video website. We can achieve some functions like login, user can upload, delete, edit video. Also, they can submit query and feedback. However, at the stage of uploading videos, since the IBM cloud need administrative account to upload videos, we did not have the permission for administrative account and so our videos cannot upload on our website. So, we failed to link the website with videos.

Although, we tried to upload some videos to check our website functionality from Amazon cloud, which allows us to upload videos which were less than 10mbs and in small number. We successfully upload it on website and our other functionality works but the number of videos is not enough to give us accurate recommendation.

As for the dataset, we tried to use more parameters datasets, such as more personalized recommendations based on user age, occupation and so on. Since we cannot get accurate outcomes with this kind of multiple parameters, we tried different datasets. Finally, we have reduced some parameters in existing dataset like age, occupation, gender which were little irrelevant and by removing these parameters, we evaluate outcomes which were much better than previous results.

10. Future work

In the near future, our recommender system can link to our existing website and by taking IBM cloud access of administrative side we can upload video dataset on our website. It will be installed in Apache Server and so it will be published in internet.

Datasets will be updated continuously, and it will make online actual rating predictions to the users whose habits are changing day by day. As a result, it can be sensitively satisfying current user tastes.

Web services in particular suffer from producing recommendations of millions of items to millions of users. The time and computational power can even limit the performance of the best recommending

systems. For larger dataset, we can work on scalability problems of recommendation systems.

11. References

[1] An overview of different approaches to Recommendation
https://ieeexplore-ieee-org.ledproxy2.uwindsor.ca/document/8276172

[2] Tapestry D. Goldberg, D. Nichols, B. M. Oki, and D. Terry
Using collaborative filtering to weave an information Communications of the ACM, vol. 35, no. 12, pp. 6170, 1992.

[3] A Study on Content-based Video Recommendation
https://ieeexplore-ieee-org.ledproxy2.uwindsor.ca/document/8297150

[4] Mladenic, D.:
Text-learning and Related Intelligent Agents: A Survey
IEEE Intelligent Systems14(4), 44–54 (1999).

[5] Search Based Video Recommendations
https://ieeexplore-ieee-org.ledproxy2.uwindsor.ca/document/6776190

[6] G. Adomavicius, A. Tuzhilin
Toward the next generation of recommender system. A survey of the state-of-the-art and possible extensions
IEEE Trans Knowl Data Eng, 17 (6) (2005), pp. 734-749
CrossRefView Record in Scopus Google Scholar

[7] Condiff MK, Lewis DD, Madigan D, Posse C. Bayesian mixed-effects models for recommender systems. In: Proceedings of ACM SIGIR workshop of recommender systems: algorithm and evaluation; 1999.
Google Scholar

[8] Oard DW, Kim J. Implicit feedback for recommender systems. In: Proceedings of 5th DELOS workshop on filtering and collaborative filtering; 1998. p. 31–6.
Google Scholar

[9] J. Buder, C. Schwind
Learning with personalized recommender systems: a psychological view
Comput Human Behav, 28 (1) (2012), pp. 207-216
Article Download PDF View Record in Scopus Google Scholar

[10] Gadanho SC, Lhuillier N. Addressing uncertainty in implicit preferences. In: Proceedings of the 2007 ACM conference on Recommender Systems (RecSys '07). ACM, New York, NY, USA; 2007. p. 97–104.
Google Scholar

[11] K. Shyong, D. Frankowski, J. Riedl
Do you trust your recommendations? An exploration of security and privacy issues in recommender systems
Emerging trends in information and communication security, Springer, Berlin, Heidelberg (2006), pp. 14-29
Google Scholar

[12] T. Zhang, S.I. Vijay
Recommender systems using linear classifiers
J Mach Learn Res, 2 (2002), pp. 313-334
View Record in Scopus Google Scholar

[13] D. Billsus, M.J. Pazzani
User modeling for adaptive news access
User Model User-adapted Interact, 10 (2–3) (2000), pp. 147-180
CrossRefView Record in Scopus Google Scholar

[14]Zhao ZD, Shang MS. User-based collaborative filtering recommendation algorithms on Hadoop. In: Proceedings of 3rd international conference on knowledge discovering and data mining, (WKDD 2010), IEEE Computer Society, Washington DC, USA; 2010. p. 478–81. doi: 10.1109/WKDD.2010.54.
Google Scholar

[15] Zhu X, Ye HW, Gong S. A personalized recommendation system combining case-based reasoning and user-based collaborative filtering. In: Control and decision conference (CCDC 2009), Chinese; 2009. p. 4026–28.
Google Scholar

[16] Melville P, Mooney-Raymond J, Nagarajan R. Content-boosted collaborative filtering for improved recommendation. In: Proceedings of the eighteenth national conference on artificial intelligence (AAAI), Edmonton, Canada; 2002. p. 187–92.
Google Scholar

[17] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich
Recommender systems – an introduction
Cambridge University Press (2010)
Google Scholar

[18] B. Mobasher, X. Jin, Y. Zhou
Semantically enhanced collaborative filtering on the web
Web mining: from web to semantic web, Springer, Berlin Heidelberg (2004), pp. 57-76
CrossRefView Record in Scopus Google Scholar

[19] H.C. Yoon, K.K. Jae, H.K. Soung
A personalized recommender system based on web usage mining and decision tree induction
Expert Syst Appl, 23 (2002), pp. 329-342
CrossRefView Record in Scopus Google Scholar

[20] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen
Collaborative filtering recommender systems
P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, LNCS 4321, Springer, Berlin Heidelberg (Germany) (2007), pp. 291-324, 10.1007/978-3-540-72079-9_9
CrossRefView Record in Scopus Google Scholar

[21] R. Burke
Web recommender systems
P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, LNCS 4321, Springer, Berlin Heidelberg (Germany) (2007), pp. 377-408, 10.1007/978-3-540-72079-9_12
CrossRefView Record in Scopus Google Scholar

[22] D.H. Park, H.K. Kim, I.Y. Choi, J.K. Kim
A literature review and classification of recommender systems research
Expert Syst Appl, 39 (11) (2012), pp. 10059-10072
Article Download PDF View Record in Scopus Google Scholar

[23] X. Su, T.M. Khoshgoftaar
A survey of collaborative filtering techniques
Adv Artif Intell, 4 (2009), p. 19
CrossRefView Record in Scopus Google Scholar

[24] U. Shardanand, P. Maes
Social information filtering: algorithms for automating "word of mouth"
Proceedings of the SIGCHI conference on human factors in computing systems, ACM Press/Addison-Wesley Publishing Co. (1995), pp. 210-217
CrossRefView Record in Scopus Google Scholar

[25] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, J. Riedl
Applying collaborative filtering to usenet news
Commun ACM, 40 (3) (1997), pp. 77-87
CrossRefView Record in Scopus Google Scholar

[26] T.Q. Lee, P. Young, P. Yong-Tae
A time-based approach to effective recommender systems using implicit feedback
Expert Syst Appl, 34 (4) (2008), pp. 3055-3062
Article Download PDF View Record in Scopus Google Scholar

[27] Q. Shambour, J. Lu
A trust-semantic fusion-based recommendation approach for e-business applications
Decis Support Syst, 54 (1) (2012), pp. 768-780
Article Download PDF View Record in Scopus Google Scholar

[28] O'Donovan J, Smyth B. Trust in recommender systems. In: Proceedings of the 10th international conference on intelligent user interfaces, ACM; 2005. p. 167–74.
Google Scholar

[29] G. Adomavicius, J. Zhang
Impact of data characteristics on recommender systems performance
ACM Trans Manage Inform Syst, 3 (1) (2012)
Google Scholar

[30] Stern DH, Herbrich R, Graepel T. Matchbox: large scale online bayesian recommendations. In: Proceedings of the 18th international conference on World Wide Web. ACM, New York, NY, USA; 2009. p. 111–20.
Google Scholar

[31] Claypool M, Gokhale A, Miranda T, Murnikov P, Netes D, Sartin M. Combining content- based and collaborative filters in an online newspaper. In: Proceedings of ACM SIGIR workshop on recommender systems: algorithms and evaluation, Berkeley, California; 1999.
Google Scholar

[32] Billsus D, Pazzani MJ. A hybrid user model for news story classification. In: Kay J, editor. In: Proceedings of the seventh international conference on user modeling, Banff, Canada. Springer-Verlag, New York; 1999. p. 99–108.

Google Scholar

[33] B. Smyth, P. Cotter
A personalized TV listings service for the digital TV age
J Knowl-Based Syst, 13 (2–3) (2000), pp. 53-59
Article Download PDF View Record in Scopus Google Scholar