Abstract

This System has been developed to recommend movies to its users. The act of recommendation is not contingent. It uses certain algorithms and performs some calculations. Based on these calculations, the system recommends user with the list of relevant movies. The system first gathers the full information of user that includes movies he/she watched along with the ratings. Then the system uses K-mean clustering techniques to find the group (cluster) he/she may belong to. Each cluster contains group of similar users. Finally he/she gets recommended from all other users from the same group and this is achieved by well known techniques called Collaborative Filtering.

Table of Contents

List of Figures

Abbreviation

CF	Collaborative Filtering
CUI	Character User Interface
GUI	Graphic User Interface
IR	Information Retrieval
tf-idf	Term frequency – Inverse Document frequency

1. INTRODUCTION

It is often necessary to make choices without sufficient personal experience of the alternatives. In everyday life, we rely on recommendations from other people either by word of mouth, recommendation letters, movie and book reviews printed in newspapers. Recommender System assist and augment this natural social process. In a typical recommender system people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients.
The developer of the first recommender system, Tapestry [1], coined the phrase "collaborative filtering" and several others have adopted it. We prefer the more general term "recommender system" for two reasons. First, recommenders may not explicitly collaborate with recipients, who may be unknown to each other. Second, recommendations may suggest particularly interesting items, in addition to indicating those that should be filtered out.
Recommendation is a particular form of information filtering, that explits past behaviors and user similarities to generate a list of information items that is personally tailored to an end-user's preferences.

In simple word, recommender system recommends items. It is a facility that is designed to predict the 'rating' or 'preferences' that user would give to an item (such as books, music, apps or movies) or social element (e.g. people or groups) they had not yet considered, in order to recommend relevant options that the user may like. To be more specific, recommender system or recommendation systems (sometimes replacing "system" \ with a synonym such as platform or engine) are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item.

Recommender systems have become extremely common in recent years and are applied in a variety of applications. Recommender systems changed the way inanimate websites communicate with their users. Rather than providing a static experience in which users search for and potentially buy products, recommender systems identify recommendations autonomously for individual users based on past purchases and searches, and on other user's behaviour.

This project, therefore, is mainly a recommender system built to solve the searching problem by recommending its user with the list of movies he/she may like. There are number of approaches that are used in solving the recommendation problem, among them we chose K-mean clustering and Collaborative Filtering. K mean clustering aims to partition n observations into K clusters in which each observation belongs to the cluster with nearest mean. It is an unsupervised learning and models the cluster of users on the basis of their interest. Collaborative filtering is the method of making automatic predictions about interest of user by collecting preferences or taste information from many user. Underlying assumption of Collaborative Filtering is that if person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on different issue X than to have opinion on X of a person chosen randomly.

1.1 Problem Statement

Searching the relevant items manually is the tedious task. It is time consuming and also the result obtained may not be reliable enough. It would be always better if someone (whom you trust) suggest you to watch the movies that you may like rather than you search yourself for the movies you like. Recommender System try to solve this problem by recommending the items (movies in our case) based on the user's interest and past experience.

1.2 Objective

1.2.1 Solve/Ease the searching problem:

The main objective of this project is to build the alternative solution to the searching problem. It is always the better option for the user if interesting items are recommended rather than searching. Most of the e-commerce website (like Amazon, eBay, and others) has adopted recommendation system thus it eases users to search preferable items.

1.2.2 To guide users to useful and interesting items (e.g. movies):

Recommendation system suggests users various products and items according to their previous search hence it guides them to preferable items. This eases users to search items they may like.

1.3 Related Work

1. Sites like Amazon and Netflix uses information about the things people buy or rent to determine which people or items are similar to one another, and then make recommendations based on purchase history [5, 6].

2. Youtube creates a list of recommended videos on the basis of user's past experiences and preferences.

1.4 Limitation

The purpose of this project is to recommend the list of movies to the user. This project focus on implementing the good recommendation approaches for better recommendation. The UI for this project is very simple and lacks GUI. So, considering this as a limitation for our project, we will build a full GUI recommender app in the near future.

1.5 Application

Recommender System is much needed but yet ignored in context of online stores here in Nepal. Recommender System is an internal system which is not visible to the user but is very important for them.

User Friendliness is the most important factor for an online form. Recommendation system analyses the user data's and finds suitable product that the user might like to buy. This not only increase the search efficiency for user but can also be helpful to increase sales.

International online shops such as ebay and amazon uses recommendation systems but this is yet to be applied in context of Nepal. With increasing use of internet and new technologies, online shopping is starting to become popular here as well. But still the online shopping form are far from user friendliness. Yes, there are search engines but the same shopping would be so easy if recommendation system is used.

Our system thus can be applied for the online shops and can reduce the overload of search problem. It not only helps to improve the user interaction but also help to improve the revenue for company.

2. LITERATURE REVIEW

2.1 Recommender System

Recommender System were originally defined as ones in which “people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients”. Now, a broader and more general definition is taking place in the field, referring to recommender systems as those systems that “have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options” [7].

Recommendation is a particular form of information filtering, that exploits past behaviors and user similarities to generate a list of information items that is personally tailored to an end-user’s preferences.

Fig 2.1 shows the different types of approaches for Recommender systems.

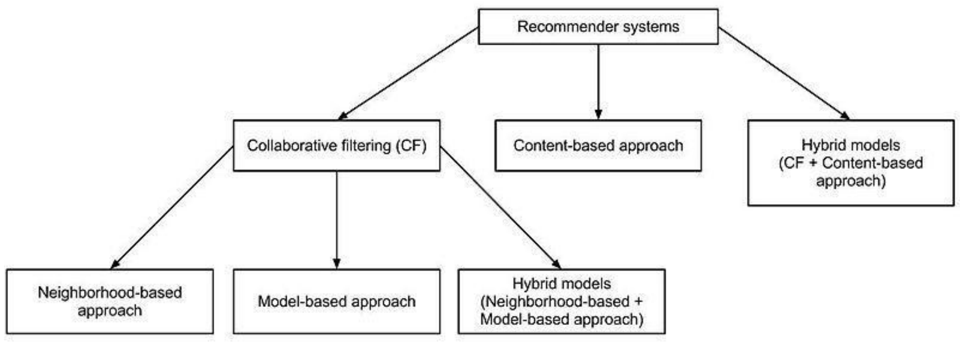


Fig 2.1. Types of Recommender System

Generally we can classify these systems into two broad groups [8].

2.1.1 Content-based systems examine properties of the items recommended. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the “cowboy” genre. It focus on properties of items. Similarity of items is determined by measuring the similarity in their properties.

2.1.2 Collaborative Filtering (CF) System recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users. It focus on the relationship between users and items. Similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.

2.2 Content-based systems

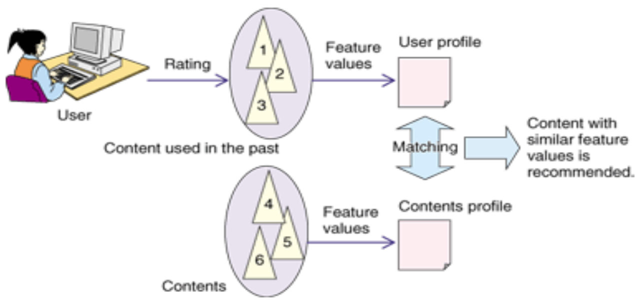


Fig 2.2. Content-based system

This approaches treat all users and items as atomic units, where predictions are made without regard to the specifics of individual users or items. However, one can make a better personalized recommendation by knowing more about a user, such as demographic information [9], or about an item, such as the director and genre of a movie [8]. For instance, given movie genre information, and knowing that a user liked “Star Wars” and “Blade Runner”, one may infer a predilection for Science Fiction and could hence recommend “Twelve Monkeys”. Content-based recommender refer to such approaches (as shown in fig 2.2), that provide recommendations by comparing representations of content describing an item to representations of content that interests the user. These approaches are sometimes also referred to as content-based filtering.

Much research in this area has focused on recommending items with associated textual content, such as web-pages, books, and movies; where the web-pages themselves or associated content like descriptions and user reviews are available. As such, several approaches have treated this problem as an Information Retrieval (IR) task, where the content associated with the user's preferences is treated as a query, and the unrated documents are scored with relevance/similarity to this query [10]. In NewsWeeder [11], documents in each rating category are converted into tf-idf word vectors, and then averaged to get a prototype vector of each category for a user. To classify a new document, it is compared with each prototype vector and given a predicted rating based on the cosine similarity to each category.

2.3 Collaborative Filtering

As the Internet continues to grow, it becomes increasingly difficult to sift through all the information that is available. With the overwhelming amount of data and choices that can be made, people need a filter to increase the Internet's usability. A technique used for dealing with this problem is collaborative filtering (also known as social filtering), which reduces the time spent searching and increases the accuracy of retrieval. In *Social Information Filtering: Algorithms for Automating* "Word of Mouth," Shardanand and Maes write:

Social Information filtering exploits similarities between the tastes of different users to recommend (or advise against) items. It relies on the fact that people's tastes are not randomly distributed: there are general trends and patterns within the taste of a person and as well as between groups of people. Social Information filtering automates a process of "word-of-mouth" recommendations. A significant difference is that instead of having to ask a couple friends about a few items, a social information filtering system can consider thousands of other people, and consider thousands of different items, all happening autonomously and automatically (Shardanand and Maes, 1995).

Collaborative filters predict someone's personal preferences for information and/or products by keeping track of their likes and dislikes, and then connecting that information with a database of other peoples' preferences to look for matches, making predictions based on such things as purchases [18].

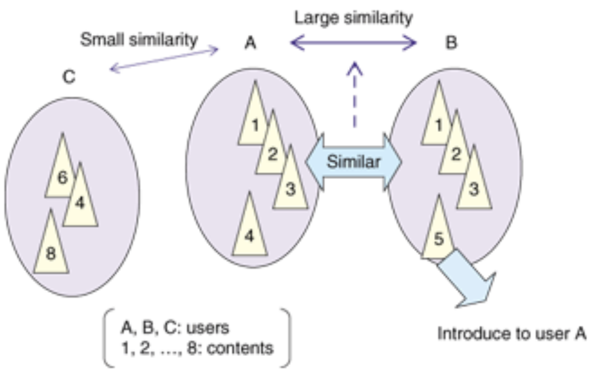


Fig 2.3. Collaborative Filtering

Collaborative Filtering (CF) systems work (as shown in fig 2.3) by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behavior amongst several users in determining how to recommend an item.

Collaborative filtering has two senses, a narrow one and a more general one [2]. In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue *x* than to have the opinion on *x* of a person chosen randomly.

Collaborative Filtering methods can be further subdivided into neighborhood-based and model-based approaches. Neighborhood-based methods are also commonly referred to as memory-based approaches [12].

2.3.1 Neighborhood-based Collaborative Filtering

The general problem in recommendations systems can be cast as, given an {$item_m$, $user_i$}, determine the mostly likely $rating_{mi}$. A common framework for generating such predictions is a neighborhood-based approach [3]. In neighborhood-based techniques, a subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for this user. Most of these approaches can be generalized by the algorithm summarized in the following steps [8]:

1. Assign a weight to all users with respect to similarity with the active user.

2. Select k users that have the highest similarity with the active user - commonly called the neighborhood.

3. Compute a prediction from a weighted combination of the selected neighbors' ratings.

The neighborhood-based algorithm calculates the similarity between two users or items, produces a prediction for the user taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple mechanisms such as Euclidean distance, Pearson correlation and vector cosine based similarity are used for this.

2.3.2 Model-based Collaborative Filtering

Model-based techniques provide recommendations by estimating parameters of statistical models for user ratings. Model-based method fit a parametric model to the training data that can later be used to predict unseen ratings and issue recommendations. Model-based methods include cluster-based CF, Bayesian classifiers, and regression-based methods.

Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data.

Most of the models are based on creating a classification or clustering technique to identify the user based on the test set.

There are several advantages with the paradigm. It handles the sparsity better than memory based ones (Neighborhood based collaborative filtering). This helps with scalability with large data sets. It improves the prediction performance. It gives an intuitive rationale for the recommendations.

2.4 Hybrid Models

Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model.

Netflix is a good example of hybrid systems. They make 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).

The term hybrid recommender system is used here to describe any recommender system that combines multiple recommendation techniques together to produce its output. There is no reason why several different techniques of the same type could not be hybridized, for example, two different content-based recommenders could work together, and a number of projects have investigated this type of hybrid: NewsDude, which uses both naive Bayes and kNN classifiers in its news recommendations is just one example [4].

3. EXPERIMENTAL SETUP

3.1 MovieLens Data Set

In our recommender system, we used the MovieLens 100K data set. MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. This data set consists of 100 000 ratings (from 1 to 5; 1 being worst and 5 being best) from 943 users on 1682 movies. Each user has rated at least 20 movies. Simple demographic information for the users (like age, gender, occupation, zip) has been included in datasets. Since these demographic information is beyond our scope, we filtered out those fields during data pre-processing part.

MovieLens data set contains following files which we have used in our recommendation system:

i. *u.data* :- It is a full data set with 100 000 ratings by 943 users on 1682 items. This is a tab separated list of **user id | item id | rating | timestamp**

ii. *u.item* :- It contains the information about the items (movies). Information like genres, IMDb URL and release date has been included in this file. The movie ids in *u.item* are the ones used in the *u.data* data set.

iii. *u.user* :- It contains the demographic information about the users like age, gender, occupation and zip code. The user ids are the ones used in the u.data data set.

iv. **.base* and **.test* files :- The data sets *u1.base* and *u1.test* through *u5.base* and *u5.test* are 80%/20% splits of the u data into training and test data. Each of u1,....,u5 have disjoint test sets. We used these training and testing set for our experimental purpose.

In addition to this MovieLens data sets, we have created one more test set with 1672 ratings by single users on 1672 movies. The rest of the 10 movies has a ratings of zero, i.e. user haven't watched those 10 movies yet. The purpose of this test data is to serve as a common test data for multiple experiments performed later to analyze the output of the system.

3.2 External tools and Libraries

3.2.1 Python Programming Language

We used python programming language (version 3+) for the implementation of our Recommender System. Python is highly used general purpose, high level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code. Because of all these reason and its simplicity we chose python for the development of the project.

3.2.2 Pandas: Python Data Analysis Library

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the python programming language. It is used in data manipulation and analysis. We used pandas for pre-processing module where the MovieLens data needs to be manipulated.

3.3 Kendall's Tau coefficient and *p*-value

Kendall's Tau is a non-parametric measure of correlation between two ranked variables. Specifically, it is a measure of rank correlation, i.e., the similarity of the orderings of the data when ranked by each of the quantities. It is named after Maurice Kendall, who developed it in 1938 [19], though Gustav Fechner had proposed a similar measure in the context of time series in 1897 [20].

Kendall's Tau represents a probability. In other words it is the difference between the probability that the observed data are in the same order versus the probability that the observed data are no in the same order. It provides a distribution free test of independence and a measure of the strength of dependence between two variables. Kendall's Tau coefficient has a value ranging from -1 to +1. A value of

+1 indicates that all pairs are concordant (consistent or in agreement), value of -1 indicates that all pairs are discordant, and a value of 0 indicates no relation (i.e. independence).

In statistical significance testing, the *p*-value is the probability of obtaining a test statistic result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true [21]. When you perform a hypothesis test in statistics, a *p*-value helps you determine the significance of your results. Hypothesis tests are used to test the validity of a claim that is made about a population. This claim that's on trial, in essence, is called the *null hypothesis.* The null hypothesis refers to a general or default position: that there is no relationship between two measured phenomena, or that a potential medical treatment has no effect. Rejecting or disproving the null hypothesis - and thus concluding that there are grounds for believing that there is a relationship between two phenomena or that a potential treatment has a measurable effect.

The primary purpose of an hypothesis test is to decide if the results of a study, based on a small sample, provide enough evidence against the null hypothesis (denoted by H_0), so that it is reasonable to believe that in a larger target population, H_0 is false, thus accepting the associated alternative hypothesis as being true.

p-value is simply a measure of the strength of evidence against $H_{0.}$ A study with a p=0.531 has much less evidence against H_0 than a study with a p=0.05. Thus, a very small *p* value does provide strong evidence that H_0 is not true.

4. RECOMMENDING MOVIES

4.1 Overview

Searching for the interesting movies is really a tough job. It would be highly preferable if someone having similar taste with you recommends the movies for you. This system focus on recommending the user with the list of interesting movies. Recommending movies follows some data mining approaches and is not just an act of random selection. The movies recommended to any specific user depends on his/her profile and behaviour. User's profile and behaviour includes his/her ratings to different movies. And based on these information, recommended movies are generated by the system.

4.2 System Architecture

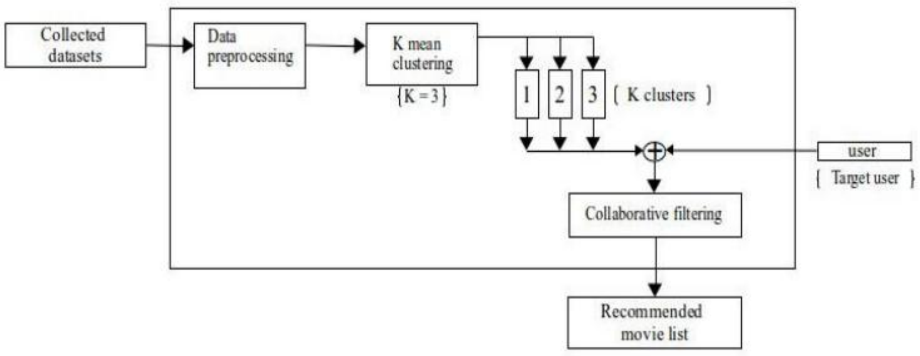


Fig 4.2. Block Diagram

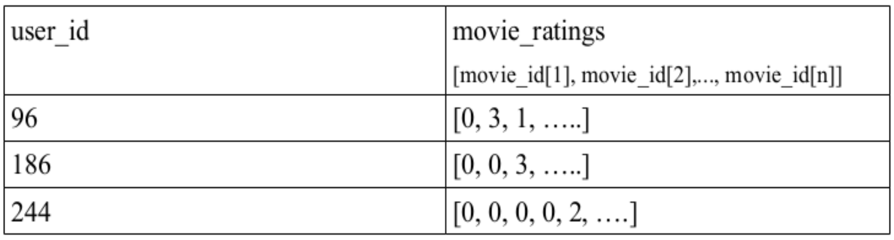
1. Data Preprocessing:- The collected MovieLens Dataset contains a data in a tab separated format as: *user-id* | *movie-id* | *ratings* | *timestamps*

Also, the data set includes some extra information (like demographic info) that needs to be filtered out before we provide it to the system. This task of filtering and formatting the data sets is done in Data Preprocessing module. This data preprocessing module will convert the format of the data set as expected by the system. Plus, it filter out the extra information like demographic information and convert the data set into the appropriate one. For instance, following tuples are the part of MovieLens **u** data with 5 ratings from 4 users on 4 movies:

96	2	3	881250949
186	3	3	891717742
96	3	1	878887116
244	5	2	880606923

This **u** data is first processed by our pre-processing module and convert it to the appropriate format. Since timestamps field is not necessary in our system, it could be removed from the pre-processing module. Also the format our system expects is of *dictionary* type in which the key is the user_id and the value is n-length *list* of ratings for each movies. So the job of pre-processing module is to filter out the timestamps field and to convert the given tuples into the dictionary type.

i.e. Output of pre-processing module is:



user_id	movie_ratings [movie_id[1], movie_id[2],..., movie_id[n]]
96	[0, 3, 1, …..]
186	[0, 0, 3, …..]
244	[0, 0, 0, 0, 2, ….]

Where rating 0 implies “haven’t watched yet”.

2. K-mean Clustering:- After processing the data’s (training + testing data), K-mean clustering takes training data as a input and create a model of clusters. This process of creating a model is known as unsupervised learning, or simply learning.

Given a set of training data, k-mean randomly initializes the **k** centroids and user gets assigned to its nearest centroid. On each iteration, the new centroid is calculated as an average of each and every users. This process is repeated until the cluster remains unchanged. The final cluster created by k-mean clustering is a model of our system. The new user for which the recommendation needed to be done is first assigned into its nearest cluster. Then the next module, collaborative filtering is applied into this cluster for that new user.

3. Collaborative Filtering:- The inputs for this module is the new/target user (for which the recommendation problem is defined) and the cluster that user belongs. The weights for each movies are calculated with respect to each and every users from that cluster and weighted movies are given to the target user. Based on this calculated weights, the list of movies (new user haven't watched yet) are recommended to the target user. For instance, the actual ratings and calculated weights for 5 movies for a target user are:

movie_id	Actual Ratings	Calculated Weights
3	0	1352.21
16	2	547.15
44	0	943.34
789	5	8373.11
234	3	6034

CF module will now check the movies with ratings 0. In our case, movie_id 3 and 44 has a actual rating 0. The weights corresponding to these movies are now compared and the movie with the highest weights are recommended first. i.e. movie_id 3 is recommended first and then movie_id 44 comes in second place.

4.3 Euclidean Distance: Finding similar user

Simply, the Euclidean distance or Euclidean metric is the “ordinary” distance between two points that one would measure with a ruler. Generally, euclidean distance is used to find the similarities between two objects in space. In recommendation problem, euclidean distance is a very simple way to calculate a similarity score between two ranking sets (of two users).

The Euclidean distance, **d** between points $\mathbf{x_i}$ ($=x_1, x_2, ..., x_k$) and $\mathbf{y_i}$ ($=y_1, y_2, ..., y_k$) is the length of the line segment connecting them and is given as,

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

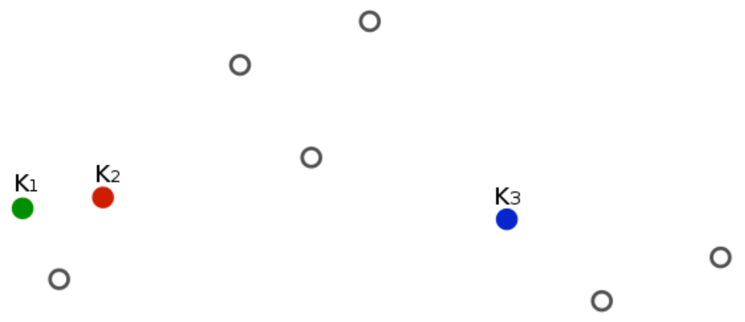
4.4 K-mean Clustering: Learning from recommendation

K-mean clustering is a Centroid-Based, unsupervised learning technique that groups a set of physical or abstract objects into classes of similar objects. The K-mean algorithm takes the input parameter *k,* and partitions a set of *n* objects into *k* clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's *centroid* or *center of cluster* [13].

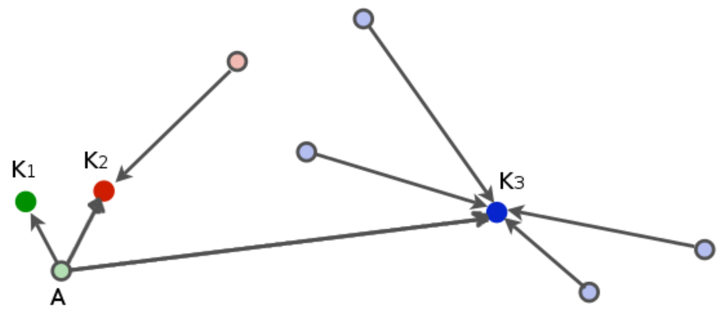
The term "*k-means"* was first used by James MacQueen in 1967 [14], though the idea goes back to Hugo Steinhaus in 1956 [14]. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside of Bell Labs until 1982 [15]. In 1965, E.W.Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy [15]. A more efficient version was proposed and published in Fortran by Hartigan and Wong in 1975/1979 [15][16].

4.4.1 Algorithm

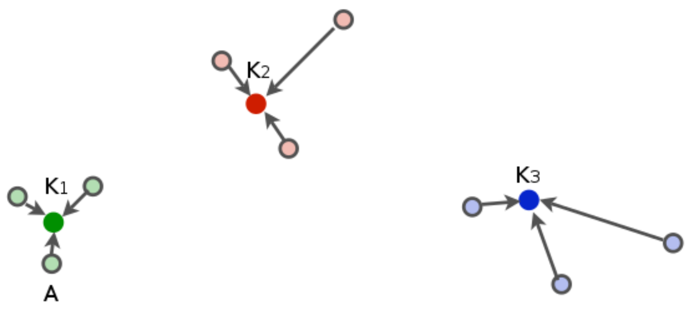
Step 1: *From the given data sets choose K initial centers at random. In the figure below, let K_1, K_2 and K_3 are three centers chosen randomly from given datasets. These chosen centers are the initial centroid for each clusters.*



Step 2: *Calculate* ***Euclidean Distance*** *between each points and the centroids of the clusters.*



Step 3: *Points with minimum distance (Euclidean) to the centroid is placed inside that cluster. e.g. A is in cluster I with centroid* K_1.



Step 4: *Calculate average of every point and assign it as a new centroid.*

Step 5: *Repeat step 2, 3 & 4 until the centroid remains unchanged. [i.e. new centroid = old centroid.]*

In recommendation problem, K-mean clustering is used to group a set of similar users (based on euclidean distance) into a single cluster such that the users in a same cluster share a similar properties. This would be useful and advantageous in the sense that it will partition the total datasets into the number of clusters and operating in single cluster will always be faster than operating on whole data sets. The program do not need to evaluate each and every users in a dataset, instead the program can evaluate the users from single cluster. i.e. if A belongs to cluster I (with centroid K_1) then recommending items (movies in our case) to the user A can be done with respect to users belonging to cluster I only.

4.5 Collaborative Filtering: Generating Recommendation

In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc [2]. In collaborative filtering, an item is considered as a black box - we don't look at its content - and user interactions (such as rating, saving, purchasing) with the item are used to recommend an item of interest to the user [17].

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). In simple sense, a collaborative filtering algorithm usually works by searching a large group of people and finding a smaller set with tastes similar to yours. It looks at other things they like and combines them to create a ranked list of suggestions [5].

The simplest Collaborative Filtering approach would be to make almost the same common prediction or recommendation for every customer. For instance, recommending a user with top-5 movies (collected from all users). But this is very basic. Our recommendation system implements the more sophisticated approach than the basic one.

Consider the friend telling you about a movie you should see. If one friend Emily (whose movie sense you tend to trust) tells you she hated a certain movie, but five other friends enjoyed the movie; you might decide to go see it. The algorithm will look at each and every user and calculate how similar they are with you. Based on their ratings and how similar they are with you, algorithm will predict the weights for each movie. This calculated weights are directly proportional to the recommendation task, i.e. the movie with higher weight will be in the top of the recommendation list.

The weight for each movies can be calculated by the following formula,

W_b = user[1].ratings * Edistance(user[1],newUser) + ...+ user[n].ratings * Edistance(user[n], newUser)

where,

W_b is a list of weights for each movie.

user[i].ratings is a ratings for each and every movie given by user[i]

Edistance(user[i], newUser) is an euclidean distance between user[i] and a new user for whom a program will recommend with a list of movies.

5. ANALYSIS & EVALUATION

5.1 Output Analysis

The output of the recommender system is analyzed to learn the characteristics of datasets. For this analysis, we conducted six different experiments with our own test data set (as mentioned earlier). Keeping the test data same for each experiments, training data used is different in each case. For first experiment, we used MovieLens **u** data as a training data and for other five experiments, five different training data is used. These five different training data is 80% splits of the **u** data. The statistics collected from these six different experiment is plotted in the graph as below:

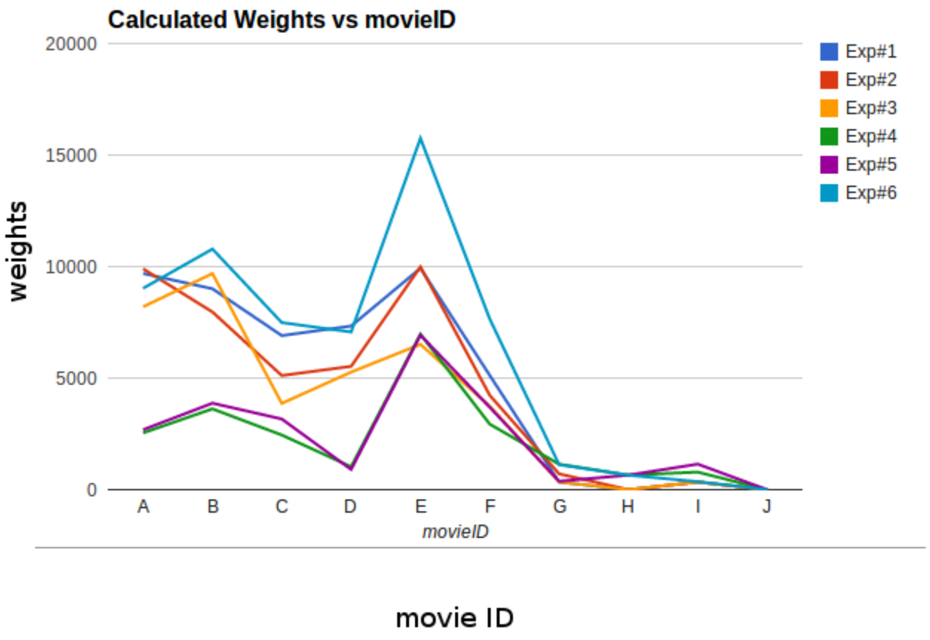


Fig 5.1 Statistics for six different experiment

The vertical axis is the representation of calculated weights for each movies given to the new/target user and the horizontal axis represents the ten movies that new user haven't watched yet. For instance, In 6th experiment the user will get the higher recommendation for movie E and lower recommendation for movie J. Here in the case of movie J, the weight calculated is zero. This implies that the cluster (where the new user belongs) does not contain any user who had watched the movies yet, i.e. every user in that cluster had a rating 0 for movie J. Similarly, movie E is the highly rated movie in that cluster.

5.2 Determine best K for K-Mean

We conducted six different experiment for six different value of K, keeping the testing and training data set same for each experiment. The purpose of this set of experiment is to analyze the appropriate value for K. The value for K must be selected in such a way that the total execution time is minimized. Obviously there exist some other good criteria for the selection of K, but for simplicity we chose the time complexity as an evaluation metrics. The statistics of K vs execution time for each experiment is plotted in the graphs below:

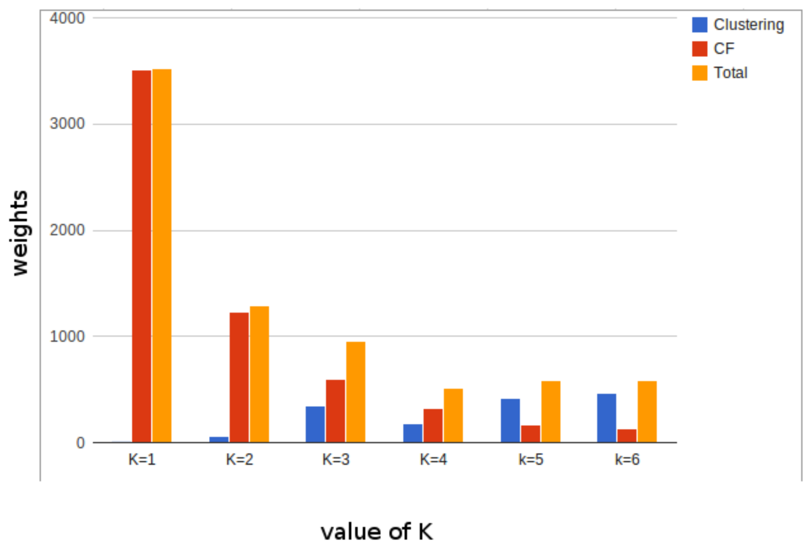


Fig 5.2(a): K vs execution time (in secs).

By analyzing the statistics above, execution time for K=4 is minimum so we could choose the value of K. But, this is not only the criteria to choose the value of K.

We analyzed the output for each value of K and got the statistic as shown in Fig 5.2(b). The graph in fig 5.2(b) compares the weight of each movies for each value of K. Vertical axis represents the weight for each movies represented in horizontal axis and labeled as A, B, …., J. For K=5 and 6, the four movies G, H, I & J has a weight zero. This is a worst case. No one in the cluster (that user belongs) knows the rating of this four movies. This may result in poor recommendation and selecting such value of K may not be wise. Thus the candidate values for K is reduced to four.

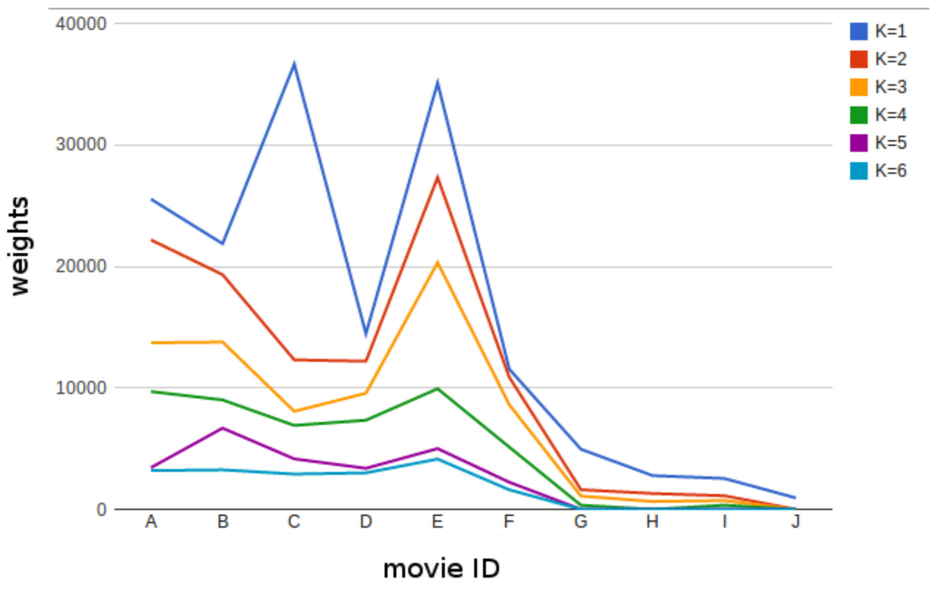


Fig 5.2(b): Comparing the output for each K.

Further analyzing the statistics, the execution time for K=1 & 2 is comparatively higher than for K=3 and K=4. Also for K=4 (in fig 5.2(b)) there are two movies with weight equals zero while there is only one movies with weight equals zero for the case of K=3 but analyzing fig 5.2 the execution time is faster for K=4.

5.3 Statistical Test

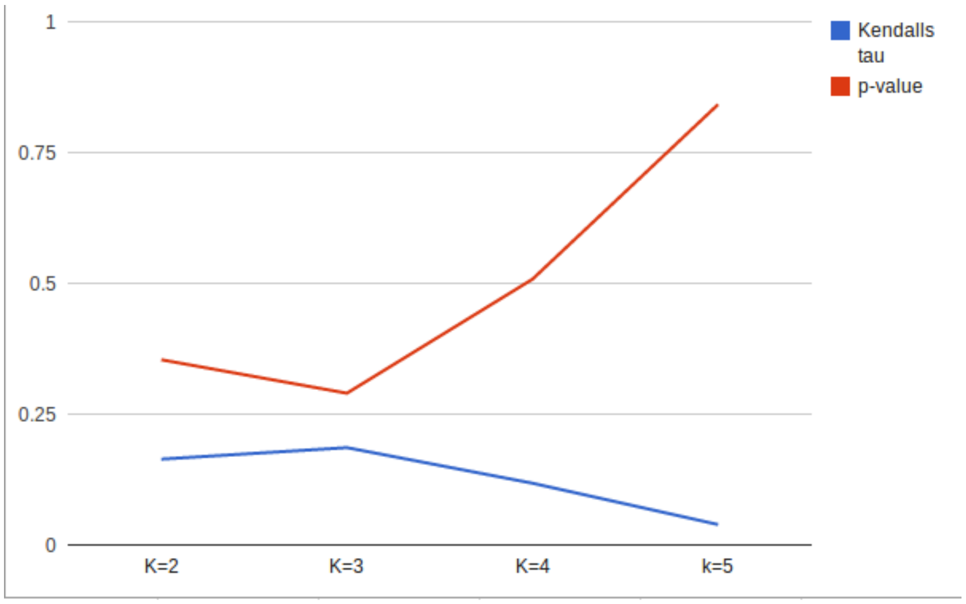


Fig 5.3: Kendall's tau coefficient and p-value

In this set of experiments we performed two statistical test, one is tau test and another is hypothesis test. Tau test calculates the association between two measured quantities and hypothesis test calculates *p*-value to support the null hypothesis.

For tau test, we calculate the tau coefficient between actual ratings and predicted ratings. For K=3 (in fig 5.3), Kendall Tau coefficient is maximum and implies that the association between these two measured quantities is higher than for any other value of K. Thus, the tau test suggest K=3 to be best and appropriate.

For hypothesis test, we calculate the *p*-value for four different values of K to support our null hypothesis (that there is no relation between actual ratings and predicted ratings). From fig 5.3, we can see that *p*-value for K=3 is smaller. This implies that, for K=3 there is not much support for our null hypothesis. There must be some relation between actual ratings and predicted ratings.

6. CONCLUSION

Lastly, Recommender Systems are a useful alternative to search algorithms since it help users discover new items they might not have found by themselves. It address the general problem of recommending items to an active user relying on the implicit sharing of preferences and experience with other users. This project thus build to solve the searching problem that today's busy world might face in daily work. Its preferable if someone (whom you trust) recommend you with the list of movies (or may be information or items), than you search for yourself. It not only saves your time, but also provides you with more movies (or extra information) that you might find it useful and interesting.

Thus, recommender system seems very promising in solving the search problem. Many company like facebook, youtube, amazon have already adapted this type of recommender system to provide the best service to their users and this is the reason why many people are using it today.

7. FUTURE WORK

This project lacks the GUI in interface. It has a simple CUI to interact with the user. This is an obvious limitation and we will work out to limit this limitation. Our future work involves to develop the complete GUI movie recommender app so that the user can interact with the app easily and more effectively.

8. REFERENCES

1. Goldberg, D. Nichols, D., Oki, B. M., and Terry, D. Using collaborative filtering to weave an information tapestry.

2. Terveen, Loren; Hill, Will (2001). "Beyond Recommender Systems: Helping People Help Each Other". Addison-Wesley. p. 6.

3. Ramesh Dommeti, Neighborhood based methods for Collaborative Filtering

4. Robin Burke , Hybrid Web Recommender Systems.

5. Toby Segaran, “Programming Collective Intelligence”, O'Reilly.

6. Shani Guy, Gunawardana Asela, “Evaluating Recommendation System”, Microsoft research.

7. Felix Hernandez del Olmo, Elena Gaudioso, “Evaluation of recommender systems: A new approach”, Universidad Nacional de Educacion a Distancia, Madrid, Spain

8. Prem Melville and Vikas Sindhwani, “Recommender Systems”, Watson Research Center, Yorktown Heights.

9. Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering.

10. Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. Communications of the Association for Computing Machinery

11. Ken Lang. NewsWeeder: Learning to filter netnews. In Proceedings of the Twelfth International Conference on Machine Learning (ICML-95), pages 331–339, San Francisco, CA, 1995.

12. Beck, K. “Test-Driven Development by Example”, Addison-Wesley.

13. Jiawei Han and Micheline Kamber, “Data Mining, Concepts and Techniques”, second edition, Morgan Kaufmann Publishers.

14. MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281–297.

15. Lloyd, S. P. (1957). "Least square quantization in PCM". Bell Telephone Laboratories Paper.

16. Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A K-Means Clustering Algorithm".

17. Satnam Alag, “Collective Intelligence in Action”, Manning Publications Co.

18. Short History of Collaborative Filtering by Moya K. Mason.

19. Kendall, M. (1938). "A New Measure of Rank Correlation".

20 Kruskal, W.H. (1958). "Ordinal Measures of Association". Journal of the American Statistical Association

21. Goodman, SN (1999). "Toward Evidence-Based Medical Statistics. 1: The P Value Fallacy."