International Journal of Engineering & Technology, *7 (3.29) (2018) 304-310*

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Comparative analysis of recommender systems and its enhancements

Immidi Kali Pradeep [1] *, M. Jaya Bhaskar [2]

[1] Research scholar KL University, India
[2] Associate Professor KL University, India
**Corresponding author E-mail: kalipradeep.i@vishnu.edu.in*

Abstract

Recommenders are being used in many applications and circumstances to make ease of social life by generating categorized and personalized recommendations to the individuals. These categories may be chosen by the users to get recommendations for movies, songs, products and various services etc. One of the challenges of a recommender system is to generate recommendations in real time to many people by analyzing huge amount of data. In this paper, authors considered traditional recommender and hybrid recommender techniques to generate recommendations. Traditional recommender systems include similarity measure, matrix factorization, co-clustering and slope-one approach, where as the second type of recommender system consists of the role of hybridization techniques and contextual parameters with traditional recommenders. Here, authors worked on movie lens dataset with above mentioned recommender systems and observed that SVD approach has less RMSE and MAE values comparing with other models.

Keywords: *Recommender System; Information Retrieval; Similarity Measure; Contextual Parameters*

1. Introduction

Recommendation systems are the type of information retrieval mechanisms used to predict user's interest in a given context on an item. Recommendation system found its applications in NEWS personalization, product/item recommendations, in E-commerce websites, Songs and movie recommendation in online streaming websites and friend recommendation in social networking sites. The process of making recommendations is two steps: 1) Learning the data: also known as model building step or offline step and 2) Generating predictions: also known as execution or online step. In many cases, Recommendations should be real time, so offline step should be able to scale up a massive amount of data and help to generate real-time recommendations.
The challenge of recommendation systems is to mainly understand the user's requirement and recommend items which are related to a user's interest which he may not know but like it when recommended.
Recommendation system uses many parameters to generate recommendations [1]:

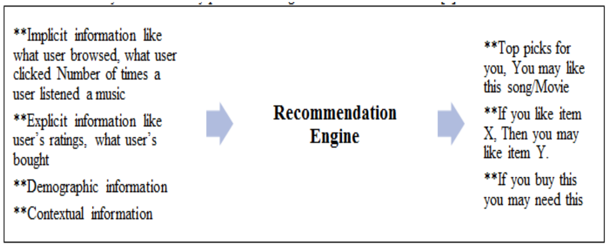

**Implicit information like what user browsed, what user clicked Number of times a user listened a music
**Explicit information like user's ratings, what user's bought
**Demographic information
**Contextual information
Recommendation Engine
**Top picks for you, You may like this song/Movie
**If you like item X, Then you may like item Y.
**If you buy this you may need this

Fig. 1.1: Recommendation Process.

1) The history of recommender systems ranges from Google Page rank system, Pandora music streaming website [4], CDnow [2], to Amazon.com [3]. These websites use recommendation engines to provide valid recommendations in real time. Recommender systems are of two types. 1) Non-personalized recommenders & 2) Personalized recommenders. Non-personalized recommenders never consider user's interest into account to provide recommendations. These types of recommenders are highly beneficial. For example, in a NEWS recommender system, even if a user is not interested in politics, it is essential to recommend a major political news like new president-elect of the country to all users. On the other hand, Personalized recommenders customise recommendations according to user's taste. Personalized recommender systems are of three types.
1) Content-based,
2) Collaborative based and
3) Hybrid approach.

This paper is organized as follows. The next section describes non-personalised recommender system.
Section 2 gives a brief overview of content based personalized recommendation. Section 3 is about collaborative filtering techniques. Section 4 presents hybrid recommender system. Section 5 provides the metrics for evaluating a recommender system. Section 6 presents the survey results of traditional recommendation systems. The final section provides concluding remarks and future research directions.

2. Non-personalized recommender system and content-based systems

2.1. Non-personalized recommender

Non-personalized recommender system gives common recommendations to all users. The simple formula used by earlier non-personalized recommender systems like Zagat is [5]

Score= (Mean (ratings)) *10

The rating values are between 1 to 5 and mean multiplied by ten to make it non-decimal.
Some non-personalized recommender systems like conda nast[5] used the formula to calculate rating prediction Ru as:

$$\mathrm{Ru} = \frac{\sum_{\mathrm{i=i^*}} \mathrm{ri}}{\sum_{\mathrm{i=I}} \mathrm{ri}} \times 100$$

Where $\sum_{\mathrm{u}} \mathrm{r_{ui}}$ is Number of people with good ratings (say 4 and 5 out of 5-star ratings) for the item i. And I is the total number of ratings for that item.
Non-personalized recommenders are used to provide new movie review. For example, the average rating of a new movie in blogs has nothing to do with a user's interest.
In cases of fewer ratings, mean can be misleading. So, the mean ratings are modified by,

$$\mathrm{Ru} = \frac{\sum_{\mathrm{u}} \mathrm{r_{ui}} + \mathrm{k}\mu}{\mathrm{n+k}}$$

Where, is the sum of user ratings for item i, n is the total number of ratings, k is a strength of evidence required to overcome global mean and μ is the average rating of item i. The advantages of non-personalized recommender system is less time complexity, less space complexity and can generate recommendations for even new users. And, the drawback of non-personalized recommender is it never consider the user's interest.

2.2. Content-based recommendations

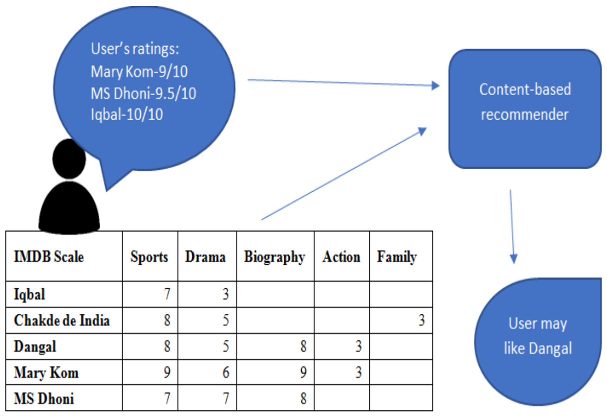
Utility matrix is considered as input to make recommendations this matrix has rows as users and columns as items. The intersection is the vector of ratings given by a user on an item.
The main agenda of a content-based filtering is to find similar items to items the user is looking for [6]. Content-based recommenders found their applications in digital documents, online articles and NEWS portals.
Steps in content-based recommenders:

1) Initially, products are represented in the form of attributes or descriptors. For example, books can be described by Genre (Science fiction, Comedy, Drama), Author's Name, words used in the book.
2) Represent the values for each descriptor by a vector in a multidimensional vector space.
3) In the same way, a user profile is created for each user based on his purchase history, explicit ratings, and reviews.
4) Now the user has same attributes like the genre (List of movies they prefer), Author's name (List of books they bought of an Author).
5) Now map each user to a movie similar to his taste.

The similarity between two items are represented by,

$$\mathrm{sim(A,B)} = \frac{\mathrm{A.B}}{||\mathrm{A}||\ ||\mathrm{B}||} = \frac{\sum_{\mathrm{i=1}} \mathrm{A_i B_i}}{\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{A_i^2}}\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{B_i^2}}}$$

Where $\mathrm{A_i}$ and $\mathrm{B_i}$ are components of vectors A and B respectively.
Music genome project by Pandora [4] has successfully implemented content-based filtering for recommending music to its users.
Some of the pros of content-based recommenders are 1) It cannot recommend items to users with unique tastes. 2) It can also recommend new and less popular items 3) It has a valid proof for recommending an item to the user. Content-based recommenders also have some disadvantages 1) It is a difficult task for discovering descriptors for every item. 2) In some cases, over specialization may lead to decidedly less or no similar items and 3) Cold start problem: User profiles are generated by aggregating item profiles that user has rated. But for a new user, the user profile is empty. So, no perfect item is recommended.



IMDB Scale	Sports	Drama	Biography	Action	Family
Iqbal	7	3			
Chakde de India	8	5			3
Dangal	8	5	8	3	
Mary Kom	9	6	9	3	
MS Dhoni	7	7	8		

Fig. 2.1: Content-Based Recommendations.

3. Collaborative filtering

Collaborative filtering predicts the preference of a user on an item based on the taste of another user [7]. The criteria here is to find a set of users similar to a user u and recommend the items consumed or preferred by these users to the user u.

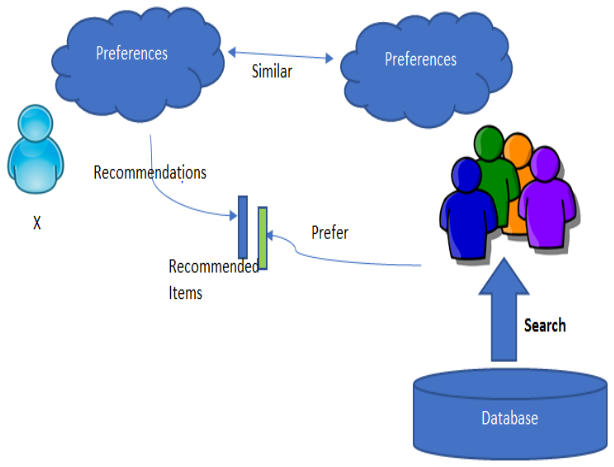


Fig. 3.1: Collaborative Filtering.

The first approach for collaborative filtering is baseline approach. Some users tend to be strict rates and some as lose raters. This is the case with items as well as movies. So, observed deviation of users and items along with μ provide better recommendations compared to simple average. The rating prediction $\boldsymbol{\hat{r}_{xi}}$ in baseline approach is,

$$\hat{r}_{xi} = \mu + b_u + b_i$$

Where is the average ratings of all available ratings in the system, is observed deviation of user x and $\boldsymbol{b_i}$ is observed deviation of item i.Baseline approach is a relatively simple approach. This approach is mostly used when no much information is available about user and items.

$$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u;i} - \mu)$$

$$b_i = \frac{1}{|U_{i|}} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

Collaborative filtering can be done by using two methods 1) Memory-based methods and 2) Model-based methods.

3.1. Memory-based methods

Memory-based collaborative filtering techniques find similarity between user/items using neighborhood methods [8]. The similarity is typically calculated by Pearson correlation, cosine similarity measures, and Jaccard coefficients. Here, the similarity between users/items is computed offline. There are two types of collaborative filtering techniques, user-user collaborative filtering, and item-item collaborative filtering.

3.1.1. User-user collaborative filtering [7]

The basic idea of user-user collaborative filtering is, let us consider a user x, find a group of users whose likes and dislikes are similar to defined user x. For example, x likes the same movies, the group of users like and x dislike the movies the group doesn't like. These group of users is called neighborhood of x. After finding the neighbor of x, find the set of items/movies which are not bought/seen by user x but are liked by neighborhood users. Then, recommend those items to user x.
Consider a user item utility matrix where Ui is the user, Mi is items, and ri is rating vectors ri.

Table 3.1: User-Item Utility Matrix

	M1	M2	M3	M4	M5	M6	M7
U1	4			5	1		
U2	5	5	4				
U3				2	4	5	
U4		3					3

In order to find the similarity between users, similarity measures like Jaccard similarity, cosine similarity or Pearson similarity are used.
Jaccard similarity [8]:

$$Sim(A,B) = \frac{|ra \cap rb|}{|ra \cup rb|}$$

Where, $|ra \cap rb|$ are set of movies watched by both a and b in common and $|ra \cup rb|$ is a total number of movies watched by a and b. Here, the rating given by the user to a movie is not considered important so, by using the above formula, user u1 and u3 are considered identical which is not true.

1) Cosine Similarity:

To incubate rating factor into consideration, cosine similarity is very useful. The similarity is represented as the dot product of two vectors and dividing it by the product of Euclidian norms.

$$\text{Sim (A, B)}=cos(\vec{a},\vec{b})=\frac{\vec{a}.\vec{b}}{||\vec{a}|| \times ||\vec{b}||}=\frac{\sum_{i \in I_{a,b}} r_{a,i}\, r_{b,i}}{\sqrt{\sum_{i \in I_a} r_{a,i}^2}\sqrt{\sum_{i \in I_b} r_{b,i}^2}}$$

Where r_{ai}, is the rating of the user a on item i and r_{bi} is the rating of user b on item i.

2) Pearson similarity:

In practice, some users are easy raters and some rates are strict raters. In order to consider the deviation, center cosine similarity or Pearson correlation is used. The Pearson correlation between two users a & b is:

$$\text{Sim (A, B)}=\frac{\sum_{i \in I_{a,b}}(r_{a,i}-\bar{r}_a)(r_{b,i}-\bar{r}_b)}{\sqrt{\sum_{i \in I_{a,b}}(r_{a,i}-\bar{r}_a)^2}\sqrt{\sum_{i \in I_{a,b}}(r_{b,i}-\bar{r}_b)^2}}$$

Where, $\bar{r}_a$ is average ratings of all items consumed by the user a and is average ratings of all items consumed by user b.
To predict a user's interest or rating over an item i is given by:

$$r_{a,i} = \frac{\sum_{b \in N} S_{ab} r_{bi}}{\sum_{b \in N} S_{ab}}$$

Where S_{ab} is the similarity between user a and b and $r_{a,i}$ is rating prediction of the user a on item i.

3.1.2. Item-item collaborative filtering

In case of user-user collaborative filtering, the similarity between users is calculated. But in many cases, the number of users are more than the number of items. So, instead of starting out with the user, it is a better choice to pick an item i and finds out items similar to an item i, then we are going to estimate the rating for an item I based on the rating of other similar items [9]. The similarity can be measured by using same formula jacquard, cosine, and Pearson correlation.
The rating prediction formula for item-item collaborative filtering is:

$$r_{xi} = \frac{\sum_{i \in N(i;x)} S_{ij} . r_{xj}}{\sum_{j \in N(i;x)} S_{ij}}$$

Where S_{ij} is the similarity between item I and j, r_{xj} is the rating of user x on item j, and N(i;x) is set of items rated by x similar to i.
The most complex step of above collaborative filtering is to calculate similarity. The updated utility matrix should be scanned every time to calculate the similarity. Consider |U| be the size of utility matrix, the time complexity for finding k most similar user/items is given as 0 (|U|). It is impractical to scan utility matrix every time to calculate similarity and generating prediction in real time. One solution is to precompute similarity measures once in a day or two. But still, if we use naïve bays theorem, the time complexity turned out to be O (n*(|U|). Where n= number of users/items. Some solutions for the above-mentioned problem is to; 1) Find out the nearest items/users in a regular manner. 2) Use clustering to pre-group items into groups and restricting the search space to a cluster. 3) Dimensionality reduction techniques can also be used to reduce the search space.
Memory-based Collaborative filtering works for almost any types of filtering like books, movies and products without the need of feature selection. But, it also suffers from some drawbacks like; 1) Cold start problem: We need to have enough of users to calculate the similarity. 2) Sparsity: There may be millions of users and millions of items and most of the users have not rated most of the items. Therefore, the user-item matrix is very sparse. 3) First rater problem: Suppose there is a new item in the catalog, we cannot find similar users or items because nobody has rated it. 4) Popularity bias problem: In many cases, collaborative filtering tends to recommend common items which are not positive surprise. This effect is referred to Harry Porter's effect [10].

3.2. Model-based Collaborative filtering

In contrast to the memory-based algorithms, model-based algorithms try to model the users based on their past ratings and use these models to predict the ratings on unseen items. Some of the techniques based on collaborative filtering are SVD, SVD++, Matrix factorization using gradient descent, Probabilistic matrix factorization, Co-Clustering, Slope one approach.

3.2.1. Singular value decomposition

In user-user collaborative filtering techniques and item-item collaborative filtering techniques, the similarity measure is used to derive computations. These methods usually overfit the data. SVD is a model based collaborative filtering techniques that work on features rather than user and item directly.
Singular value decomposition [11] is a method for dimensionality reduction technique where user item utility matrix is decomposed into three parts.
Consider a user-Item utility matrix 'A' of size m×n. This matrix is decomposed into three components

SVD (A) = U×S×V^T

Here, A is a user-item matrix of size m×n, U is a user-feature orthogonal matrix of size m×k, V^T is a feature-item orthogonal matrix of size k×n and S is a diagonal feature matrix of size k×k.

$$\begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \times \begin{bmatrix} s_{11} & \cdots & s_{1k} \\ \vdots & \ddots & \vdots \\ s_{k1} & \cdots & s_{kk} \end{bmatrix} \times \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{k1} & \cdots & v_{kn} \end{bmatrix}$$

The elements in diagonal matrix S is determined in a way that s1>s2>s3…Sn. The elements which are less than 'r' are dropped and the corresponding columns and rows are also discarded to keep only U_k and V_k^T.

Now, $A_k = U_k \times S_k \times V_k^T$

Ak is an approximation of A and is measured in terms of Frobenius norm [12] ($||A-A_k||$) which is outside scope of the paper. Ak produces better results than A because A_k has less noise compared to A.
The rating prediction r_{ij} for customer i on product j is,
SVD assumes that the utility matrix has all entries. But the utility matrix has a lot of missing values. So, the factors derived from SVD is not accountable.

3.2.2. Matrix factorization model using gradient descent approach

In SVD, the original matrix is decomposed using linear algebra. It is noticed that this decomposition involves large computations that slow the system. In Matrix factorization [13], missing values are ignored, and best k-rank approximation for the available ratings is found.

$$\begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{m1} & \cdots & R_{mn} \end{bmatrix} \approx \begin{bmatrix} q_{11} & \cdots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{m1} & \cdots & q_{mk} \end{bmatrix} \times \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{k1} & \cdots & p_{kn} \end{bmatrix}$$

Given a matrix R, it is approximated into two matrices Q and P^T respectively.

$R \approx Q.P^T$

Here, Q is an items-factor matrix. These columns are less than the number of users. Similarly, matrix P is a factor-user matrix. Every row of Q and every column of P is a three-dimensional representation of both user and item.
Stochastic gradient descent approach can be used to decompose R. In this method; guesswork is done to decompose original matrix and compute the error by performing rating prediction on the decomposed matrix and compare with original ratings. If the error is large, then update the predicted value and iterate the comparison until convergence.

The rating prediction of user x on item i is given by

$$\hat{r}_{xi} = \sum_f q_i . p_x^T$$

Where qi is ith row in matrix Q and px is column x of matrix p^T.

But to provide more accuracy, it is desirable to combine $\hat{\boldsymbol{r}}_{\boldsymbol{xi}}$ with baseline estimate as discussed in section.

$$\hat{r}_{xi} = \mu + b_x + b_i + q_i . p_x^T$$

Gradient descent approach has some advantages against SVD 1) it is very fast as compared to SVD. 2) It ignores missing values and works with available data. The drawback of gradient descent approach is, If the data is skewed and is very sparse, Matrix factorization fails to give correct predictions.

3.3.3. Probabilistic matrix factorization models

The basic idea for probabilistic matrix factorization [14] is the assumption of data generated using random process. We make an assumption about the process to generate the data and learn about parameters that would generate same data as that of available.
For example, In the case of non-personalized recommenders, it is assumed that the probability of a person u to buy a product i is expressed as,

$$p(i|u) = \frac{(Number\ of\ buys\ of\ i)}{(Total\ number\ of\ buys\ of\ all\ item)}$$

But in the case of the personalized recommender, it is needed to predict the probability of a particular user buying a particular item. It can be done using probabilistic latent semantic analysis.
Here, p (i|u) is the probability of a user u buying an item i.
P(z|u): The probability of user picking a random factor z. for example, what is the probability that user decides to watch an animated movie?
P(i|z): The probability of a user buying item I after selecting feature z. Suppose a user decides to watch an animated movie, what is the probability that item I is chosen by the user?
Here p(i|z) and p(z|u) are stochastic matrices unlike orthogonal like in the case of SVD. Means, it includes probability distribution instead of linear algebraic vector spaces. This kind of probability distributions can be derived using expectation maximization algorithm.
The rating prediction of user u on item i is given by
Here, it is assumed the ratings are normally distributed with a mean determined by the dot product of user
The disadvantage of Probabilistic matrix factorization models is; it is very slow to decompose.

3.3.4. Co-Clustering based collaborative filtering

Recommender system based on SVD and similarity measure provide useful recommendations. But, these techniques have high time and space complexity. These methods also work best in a static environment. In reality, most recommender systems need to provide recommendations in real or near real time. The basic idea is items & the users are allotted to clusters and co-clusters. In co-clustering, the neighborhood of users & items are obtained, and predictions are generated using mean ratings of co-clusters. The approximation of user-item rating matrix is obtained using biases of individual user and item along with co-clustering average [19]. The original user-item utility matrix is decomposed using the co-clustering approach which includes the user-item biases.
Let A be the original user-item utility matrix, the rating prediction of user i on item j is given below by [20].

$$\widehat{A_{ij}} = A_{gh}^{COC} + \left(A_i^R - A_g^{RC}\right) + (A_j^C - A_h^{CC})$$

Where g is set of users, h is set of items, and A_i^R, is the average rating of a user, A_j^C is average ratings of an item, A_g^{RC} is the average rating of item cluster, A_h^{CC} is average ratings of user cluster, and A_{gh}^{COC} is the average of corresponding co-cluster.

3.3.5. Slope one approach

The principle of popularity differential is used in slope one approach [21]. Means, consider a user U_1 who has rated item i and j. And consider a user U_2 who has rated item i but not j. Now the rating prediction Ruj of user U_2 on item j is given as,

Ruj=Rating of user U_2 on item i + (Difference between ratings of user U_1 on item i and j)
In this approach, users who have rated some items as of target users and items the target user also rated are considered for generating recommendations. r_{ui} is the rating of user u on item i and r_{uj} is the rating of user u on item j. The average deviation dev(i,j) is computed by,

$$dev(i,j) = \frac{1}{|U_{ij}|}\sum_{u\in U_{ij}} r_{ui} - r_{uj}$$

The rating prediction of user u on item i is given by
where Ri(u) is the set of relevant items, i.e. the set of items j rated by u that also have at least one common user with i. dev(i,j) is defined as the average difference between the ratings of i and those of j.

4. Context-based recommender systems

In a traditional recommender system, user item utility matrix is used as input for generating recommendations. But, in a contextual recommender system, user's information, item information and contextual information are used to provide recommendations. Contextual recommender takes contextual information like location, time, purchase purpose to provide more precise recommendations [16]. The context can be divided into four types. They are:
1) physical context: It includes entities like time, location, temperature, light, and weather as contextual parameters. 2) social context: It represents the influence of other people along with the user and whether the user is in a group or alone. 3) interaction media context: This context deals with the device used by the user (Laptop, smartphone, public kiosks) to access the system, the recommendations may be music, video, text. 4) modal context: It deals with mood and state of the user in present situation.
Contextual information can be used with traditional recommenders in three ways.
1) Pre-filtering: In this approach, the user-item utility matrix is filtered based on contextual information. This filtered matrix is then used by any traditional method for generating recommendations. 2) Post-filtering approach: In this approach, the recommendations are generated by using any conventional recommendation systems. These recommendations are filtered using available contextual information. 3) Integrated approach: In this method, the contextual information is combined with traditional recommendation modeling approach to generate recommendations.
Context-aware recommendation systems are found in many applications. N Hariri at all [17] used this approach in music recommendations. Soha A.El-Moemen Mohamed at all [18] used this approach for recommending places or locations based on mood and weather.

Table 4.1: Typical Parameters to Consider in A Contextual Recommender System.

User	Movie	Time	Location	Companion	Rating
U1	Titanic	Weekend	Home	Friend	4
U2	Titanic	Weekday	Home	Friend	5
U3	Titanic	Weekday	Cinema	Sister	4
U1	Titanic	Weekday	Home	Sister	?

Contextual information is used in three ways [4]. 1) contextual matching: where only those profiles are considered which match with current profiles. For example, while considering context <time = weekday, location=home, companion = sister> only profiles matching this profile are considered. 2) Contextual relaxation: Here subset of matching profiles is considered. For example, out of 3 contexts < time = weekday, location=home, companion = sister> if either 1 <time, location or companion> or 2 <time or location, time or companion, companion or location >same relevant contests are considered. 3) context weighing: here all contextual information is considered, but every context is given weight.

The primary challenge of contextual recommenders is to determine potential contextual parameters from available entities in a particular situation. For example, there may be many parameters like time, location, temperature, device used to access the system. Selecting the best parameters for generating recommendations is important. The second challenge is cold start problem, considering contextual information further reduces the search space.

5. Hybrid recommenders

Different algorithms have their strengths and weakness. For example, collaborative filtering works well when there are lots of users & items. Content-based filters often work without much user-item interaction data. In practice, hybrid techniques are used to take advantages of different recommender systems to produce better recommendations. We can hybridize recommender systems using following methods [15].

5.1. Combined item score

Combined item score takes the linear blends of multiple recommender algorithms.

$$s(i;u,q,x) = b + \beta_1 s_1(i;-) + \beta_2 s_2(i;-)$$

Where β is Blending weights or weight of a recommender system and b is baseline offset. In General, b is the number of rating by the user or number of ratings for an item. Above equation can be extended by replacing .This can be written as,

$$s(i;-) = b + f_1(u,i)s_1(i;-) + f_2(u,i)s_2(i;-)$$

Where, $\mathrm{f_i(u,i)}$ is a function which defines the characteristics between users and items. The above equation is called feature vector linear stacking.

5.2. Combined item ranks

Here, each recommender gives its output score to a set of items. These scores are aggregated to compute the overall score of items. For instance, consider below table,

Table: 5.1: Example of Combined Item Ranks

Recommender algorithm X's rank to items	Recommender algorithm Y's rank to items
A- Rank 1	B-Rank 1
B- Rank 2	C-Rank 3
C-Rank 3	D-Rank 3
D-Rank 4	A-Rank 4

here, X ranks item B as 2 and Y ranks item B as 1, So, the overall rank of B is 1 or 2. Similarly, X ranks item A as 1 and Y Ranks A as 4. So, the overall rating of A may be 2 or 3 and so on.

5.3. Switching hybrid recommender system

A system uses different recommenders in different situations. For example, in case of insufficient ratings, the recommendations would be generated by using content-based filtering. In case of sufficient ratings by users for items, collaborative filtering will be used. Similarly, if no user is logged into an e-commerce site, popularity based recommender system may be used to generate recommendations since no information about the user is not available. And, when the user is logged in, collaborative filtering may be used to generate recommendations.

5.4. Mixed recommenders

At a given instance, the recommendations are generated by using different algorithms. For example, for generating X=x1+x2+x3 recommendations, x1 recommendations are generated by using content-based recommender, x2 recommendations are generated by using collaborative filtering and x3 recommendations are generated by using popularity-based recommenders.

5.5. Feature combination recommender system

Here, the recommendation logic of one recommender system is used instead of using the whole recommender by using the concept of pseudo-users. For example, in case of books, we have a pseudo user for different books. If user u likes a book B1 written by Author A, then there will be a pseudo user who likes all books written by author A and does not like any other books. This pseudo-user is considered as a similar user to user u. If author A writes any other book B2, the system do not need some people to rate book B2 to generate commendations. Instead, the pseudo-user will be automatically updated that he likes all books written by author A. Since pseudo-user is a user similar to user u, Book B2 will also be recommended to user u.

5.6. Cascade recommender systems

A recommender system generates a set of recommendations. These recommendations are refined by another recommendation system to produce better recommendations.
Hybrid recommender systems have some drawbacks. First, these algorithms should be neatly tuned. Otherwise, they may lead to errors or bad recommendations. Another drawback is computational cost and time complexity that need to be balanced to produce real-time recommendations

6. Experimental results

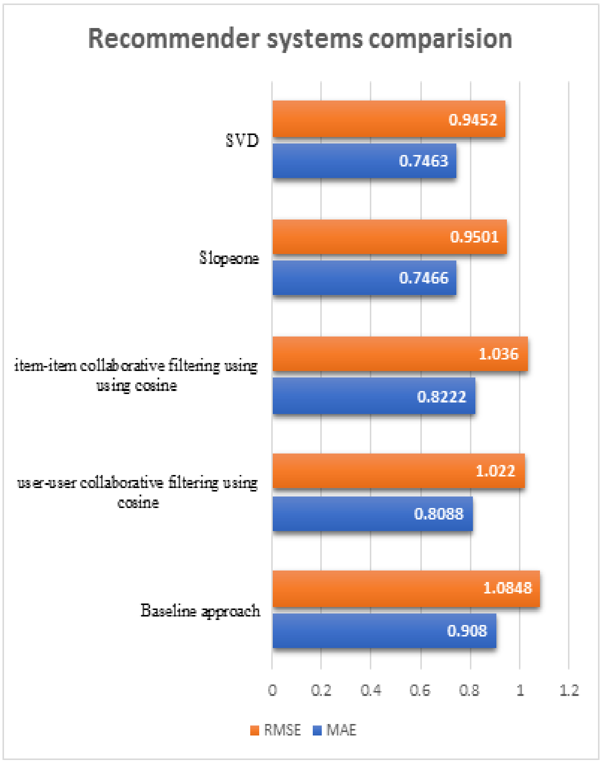
This paper explores evaluation of eight recommendation system algorithms. These algorithms are applied on movie lens 100k dataset. This dataset is taken from grouplens research group [22]. The dataset has 100,000 rating from 1 to 5 from 943 users on 1682 movies. Root mean square (RMSE) and mean absolute error(MAE) are used to evaluate these eight recommender system algorithms [23].

$$RMSE = \sqrt{\frac{1}{|\tau|}\sum_{(u,i)}(\hat{r}_{ui} - r_{ui})^2}$$

User is the test set, u is the user, i is the item, is the original rating given by user u on item i. and is the rating prediction of user u on item i by the recommender system.

$$MAE = \frac{1}{|\tau|}\sum_{(u,i)}(\hat{r}_{ui} - r_{ui})$$

The alternative measure of evaluating recommender system is Mean absolute error and is given below as,
The RMSE and MAE of eight recommender systems are as follows. This experiment is performed in offline mode. And, it is observed that SVD outperformed among all eight recommender systems.



7. Conclusion and future scope

In this paper, we compared some traditional recommender systems and observed that SVD outperformed among all algorithms. It is noticed that techniques above need large computations and are challenging to implement in online mode. In most cases, recommender systems also suffer from cold start problem. It has been observed that user's taste changes rapidly. Scaling according to the need of the user and yet having stabilized recommender system is always an order of the day. Techniques from fuzzy neural networks and artificial intelligence can be used to have such recommender systems.

References

[1] F.O. Isinkaye , Y.O. Folajimi , B.A. Ojokoh , Recommendation systems: Principles, methods and.
[2] Evaluation, Egyptian Informatics Journal (2015) 16, pg 261–273.
[3] Magnus Mortensen, Design and Evaluation of a Recommender System, INF-3981 Master's Thesis in Computer Science, Faculty of Science Department of Computer Science University of Tromsø.
[4] Greg Linden, Brent Smith, and Jeremy York, Amazon.com Recommendations Item-to-Item Collaborative Filtering, February 2003, IEEE Internet computing, pg 76-79.
[5] Music Genome Project® Pandora.
[6] Michael D. Ekstrand, Joseph A Konstan, coursera, Introduction to Recommender Systems: Non-Personalized and Content-Based https://www.coursera.org/learn/recommender-systems-introduction/lecture/ZkG45/summary-statistics-i.
[7] D. Asanov. Algorithms and methods in recommender systems. Berlin Institute of Technology, Berlin, Germany, 2011.
[8] Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan, Collaborative Filtering Recommender Systems, Foundations and Trends in Human-Computer Interaction Vol. 4, No. 2 (2010) pg 81–173.
[9] L Lü, M Medo, CH Yeung, YC Zhang, ZK Zhan Recommender systems- Physics Reports, 2012 – Elsevier.
[10] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl,, Item-based Collaborative Filtering Recommendation Algorithms, WWW10, May 1-5, 2001, Hong Kong.
[11] Jeriad Zoghby, Nigel Paice, Personalized Recommendations: Finding the needle in todays.
[12] Ever-growing digital haystack, Accenture Interactive – Point of View Series 2014.

[13] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Incremental singular value

[14] Decomposition algorithms for highly scalable recommender systems. Fifth International

[15] Conference on Computer and Information Science, 27–28.

[16] Weisstein, Eric W. "Frobenius Norm." From MathWorld-A Wolfram Web Resource. http://mathworld.wolfram.com/FrobeniusNorm.html.

[17] Yehuda Koren, Robert Bell and Chris Volinsky Matrix factorization techniques for recommender systems, IEEE Computer Society, August 2009, pg: 42-49.

[18] Ruslan Salakhutdinov and Andriy Mnih , Probabilistic Matrix Factorization, Advances in Neural Information Processing Systems 20 (NIPS 2007) pg 1-8.

[19] Burke, R. User Model User-Adap Inter (2002) 12: 331. https://doi.org/10.1023/A:1021240730564.

[20] Adomavicius G., Tuzhilin A. (2015) Context-Aware Recommender Systems. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA https://doi.org/10.1007/978-1-4899-7637-6_6.

[21] N Hariri, B Mobasher, R Burke. "Context-aware music recommendation based on latent topic sequential patterns", RecSys'12, September 9–13, 2012, Dublin, Ireland.

[22] Soha A.El-Moemen Mohamed, Taysir Hassan A.Soliman, Adel A.Sewisy. A Context-Aware Recommender System for Personalized Places in Mobile Applications International Journal of Advanced Computer Science and Applications, Vol. 7, No. 3, 2016.

[23] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In KDD, pages 509–514, 2004.

[24] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," Fifth IEEE International Conference on Data Mining (ICDM'05), 2005, pp. 4 pp.-.doi: 10.1109/ICDM.2005.14.

[25] Daniel Lemire, Anna Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering" Proceedings of the 2005 SIAM International Conference on Data Mining. 2005, 471-475.

[26] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872.

[27] Guy Shani and Asela Gunawardana, Evaluating Recommender Systems, Recommender Systems Handbook, 2011.