**Recommender Systems**

Prepared By

Gopal Das

Registration No: 171010120007 of 2017-2018

Under the guidance of

Dr. Dipak Kumar Kole

Associate professor

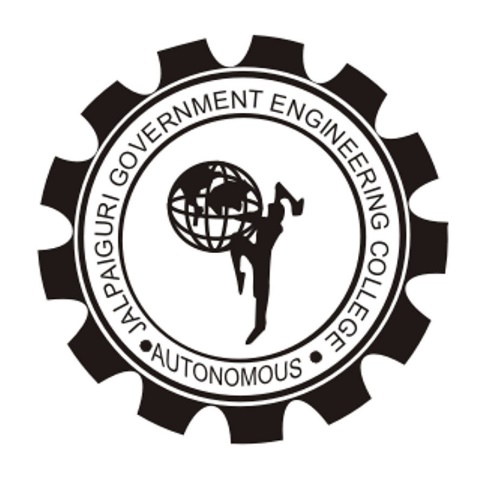
Department of Computer Science & Engineering

Jalpaiguri Government Engineering College

Project Report

Submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Computer Science & Engineering

Jalpaiguri Government Engineering College

****

Jalpaiguri Government Engineering College

**CERTIFICATION**

Jalpaiguri Government Engineering College

Faculty of Computer Science & Engineering

This is to certify that the work in preparing the project entitled “ Recommendation System using Machine Learning” has been carried out by Gopal Das under my guidance during the session 2019-2020 for the degree of Bachelor of Technology in Computer Science & Engineering.

……………….................................. ....……………………............................…

Prof. Dhiman Mondal Dr. Dipak Kumar Kole

Head of Department Supervisor

Computer Science & Engineering Computer Science & Engineering

Jalpaiguri Government Engineering College Jalpaiguri Government Engineering College

**Preface**

The project work summarized in this report, This project deals with the development of a machine learning system to devlop A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. Various tecniques have been used to meet the requirements. We hereby thank our project supervisor Dr. Dipak Kumar Kole. A word of thanks also goes to all our friends for being our best critics. And finally, this documentation would never have been more educative and efficient without the constant help and guidance of our project guide Dr. Dipak Kumar Kole. We would like to thank him for giving us the right guidance and encouraging us to complete the project within time. We also express our deepest and sincere gratitude to all our teachers for their kind comments and advice for our project. We would also like to express our heartiest gratitude to Prof. Amitava Roy (Principal), Prof. Dhiman Mondal(Head of the Department of Computer Science & Engineering).

……………………………………………..

Gopal Das

**Contents**

**1 Introduction**

* 1. Recommender system………………....1
  2. Application………………………………………………….2
  3. Advantages of recommender system………….3
  4. Types of recommender system……………………4
  5. Implementing recommender systems………….5

1. **Content-based recommender systems**
   1. What are content-based recommender systems………………………………………………………..6
   2. Calculating weighing the genres……………………8
   3. Candidate movies for recommendation………..9
   4. Finding the recommendation………………………10
2. **Collaborative filtering**
   1. User-based collaborative filtering……………….12
   2. Creating user ratings matrix………………………..13
   3. Why pearson correlation?.............................14
   4. Learning the similarity weights……………………15
   5. Creating the weighted rating matrix……………16
   6. Challenges of collaborative filtering…………….18
3. **Conclusion and Future Scope**……………………………………….20
4. References…………………………………………………………………..24

**LIST OF FIGURES**

1. Types of recommender system …………………………………. 3
2. Content-based recommender system …………………………………………… 5
3. Finding which is best for user …………………………………………………........ 6
4. Figure out genres of movies ……………………………………………….. 7
5. Weighing the genres for user ……………………………………………….. 7
6. Figure out each movies genres that we are going to recommendation for user ………………………………………………………………………………. 8
7. Finding the recommendation matrix ……………………………………….. 9
8. Figure out probability of all movies that we are recommender for user …10
9. User-based collaborative filtering ………………………………………….. 11
10. User ratings matrix …………………………………………………………. 12
11. Finding similarity weights based on user profile ……………………….. 14
12. Creating weighted rating matrix …………………………………………. 15
13. Finding recommendation matrix …………………………………………. 15

14 Showing user-based and item-based filtering ………………………….. 16

**Introduction**

* Recommender systems are used to provide personalized recommendations according to user profile and previous behaviour. Recommender systems are widely used in the internet Industry. Services like Amazon, Netflix and YouTube are typical examples of recommender system users. Recommender systems cannot only help the users find their favourite products, but also bring potential profit to online service providers.

**Applications**

* What to buy?
  + E-commerce , books , movies , beer, shoes
* What to eat?
* Which Job to apply to?
* Who you should be friends with?
  + Linkedin , Facebook , …….
* Personalize your experience on the web
  + News platforms , news personalization

Recommender systems are usually at play on many websites. For example, suggesting books on Amazon and movies on Netflix. In fact, everything on Netflix’s website is driven by customer selection. If a certain movie gets viewed frequently enough, Netflix’s recommender system ensures that that movie gets an increasing number of recommendations. Another example can be found in a daily-use mobile app, where a recommender engine is used to recommend anything from where to eat, or, what job to apply to. On social media, sites like Facebook or LinkedIn, regularly recommend friendships. Recommender systems are even used to personalize your experience on the web.

For example, when you go to a news platform website, a recommender system will make note of the types of stories that you clicked on and make recommendations on which types of stories you might be interested in reading, in future. There are many of these types of examples and they are growing in number every day.

**Advantages of recommender system**

* Broader exposure
* Possibility of continual usages or purchase of products
* Provides better experience

One of the main advantages of using recommendation systems is that users get a broader exposure to many different products they might be interested in.This exposure encourages users towards continual usage or purchase of their product .Not only does this provide a better experience for the user but it benefits the service provider, as well, with increased potential revenue and better security for its customers.

**Two types of recommender system**

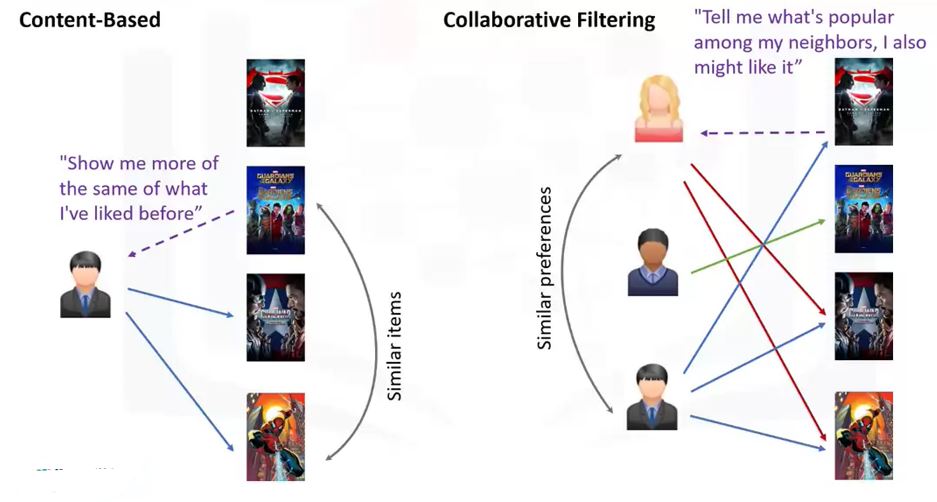
****

Fig1: Types of recommender system

There are generally 2 main types of recommendation systems: Content-based and collaborative filtering.

The main difference between each, can be summed up by the type of statement that a consumer

might make. For instance, the main paradigm of a Content-based recommendation system is driven by the statement: “Show me more of the same of what I've liked before." Content-based systems try to figure out what a user's favorite aspects of an item are, and then make recommendations on items that

share those aspects. Collaborative filtering is based on a user saying, “Tell me what's popular among my

neighbors because I might like it too.” Collaborative filtering techniques find similar groups of users, and provide recommendations based on similar tastes within that group. In short, it assumes that a user might be interested in what similar users are interested in.

Also, there are Hybrid recommender systems, which combine various mechanisms.

**Implementing recommender systems**

* Memory-based
  + Uses the entire user-item dataset to generate a recommendation .
  + Uses statistical techniques to approximate users or items e.g., Pearson Correlation , Correlation , Cosine Similarity , Euclidean Distance , etc.
* Model-based
  + Develops a model of users in an attempt to learn their preferences.
  + Models can be created using Machine Learning techniques like regression , clustering , classification , etc.

**Content-based recommender systems**

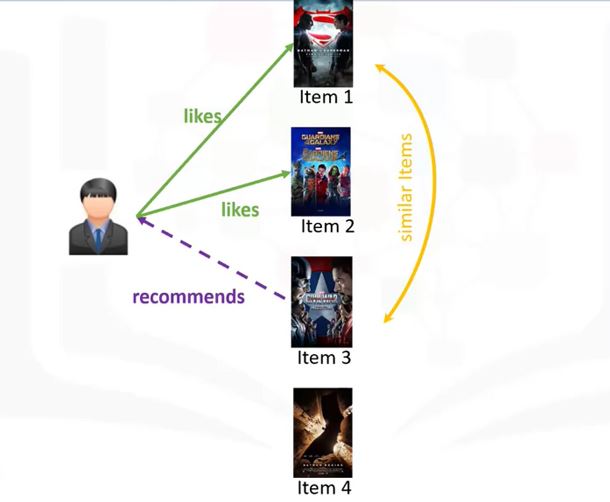
****

Fig2: Content-based recommender system

A Content-based recommendation system tries

to recommend items to users based on their profile.

The user's profile revolves around that user's preferences and tastes. It is shaped based on user ratings,including the number of times that user has clicked on different items or perhaps even liked those items. The recommendation process is based on the similarity between those items. Similarity or closeness of items is measured based on the similarity in the content of those items. When we say content, we're talking about things like the items category, tag, genre, and so on. For example, if we have four movies, and if the user likes or rates the first two items, and if Item 3 is similar to Item 1 in terms of their genre, the engine will also recommend Item 3 to the user. In essence, this is what content-based recommender system engines do. Now, let's dive into a content-based recommender system to see how it works.



Fig3: Finding which is best for user

Let's assume we have a data set of only six movies.

This data set shows movies that our user has

watched and also the genre of each of the movies.

For example, Batman versus Superman is in the Adventure, Super Hero genre and Guardians of the Galaxy is in the Comedy, Adventure, Super Hero and Science-fiction genres. Let's say the user has watched and rated three movies so far and she has given a rating of two out of 10 to the first movie, 10 out of 10 to the second movie and eight out of 10 to the third. The task of the recommender engine is to recommend one of

the three candidate movies to this user, or in other,

words we want to predict what the user's possible rating would be of the three candidate movies if she were to watch them. To achieve this, we have to build the user profile.

**Weighing the genres**

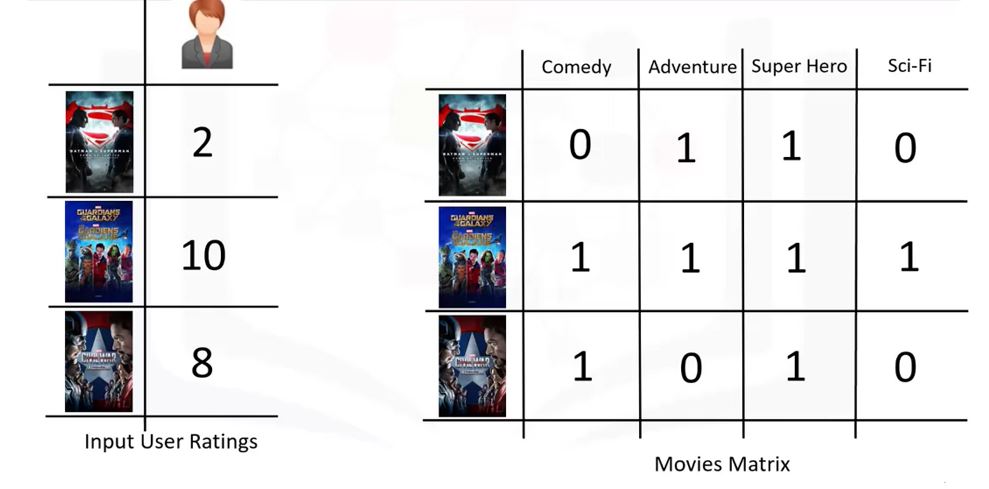
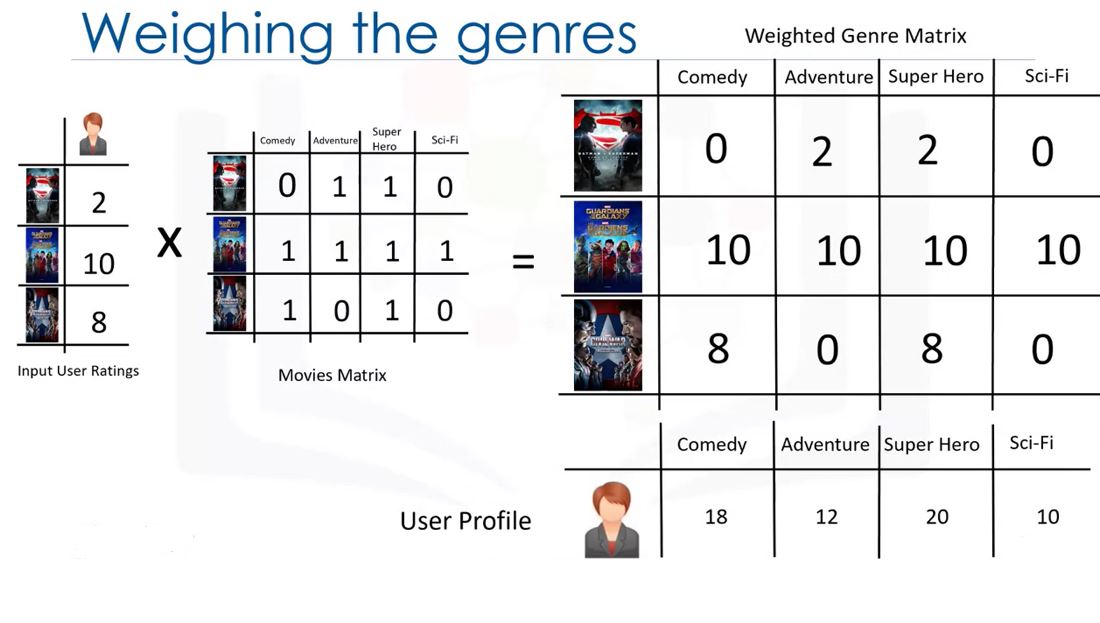


Fig4: Figure out genres of movies



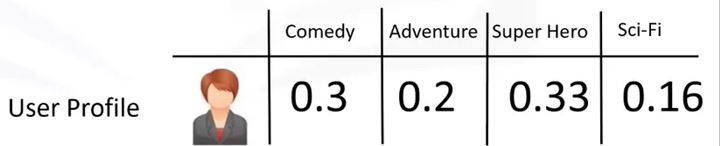


Fig5: Weighing the genres for user

First, we create a vector to show the user's ratings for the movies that she's already watched. We call it Input User Ratings. Then, we encode the movies through the one-hot encoding approach. Genre of movies are used here as a feature set. We use the first three movies to make this matrix, which represents the movie feature set matrix If we multiply these two matrices we can get the weighted feature set for the movies. Let's take a look at the result. This matrix is also called the Weighted Genre matrix and represents the interests of the user for each genre based on the movies that she's watched.

Now, given the Weighted Genre Matrix, we can shape the profile of our active user. Essentially, we can aggregate the weighted genres and then normalize them to find the user profile. It clearly indicates that she likes superhero movies more than other genres. We use this profile to figure out what movie is proper to recommend to this user.

**Candidate movies for recommendation**

****

Fig6 : Figure out each movies genres that we are going to recommendation for user

**Finding the recommendation**

****

Fig7 : Finding the recommendation matrix

Recall that we also had three candidate movies for

recommendation that haven't been watched by the user, we encode these movies as well.

Now we're in the position where we have to figure out

which of them is most suited to be recommended to the user. To do this, we simply multiply the User Profile matrix by the candidate Movie Matrix, which results in the Weighted Movies Matrix. It shows the weight of each genre with respect to the User Profile. Now, if we aggregate these weighted ratings, we get the active user's possible interest level in these three movies.

In essence, it's our recommendation lists, which we can sort to rank the movies and recommend them to the user. For example, we can say that the Hitchhiker's Guide to the Galaxy has the highest score in our list and it's proper to recommend to the user.



Fig8 : Figure out probability of all movies that we are recommender for user

Now, you can come back and fill the predicted ratings for the user. So, to recap what we've discussed so far,

the recommendation in a content-based system is based on user's taste and the content or feature set items.

Such a model is very efficient. However, in some cases, it doesn't work. For example, assume that we have a movie in the drama genre,which the user has never watch. So, this genre would not be in her profile.

Therefore, shall only get recommendations related to genres that are already in her profile and the recommender engine may never recommend any movie within other genres. This problem can be solved by other types of recommender systems such as collaborative filtering.

**Collaborative filtering**

* User-based collaborative filtering
  + Based on users’ neighbourhood
* Item-based collaborative filtering
  + Based on items’ similarity

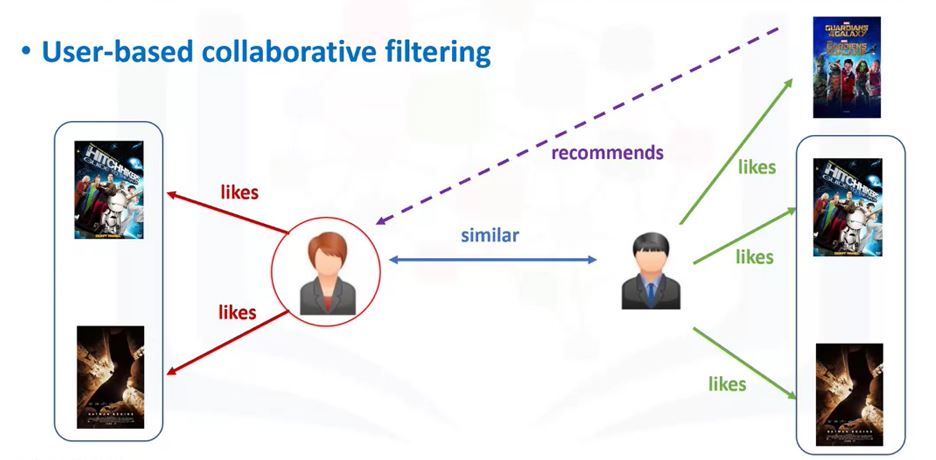


Fig9 : User-based collaborative filtering

Collaborative filtering is based on the fact that relationships exist between products and people's interests. Many recommendation systems use collaborative filtering to find these relationships and to give an accurate recommendation

of a product that the user might like or be interested

in. Collaborative filtering has basically two approaches; user-based, and item-based. User-based collaborative filtering is based on the user similarity or neighborhood. Item-based collaborative filtering is based on similarity among items. Let's first look at the intuition behind the user-based approach. In user-based collaborative filtering,we have an active user for whom the recommendation is aimed. The collaborative filtering engine first looks for users who are similar. That is users who share the active users rating patterns.Collaborative filtering basis this similarity on things like history, preference, and choices that users make when buying, watching, or enjoying something. For example, movies that similar users have rated highly. Then it uses the ratings from these similar users to predict the possible ratings by the active user for a movie that she had not previously watched. For instance, if two users are similar or are neighbors in terms of their interested movies,

we can recommend a movie to the active user that her neighbor has already seen.

**User ratings matrix**

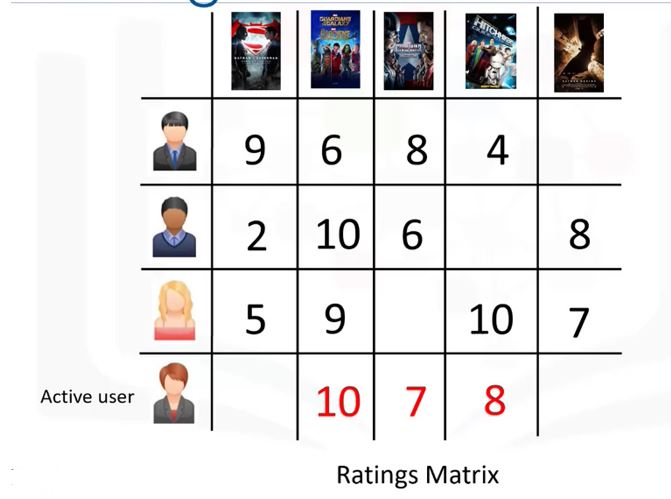


Fig10 : User ratings matrix

Assume that we have a simple user item matrix, which shows the ratings of four users for five different movies. Let's also assume that our active user has watched and rated three out of these five movies.

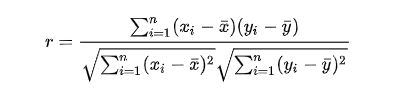
Let's find out which of the two movies that our active user hasn't watched should be recommended to her.

The first step is to discover how similar

the active user is to the other users.

**Why Pearson Correlation?**

Pearson correlation is invariant to scaling, i.e. multiplying all elements by a nonzero constant or adding any constant to all elements. For example, if you have two vectors X and Y,then, pearson(X, Y) == pearson(X, 2 \* Y + 3). This is a pretty important property in recommendation systems because for example two users might rate two series of items totally different in terms of absolute rates, but they would be similar users (i.e. with similar ideas) with similar rates in various scales.



The values given by the formula vary from r = -1 to r = 1, where 1 forms a direct correlation between the two entities (it means a perfect positive correlation) and -1 forms a perfect negative correlation.

In our case, a 1 means that the two users have similar tastes while a -1 means the opposite.

**Learning the similarity weights**

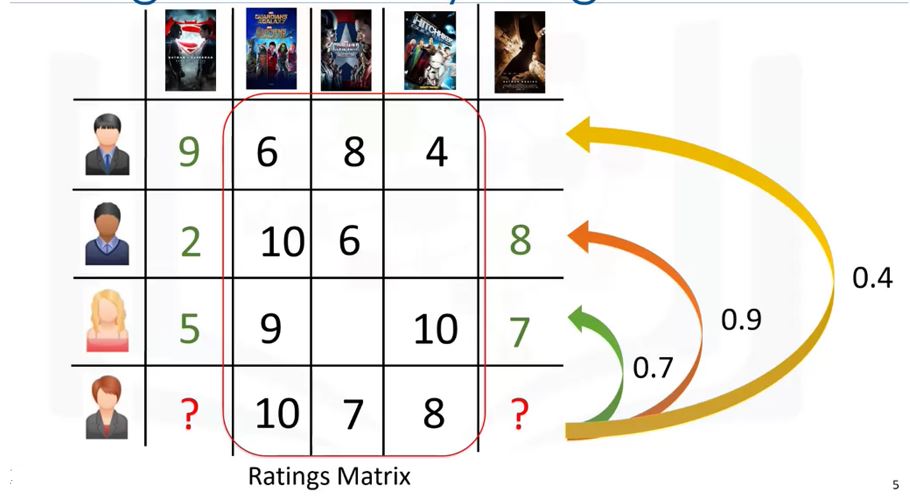


Fig11: Finding similarity weights based on user profile

How do we do this? Well, this can be done through several different statistical and vectorial techniques such as distance or similarity measurements including Euclidean Distance, Pearson Correlation, Cosine Similarity, and so on. To calculate the level of similarity between two users, we use the three movies that both the users have rated in the past. Regardless of what we use for similarity measurement, let's say for example, the similarity could be 0.7, 0.9, and 0.4 between the active user and other users. These numbers represent similarity weights or proximity of the active user to other users in the dataset.

**Creating the weighted ratings matrix**

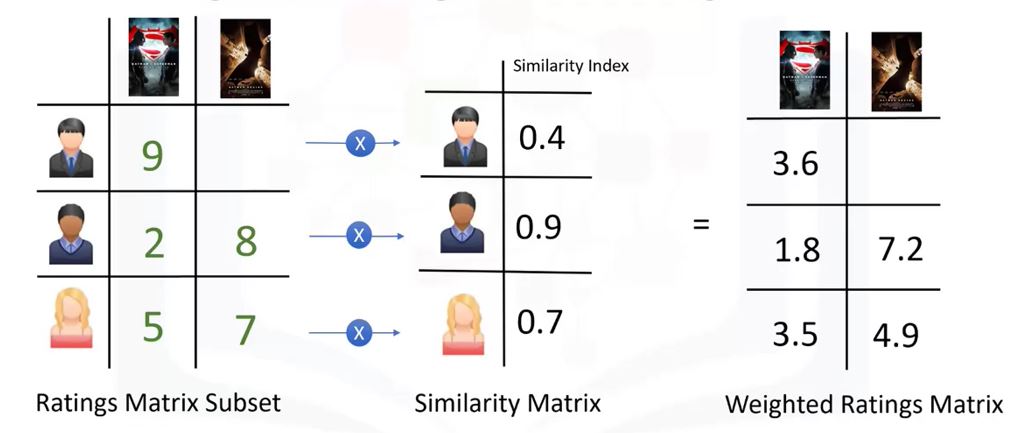


Fig12: Creating weighted rating matrix

The next step is to create a weighted rating matrix. We just calculated the similarity of users to our active user in the previous slide. Now, we can use it to calculate the possible opinion of the active user about our two target movies. This is achieved by multiplying the similarity weights to the user ratings. It results in a weighted ratings matrix, which represents the user's neighbors opinion about are two candidate movies for recommendation. In fact, it incorporates the behavior of other users and gives more weight to the ratings of those users who are more similar to the active user.

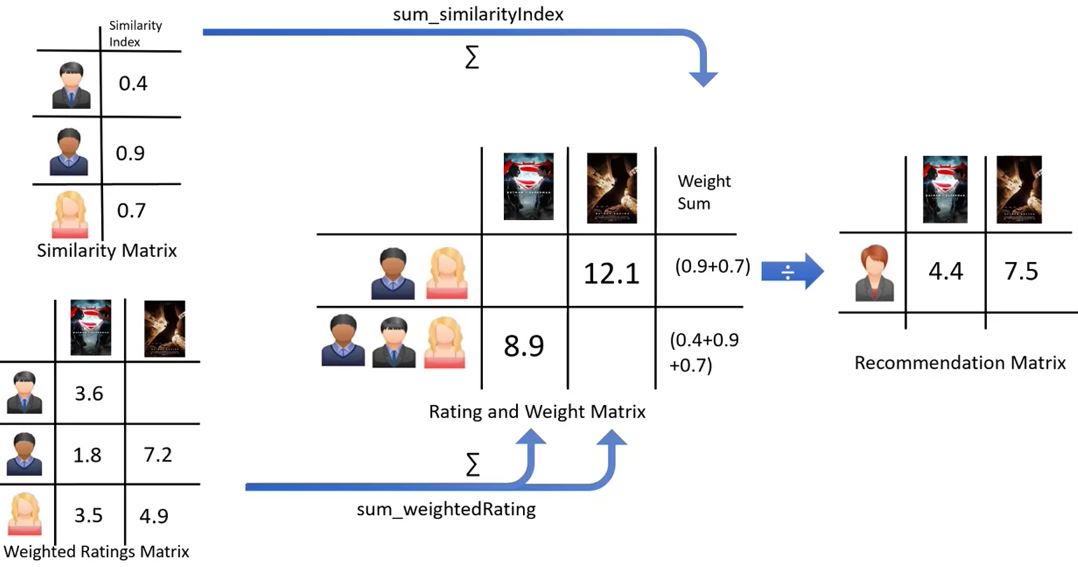


Fig13 : Finding recommendation matrix

Now, we can generate the recommendation matrix by aggregating all of the weighted rates. However, as three users rated the first potential movie and two users rated the second movie, we have to normalize the weighted rating values. We do this by dividing it by the sum of the similarity index for users. The result is the potential rating that our active user will give to these movies based on her similarity to other users. It is obvious that we can use it to rank the movies for providing recommendation to our active user.

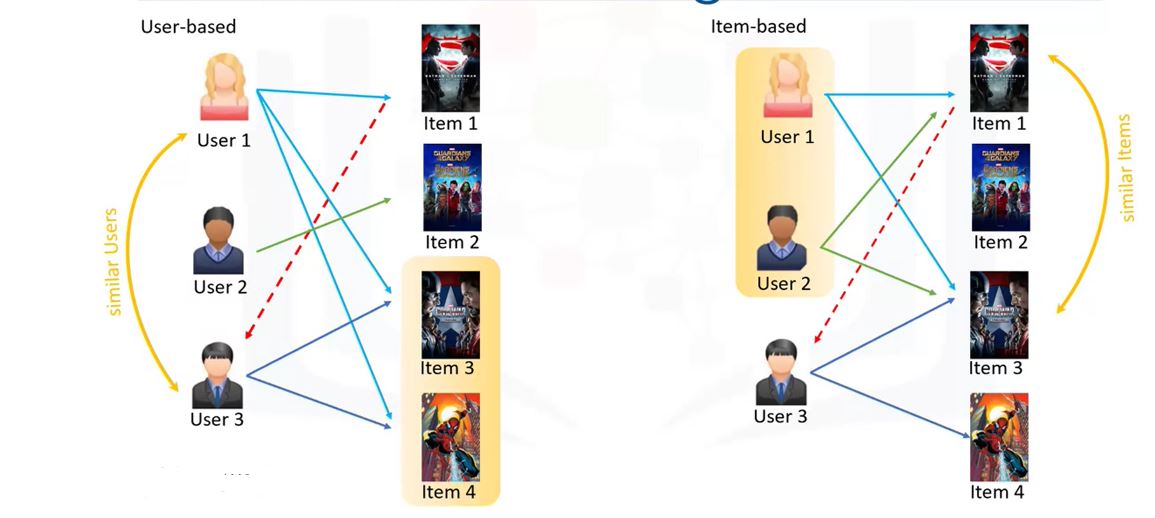


Fig14: Showing user-based and item-based filtering

Now, let's examine what's different between user-based and item-based collaborative filtering. In the user-based approach, the recommendation is based on users of

the same neighborhood with whom he or she shares common preferences. For example, as User 1 and User 3 both liked Item 3 and Item 4, we consider them as similar or neighbor users, and recommend Item 1 which is positively rated by User 1 to User 3. In the item-based approach, similar items build neighborhoods on the behavior of users. Please note however, that it is not based on their contents. For example, Item 1 and Item 3 are considered neighbors as they were positively rated by both User 1 and User 2. So, Item 1 can be recommended to User 3 as he has already shown interest in Item 3. Therefore, the recommendations here are based on the items in the neighborhood that a user might prefer.

**Challenges of collaborative filtering**

* Data Sparsity
  + Users in general rate only a limited number of items
* Cold Start
  + Difficulty in recommendation to new users or new items
* Scalability
  + Increase in number of users or items

Collaborative filtering is a very effective recommendation system. However, there are some challenges with it as well. One of them is data sparsity. Data sparsity happens when you have a large data set of users who generally rate only a limited number of items. As mentioned, collaborative based recommenders can only

predict scoring of an item if there are other users who have rated it. Due to sparsity, we might not have enough ratings in the user item dataset which makes it impossible to provide proper recommendations. Another issue to keep in mind is something called cold start. Cold start refers to the difficulty the recommendation system has when there is a new user,

and as such a profile doesn't exist for them yet. Cold start can also happen when we have a new item which has not received a rating. Scalability can become an issue as well. As the number of users or items increases and the amount of data expands, collaborative filtering algorithms will begin to suffer drops in performance, simply due to growth and the similarity computation.

**Conclusion and Future Scope**

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of user data in existing corporate databases, and will be stressed even more by the increasing volume of user data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems.

In this paper we presented and experimentally evaluated a new algorithm for CF-based recommender systems. Our results show that item-based techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high-quality recommendations.

**FUTURE WORK:**

Some decisions are to be made to overcome the limitations. Some of the future improvements to be done are:

**Multi-criteria recommender systems**

Multi-criteria recommender systems (MCRS) can be defined as recommender systems that incorporate preference information upon multiple criteria. Instead of developing recommendation techniques based on a single criterion values, the overall preference of user u for the item i, these systems try to predict a rating for unexplored items of u by exploiting preference information on multiple criteria that affect this overall preference value. Several researchers approach MCRS as a multi-criteria decision making (MCDM) problem, and apply MCDM methods and techniques to implement MCRS systems.

### Risk-aware recommender systems

### The majority of existing approaches to recommender systems focus on recommending the most relevant content to users using contextual information, yet do not take into account the risk of disturbing the user with unwanted notifications. It is important to consider the risk of upsetting the user by pushing recommendations in certain circumstances, for instance, during a professional meeting, early morning, or late at night. Therefore, the performance of the recommender system depends in part on the degree to which it has incorporated the risk into the recommendation process. One option to manage this issue is *DRARS*, a system which models the context-aware recommendation as a bandit algorithm. This system combines a content-based technique and a contextual bandit algorithm.

### Mobile recommender systems

Mobile recommender systems make use of internet-accessing smart phones to offer personalized, context-sensitive recommendations. This is a particularly difficult area of research as mobile data is more complex than data that recommender systems often have to deal with. It is heterogeneous, noisy, requires spatial and temporal auto-correlation, and has validation and generality problems.

There are three factors that could affect the mobile recommender systems and the accuracy of prediction results: the context, the recommendation method and privacy. Additionally, mobile recommender systems suffer from a transplantation problem – recommendations may not apply in all regions (for instance, it would be unwise to recommend a recipe in an area where all of the ingredients may not be available).

One example of a mobile recommender system are the approaches taken by companies such as Uber and Lyft to generate driving routes for taxi drivers in a city. This system uses GPS data of the routes that taxi drivers take while working, which includes location (latitude and longitude), time stamps, and operational status (with or without passengers). It uses this data to recommend a list of pickup points along a route, with the goal of optimizing occupancy times and profits.

Mobile recommendation systems have also been successfully built using the "Web of Data" as a source for structured information. A good example of such system is SMARTMUSEUM The system uses semantic modelling, information retrieval, and machine learning techniques in order to recommend content matching user interests, even when presented with sparse or minimal user data.

### Hybrid recommender systems

Most recommender systems now use a hybrid approach, combining collaborative filtering, content-based filtering, and other approaches . There is no reason why several different techniques of the same type could not be hybridized. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model (see for a complete review of recommender systems). Several studies that empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrated that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem, as well as the knowledge engineering bottleneck in knowledge-based approaches.

Netflix is a good example of the use of hybrid recommender systems. The website makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).

Some hybridization techniques include:

* **Weighted**: Combining the score of different recommendation components numerically.
* **Switching**: Choosing among recommendation components and applying the selected one.
* **Mixed**: Recommendations from different recommenders are presented together to give the recommendation.
* **Feature Combination**: Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
* **Feature Augmentation**: Computing a feature or set of features, which is then part of the input to the next technique.
* **Cascade**: Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
* **Meta-level**: One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

**References**

1. Francesco Ricci and Lior Rokach and Bracha Shapira, [Introduction to Recommender Systems Handbook](http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf), Recommender Systems Handbook, Springer, 2011, pp. 1-35
2. [*"Facebook, Pandora Lead Rise of Recommendation Engines - TIME"*](http://www.time.com/time/magazine/article/0,9171,1992403,00.html). TIME.com. 27 May 2010*. Retrieved 1 June 2015*.
3. Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh [WTF:The who-to-follow system at Twitter](http://dl.acm.org/citation.cfm?id=2488433), Proceedings of the 22nd international conference on World Wide Web
4. H. Chen, A. G. Ororbia II, C. L. Giles [ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries](https://arxiv.org/pdf/1511.02058.pdf), in arXiv preprint 2015
5. H. Chen, L. Gou, X. Zhang, C. Giles [Collabseer: a search engine for collaboration discovery](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.8659&rep=rep1&type=pdf), in ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2011
6. Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, [The VITA Financial Services Sales Support Environment](http://www.aaai.org/Papers/AAAI/2007/AAAI07-274.pdf), in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007.
7. Hosein Jafarkarimi; A.T.H. Sim and R. Saadatdoost [A Naïve Recommendation Model for Large Databases](http://www.ijiet.org/show-31-222-1.html), International Journal of Information and Education Technology, June 2012
8. Prem Melville and Vikas Sindhwani, [Recommender Systems](http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf), Encyclopedia of Machine Learning, 2010.
9. R. J. Mooney & L. Roy (1999). *Content-based book recommendation using learning for text categorization*. In Workshop Recom. Sys.: Algo. and Evaluation.
10. Rubens, Neil; [Elahi](https://www.linkedin.com/in/mehdielahi), Mehdi; Sugiyama, Masashi; Kaplan, Dain (2016). ["Active Learning in Recommender Systems"](https://rd.springer.com/book/10.1007/978-1-4899-7637-6). In Ricci, Francesco; Rokach, Lior; Shapira, Bracha (eds.). *Recommender Systems Handbook* (2 ed.). Springer US. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1007/978-1-4899-7637-6\_24](https://doi.org/10.1007%2F978-1-4899-7637-6_24). [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-1-4899-7637-6](https://en.wikipedia.org/wiki/Special:BookSources/978-1-4899-7637-6).
11. [Elahi](https://www.linkedin.com/in/mehdielahi), Mehdi; [Ricci](http://www.inf.unibz.it/~ricci/), Francesco; Rubens, Neil (2016). ["A survey of active learning in collaborative filtering recommender systems"](https://www.researchgate.net/publication/303781992). *Computer Science Review*. **20**: 29–50. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/j.cosrev.2016.05.002](https://doi.org/10.1016%2Fj.cosrev.2016.05.002)
12. Andrew I. Schein, Alexandrin Popescul, [Lyle H. Ungar](https://en.wikipedia.org/wiki/Lyle_Ungar), David M. Pennock (2002). [*Methods and Metrics for Cold-Start Recommendations*](https://archive.org/details/sigir2002proceed0000inte). Proceedings of the 25th Annual International [ACM](https://en.wikipedia.org/wiki/Association_for_Computing_Machinery) [SIGIR](https://en.wikipedia.org/wiki/Special_Interest_Group_on_Information_Retrieval) Conference on Research and Development in Information Retrieval (SIGIR 2002). [New York City, New York](https://en.wikipedia.org/wiki/New_York_City): [ACM](https://en.wikipedia.org/wiki/Association_for_Computing_Machinery). pp. 253–260. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [1-58113-561-0](https://en.wikipedia.org/wiki/Special:BookSources/1-58113-561-0). Retrieved 2008-02-02.
13. Karlgren, Jussi. 1990. "An Algebra for Recommendations." Syslab Working Paper 179 (1990).
14. Karlgren, Jussi. "[Newsgroup Clustering Based On User Behavior-A Recommendation Algebra](http://soda.swedish-ict.se/2225/2/T94_04.pdf)." SICS Research Report (1994).
15. Karlgren, Jussi (October 2017). ["A digital bookshelf: original work on recommender systems"](https://jussikarlgren.wordpress.com/2017/10/01/a-digital-bookshelf-original-work-on-recommender-systems/). Retrieved 27 October 2017.