International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 07 | July 2019 www.irjet.net p-ISSN: 2395-0072

Hybrid Book Recommendation System

Anagha vaidya, Dr. Subhash Shinde

[1]*ME Computer Engineering, Department of Computer Engineering, Lokmanya Tilak College of Engineering, Mumbai, India.*
[2]*Vice Principle, Lokmanya Tilak College of Engineering, Mumbai, India.*

Abstract - *In this age of information, it is very difficult to find the right information from the enormous amount of data present in the online platforms. Recommendation system sorts through massive amounts of data to identify interest of users and makes the information search easier. In this paper, we have presented a model for a web-based personalized hybrid book recommendations system which exploits varied aspects of giving recommendations apart from the regular collaborative and content-based filtering approaches. Temporal aspects for the recommendations are incorporated. Also, for users of different age, country and their interests, personalized recommendation can be made on these demographic parameters. We are taking some information from user while signup which help to get more appropriate recommendations based on individual user interest and thus an attempt to overcome cold start problem. Three types of scenarios are covered in this paper viz. if user is new then recommendations are made depending upon user interests, second is recommendations based on past purchase history and the last is recommendation by using different algorithms namely K Nearest Neighbor (KNN), Singular Value Decomposition (SVD), Restricted Boltzmann Machines (RBM) and cosine similarity. It reduces dependency of rating-based system.*

Key Words: ***Hybrid Recommendation system, Collaborative filtering, Content filtering, and Demographic filtering***

1. INTRODUCTION

Recommendation system (RS) is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. Recommender systems typically produce recommendations in one of two ways – through collaborative filtering or through content-based filtering (also known as personality-based approach). Collaborative filtering approaches build a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties. These approaches are often combined as ***Hybrid Recommender*** System [1]. Different techniques have been developed over time to give accurate recommendations. As E- commerce is getting bigger, people are moving from retail shops to online store. On E-commerce, availability of numerous options makes the finding of most suitable item a hefty task, so RS makes this task easy by finding behaviour pattern from past history or user input. There are two major issues in existing recommendation system one is cold start and another one is accuracy [2].

In proposed model, we tried to overcome cold start problem by taking some inputs from user while creating account and accuracy by implementing different algorithms. If recommendations given are varying too much from the user's likes and tastes, he/she may simply stop using the system. So, in order to build trust, recommendations need to be personalized. Demographic recommendations are a good way of giving personalized predictions [3]. Filtering the results using collaborative approach leads to a better recommendation output. Recommendations suited to the user's age, region and Interests can be made more personalized. The cold start problem is a major issue in many recommendation systems. In such a scenario, the system is unable to give appropriate predictions until it has a better idea about the user's preferences. Demographic recommendations could help alleviate this problem to some extent, if not entirely in case of a newly added user. A user always would like to stay abreast of their liked category or liked author's books. The traditional filtering techniques may not always be able to keep a user updated about the recent trends in books.

There are three types of user, firstly, the new users. To deal with the cold start issue, we are beginning by asking users about categories (e.g. Suspense and thriller, romance etc.) and writers they are interested in. Based on these criterions, recommendations are being made. Along with this, a parallel approach is followed where we find users with similar interests and a bigger and more accurate set of recommendation is returned based on the rating profile.

The second types of user are the ones who don't prefer to rate. This case may lead to a failure of rating-based system. To tackle this issue, we are making recommendations based on their past orders. The third set of users consists of those who give feedback (Ratings). We are using 4 algorithms namely SVD, KNN, RBM, Hybrid for more options for user.

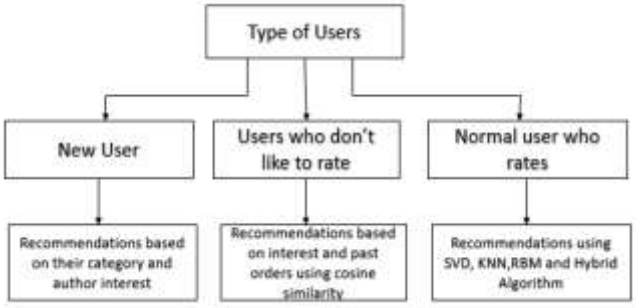

Figure 1: Types of users

To summarize the underlying approach, we are using hybrid model to provide personalized recommendations to individual user. This system is hybrid of content based as well as collaborative approach of recommender system. We are showing more accurate and more options that will increase the user experience and will raise the possibility of buying books.

2. RELATED WORK

Recommendation systems with strong algorithms are at the core of today's most successful online companies such as Amazon, Google, Netflix and Spotify. By endlessly recommending new products that suit their customer's tastes, these companies provide a personalized, attentive experience across their brand platforms, effectively securing customer loyalty.

Amazon, a popular e-commerce platform initially started its business with e-book store. It uses topic diversification algorithms to improve its recommendation [4, 5]. The system uses collaborative filtering method to overcome scalability issue by generating a table of similar items offline through the use of item-to-item matrix. The system then recommends other products which are similar online according to the users' purchase history. On the other hand, content-based techniques match content resources to user characteristics. Content-based filtering techniques normally base their predictions on user's information, and they ignore contributions from other users as with the case of collaborative techniques [6, 7]. Amazon fails to handle cold start problem.

NETFLIX provides a subscription service model that offers personalized recommendations to help us find shows and movies of our interest. To do this, they have created a proprietary, complex recommendations system. Netflix offer thousands of movies and shows. With over 7,000 movies and shows in the Netflix catalogue, it is nearly impossible for users to find movies they'll like on their own. Netflix uses the *personalized method* where movies are suggested to the users who are most likely to enjoy them based on a metric like major actors or genre. Machine learning is necessary for this method because it uses user data to make informed suggestions. This way Netflix methodology accounts for the diversity in its audiences and its very large catalogue.

2.1 CHALLENGES IN EXISTING SYSTEM

Most recommendation problems rely on the rating structure. In its most common formulations, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. This estimation is usually based on the ratings given by this user to other items. The engine in such software gives advice about what we might enjoy listening to or watching or reading next, based on user's history of liking content.

Problems in recommender systems,

The identified real-life problems which needs to be addressed in recommendation systems through the literature survey:

A. Cold Start Problem: Cold-start problem presents a collective issue of new item and new user to recommender systems. A new item can't be recommended initially when it is introduced to a content-based system with no ratings. For instance, MovieLens (movielens.org) cannot recommend new movies until these have got some initial ratings. The new-user problem is bit hard to handle because it is not possible to find similar users or to create a CB profile without previous preferences of a user.

B. Scalability of The Approach: One vital and foremost issue of Recommender systems today is the scalability of algorithms with large real-world datasets. It is becoming challenging to deal with huge and dynamic data sets produced by item-users interactions such as preferences, ratings and reviews.

C. Sparse, Missing, Erroneous and Malicious Data: Generally, majority of the users do not rate most of the items and consequently the ratings matrix becomes very sparse. Due to this, the data sparsity problem arises that declines the chances of finding a set of users with similar ratings. This is the most eminent drawback of the CF technique. This concern can be alleviated by using some additional domain information.

D. Big-data: Generally, a user can opt for an item of his interest from a recommendation list if the list reflects some diversity in the recommended items to some extent. Seamless recommendations for a restricted type of product have no value until or unless it is desired or explicitly described by the user with a narrow clique of preferences. In the initial stage, when the RS is used as a knowledge discovery tool, the users may wish to explore new and different options available.

3. HYBRID BOOK RECOMMENDATION SYSTEM

Our proposed system is Hybrid Recommendation system designed to overcome cold start issue and reduces dependency of rating-based system. It starts with general page where different books are shown to user based on their categories. User can search any books by its title or author name. While signing up, user is been asked to fill certain information like their category preferences, liked authors, location and age for finding similar users. Based on this information, books are being recommended which in turn help to overcome cold start problem. After signup, user can see their liked category and liked author's books in different titled catalogues.

If existing user has bought book(s) but not rated them yet, they will be shown books based on their purchase history as well as the information they've provided while signing up (categories and authors). Along with it, user will see random recommendations and predictions using different algorithms like SVD, KNN, RBM and Hybrid recommendations based on the books they've rated recently.

Furthermore, the system will track purchase history of users and that will reflect latest recommendations for book recommender system. So, with time user will be shown recommendation based in their most recent purchase history. User can see the details of book such as author, publication, rating, cover page, publication year. These books can be added to cart. The cart will show the books added to it and also other book recommendations in the "you may also like section". This section is being populated using Cosine similarity algorithm. We are using same algorithm (Cosine similarity) for providing with the search results.

Once a book is ordered, User can rate book based on their experiences in "My Orders" section of the application. Thus, the proposed system will ensure personalized recommendations eliminating drawbacks of rating-based approach.

Steps involved in the algorithm [8],

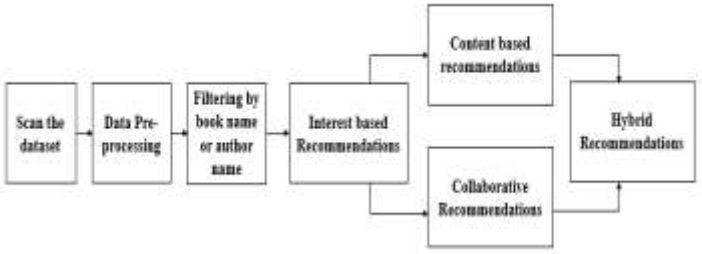

Figure 2: Steps involved for recommendations

Step 1: Scan the Books Dataset

In this step, application is scanning the entire storage server and simultaneously performing the data cleaning, which include removal of irrelevant data and keeping the relevant data for recommendations. This process reduces the data sparsity by eliminating missing, erroneous and malicious data from the working data set.

Step 2: Data Pre-processing

This step also works of the data correction part to ensure more accurate recommendations. According to our application, it includes the extraction of data that are needed for recommendations, which means extraction of only books having categories and users having demographic data.

Step 3: Filtering by book name or author name

This step revolves around proving users with the best and relevant search results. Factors like authors and book name can be searched and the result will return books using cosine similarity algorithm.

Step 4: Perform Content based Filtering

In this step we need to perform content-based filtering of books according to user preferences. For example, User1 clicked on book B1, assume that we have some related books B2, B3 and B4 in the dataset. Assume B2 is of different type, but B3 and B4 is of same type of book B1. Now we check the Meta data (category, author etc.) of the books B3 and B4, if it matches with book B1, then the system will recommend books B3 and B4 for the user. If user clicks on book B1, then the user will get books B3 and B4 as the recommended. Cosine similarity is used for finding similarity between two items while searching and showing similar items in cart. KNN is also used for finding similar users whose rating matched with User1's book rating history.

Step 5: Perform Collaborative Filtering

Here we consider the quality of the book content. In our example, recommending the books B3 and B4. This will perform based on the registered user's interest and rating. SVD is used for finding user's having similar book interest based on their ratings. RBM is used for giving accurate result even for books which does not been yet rated.

Step 6: Perform Interest Based Filtering

Here we consider interest of users, if user likes one specific genre or one author or any subject for example user likes some subjective books like on data science, so we recommend books according to their interest or author likes. It helps to overcome cold start problem.

Step 7: Final Recommendations

In the final recommendation, based on type of user, recommendations will differ like if user is new some interest-based result will be shown to user, if user don't like to rate then interest and similar books of past ordered books

will be shown to user else rating-based hybrid recommendations will be shown to user. [9]

4. ALGORITHMS

A. Cosine Similarity [10]

Cosine similarity (CS) measures the similarity between two vectors by calculating the cosine of the angle between them. It is widely used algorithm for finding similarity between two books or two users. The user Bob considers a three-star book, maybe different from what Alice considers a three-star book. Maybe Bob is just hesitant to rate things five stars unless they are truly amazing, while Alice tries to be nice and rates things five stars unless she really didn't like them. This is a real effect we will see, not just across individuals, but across different cultures too. So, adjusted cosine attempts to normalize these differences. We measure similarities based on the difference between a user's rating for an item, and their average rating for all items, instead of measuring similarities between people based on their raw rating values.

$$\text{CosSim}(x, y) = \frac{\Sigma_i \left((x_i - \bar{l})(y_i - \bar{l}) \right)}{\sqrt{\Sigma_i (x_i - \bar{l})^2} \sqrt{\Sigma_i (y_i - \bar{l})^2}}$$

So, if you look at this equation, if you replace X with X sub i minus X bar, and replaced Y with Y sub i and Y bar. X bar means the average of all user X's ratings, and Y bar means the average of all user Y's ratings. So, all that's different here from conventional cosine similarity is that we are looking at the variance from the mean of each users' ratings, and not just the raw rating itself. We can only get a meaningful average or a baseline of an individual's ratings if they have rated a lot of stuff for you to take the average in the first place. Imagine a book that most people love, like Alchemist, people who hate Alchemist are going to get a very strong similarity score from Pearson's similarity because they share opinions that are not mainstream.

Note that the only difference between this and adjusted cosine is whether we're talking about users or items. The surprise library we are using in this project refers to adjusted cosine as user-based person's similarity, because it's basically the same thing. Another way to measure similarity is the mean squared difference similarity metric. All of the items that two users have in common in their ratings, and compute the mean of the square differences between how each user rated each item. It's easy to compute since it doesn't involve angles and multidimensional space. You're just directly comparing how two people rated the same set of things. It's very much the same idea of how we measure mean absolute error, when measuring the accuracy of a recommender's system as a whole.

$$MSD(x, y) = \frac{\sum_{i \in I_{xy}} (x_i - y_i)^2}{|I_{xy}|}$$

$$MSDsim(x, y) = \frac{1}{MSD(x, y) + 1}$$

So, if we break down that top equation, it says that the mean squared difference, or MSD for short, between two users X and Y is given by the following. On the top of this fraction, we are summing up for every item i that users X and Y have both rated. The difference between the ratings from each user squared. We then divide by the number of items each user had in common that we summed across to get the average or mean. Now, the problem is that we have computed a metric of how different users X and Y are, and we want to measure how similar they are not how different they are. So, to do that, we just take the inverse of MSD dividing it by one, and we have to stick that plus one on the bottom in order to avoid dividing by zero in the case where these two users have identical rating behavior. You can, by the way, flip everything we just said to apply to items instead of users, so X and Y could refer to two different things instead of two different people, and then we'd be looking at the differences in ratings from the people these items have in common, instead of the items people have in common.

In our case, we have used cosine similarity in following scenarios:

1. Finding similar books of searched item. It is based on author and category.
2. Finding similar books of which have added in cart.
3. In KNN algorithm
4. Finding similar users by using demographic information.
   After finding similar users their past orders are being recommended to new user which reduces dependency on rating-based system.
5. Finding similar books based on Authors.

B. SVD (Singular Value Decomposition) [11]

Singular value decomposition (SVD) can be seen as a method for data reduction.

These are the basic ideas behind SVD: taking a high dimensional, highly variable set of data points and reducing it to a lower dimensional space that exposes the substructure of the original data more clearly and orders it from most variation to the least. Singular Value Decomposition is a matrix factorization technique which takes a rectangular matrix defined as A where A is an m x n matrix in which the m rows represents the users, and the n columns represents the items. The SVD theorem states,

Amxn = Umxm Smxn VTnxn .... [1]

Where, *UTU = Imxm*

VTV = Inxn

Where the columns of U are the left singular vectors; S (the same dimensions as A) has singular values and is diagonal; and VT has rows that are the right singular vectors. Calculating the SVD consists of finding the Eigenvalues and Eigenvectors of AA^T and A^TA. The Eigenvectors of A^TA make up the columns of V, the Eigenvectors of AA^T make up the columns of U. Also, the singular values in S are square roots of Eigenvalues from AA^T or A^TA. The singular values are the diagonal entries of the S matrix and are arranged in descending order. The singular values are always real numbers. If the matrix A is a real matrix, then U and V are also real.

Matrix S is a diagonal matrix having only r nonzero entries, which makes the effective dimensions of U, S and V matrices m × r, r ×r, and r × n, respectively. The diagonal entries ($s_1, s_2, \ldots, s_r$) of S have the property that si > 0 and $s_1 \geq s_2 \geq \ldots \geq s_r$.

In our proposed system, SVD is used for finding similar user based on their past ratings.

Recommendation Request (User = x, Item = y, Rating =?)

1. Find users who rated Item = y from the original matrix A.

2. Find most similar user to User = x among the users who rated Item = y using the reduced matrix U_k.

3. Get the rating of the most similar user to Item = y from the original matrix A and give it for the User = x, Item = y.

For the second part of the algorithm, if the User = x is an already existing user, it exists in the reduced matrix U_k as a row. If the User = x is a new user, before starting similarity checks, the user has to be projected from n dimensions to k dimensions. Let the ratings of the new user vector isN_u $(1xn)$. The projection P to the reduced matrix U_k is made by the formula [1]:

$$P = N_u \times V_k \times S_k^{-1}$$

In the experimental part of this study, the Euclidian distance algorithm explained above is used for the similarity check of the users. The bottleneck in this technique is the search for similar users among a large user population. SVD is the powerful algorithm in recommendation system. It gives more accurate result. SVD is used for finding collaborative recommendations. RBM Works good with large dataset so to overcome the data sparsity issue in RBM we have used both RBM and SVD as collaborative algorithm for improving accuracy and giving more options to users.

C. KNN (k-Nearest Neighbours) [12]

To implement an item based collaborative filtering, KNN is a perfect go-to model and also a very good baseline for recommender system development is a **non-parametric, lazy** learning method.

KNN does not make any assumptions on the underlying data distribution but it relies on **item feature similarity**. When KNN makes inference about a book, KNN will calculate the "distance" between the target book and every other book in its database, then it ranks its distances and returns the top K nearest neighbor books as the most similar book recommendations.

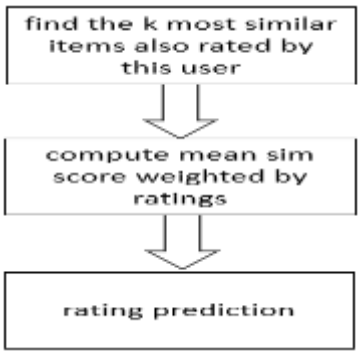

Figure 3: KNN Approach

```
# Build up similarity scores between this item and everything the user rated

neighbors = []

for rating in self.trainset.ur[u]:

categorySimilarity = self.similarities[i,rating[0]]

neighbors.append( (categorySimilarity, rating[1]) )

# Extract the top-K most-similar ratings

k_neighbors = heapq.nlargest(self.k, neighbors, key=lambda t: t[0])

# Compute average sim score of K neighbors weighted by user ratings

simTotal = weightedSum = 0

for (simScore, rating) in k_neighbors:

if (simScore > 0):

simTotal += simScore

weightedSum += simScore * rating

if (simTotal == 0):
```

```
raise PredictionImpossible('No neighbors')

predictedRating = weightedSum / simTotal

return predictedRating
```

KNN is used in proposed system for finding similar books based on their category and ratings. KNN is used for finding content-based recommendations.

D. RBM (Restricted Boltzmann Machines) [13]

Most of the existing approaches to collaborative filtering cannot handle very large data sets. The oldest algorithm of neural networks in recommender systems is the Restricted Boltzmann Machine or RBM for short. RBM's are one of the simplest neural networks consist of only two layers, a visible layer and a hidden layer.

We train it by feeding our training data into the visible layer in a forward pass, and training weights and biases between them during back propagation. An activation function such as ReLU is used to produce the output from each hidden neuron.

They are restricted because neurons in the same layer can't communicate with each other directly. There are only connections between the two different layers. RBM's get trained by doing a forward pass, which we just described, and then a backward pass, where the inputs get reconstructed. We do this iteratively over many epochs, just like when we train a deep neural network, until it converges on a set of weights and biases that minimizes the error.

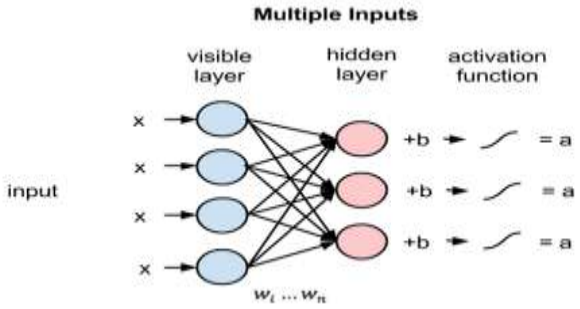

Figure 4: RBM work flow

Let's take a closer look at that backward pass. During the backward pass, we are trying to reconstruct the original input by feeding back the output of the forward pass back through the hidden layer, and seeing what values we end up with out of the visible layer. Since those weights are initially random, there can be a big difference between the inputs we started with and the ones we reconstruct. In the process, we end up with another set of bias terms, this time on the visible layer. So, we share weights between both the forward and backward passes, but we have two sets of biases. The hidden bias that's used in the forward pass, and the visible bias used in this backward pass. We can then measure the error we end up with and use that information to adjust the weights a little bit during the next iteration to try and minimize that error.

Adapting an RBM for book recommendations given five-star ratings data requires a few twists to the generic RBM architecture.

The general idea is to use each individual user in our training data as a set of inputs into our RBM to help train it. So, we process each user as part of a batch during training, looking at their ratings for every book they rated. So, our visible nodes represent ratings for a given user on every book, and we're trying to learn weights and biases to let us reconstruct ratings for user/book pairs we don't know yet.

First of all, our visible units aren't just simple nodes taking in a single input. Ratings are really categorical data, so we actually want to treat each individual rating as five nodes, one for each possible rating value. So, let's say the first rating we have in our training data is a five-star rating which will be represented as four nodes with a value of zero and one with a value of one, as represented here. Then we have a couple of ratings that are missing for user/item pairs that are unknown and need to be predicted. Then we have a three-star rating, represented like this with a one in the third slot. When we're done training the RBM, we'll have a set of weights and biases that should allow us to reconstruct ratings for any user. So, it is used to predict ratings for a new user, we just run it once again using the known ratings of the user we're interested in. We run those through in the forward pass, and then back again in the backward pass, to end up with reconstructed rating values for that user. We can then run softmax on each group of five rating values to translate the output back into a five-star rating for every item. But again, the big problem is that the data we have is sparse.

If we are training an RBM on every possible combination of users and books, most of that data will be missing, because most books have not been rated at all by a specific user. We want to predict user ratings for every book though, so we need to leave space for all of them. That means if we have N books, we end up with N time five visible nodes, and for any given user, most of them are undefined and empty. We deal with this by excluding any missing ratings from processing while we're training the RBM.

This is kind of a tricky thing to do, because most frameworks built for Deep Learning such as TensorFlow assume you want to process everything in parallel, all the time. Sparse data isn't something they were really built for originally, but there are ways to trick it into doing what we want. But, notice that we've only drawn lines between visible units that actually have known ratings data in them, and the hidden layer. So, as we're training our RBM with a given user's known ratings, we only attempt to learn the weights and biases used for the books that user actually rated. As, we

iterate through training on all of the other users, we fill in the other weights and biases as we go. We are using tensorflow for deep neural networks with sparse data for the sake of completeness.

The other twist is how to best train an RBM that contains huge amounts of sparse data. Gradient descent needs a very efficient expectation function to optimize on, and for recommender systems this function is called contrastive divergence. The basic idea is that its samples probability distributions during training using something called a Gibbs sampler. We only train it on the ratings that actually exist, but re-use the resulting weights and biases across other users to fill in the missing ratings.

RBM is used for finding collaborative recommendations.

5. EXPERIMENT

We have conducted a set of experiments to examine the effectiveness of our proposed recommender system in terms of accuracy of books being recommended to the user. We have created a Personalized recommendation system-based on user interest, past history and ratings. Recommendation engine is running on book crossing database. Recommendation Engine consists of cosine similarity, KNN, RBM, SVD and Hybrid. Personalized recommendations are shown on web application. There are two types of technologies used in experiment.

Client-side technology: Angular 7, HTML, CSS

Server-side technology: python 3.6

System configuration: 16 GB RAM, Intel core i3 CPU 2 GHZ

Fig. 5 Architecture of recommendation system

Cosine similarity is used for showing similar books which are searched by user and/or added into cart by user. Apart from searching it is also being used for finding similar users based on demographics. KNN, RBM, SVD and hybrid are used to show rating-based recommendations.

This experiment was firstly performed on a small section of book crossing dataset consisting of 675 users and 10,000 books of all category. We have used different algorithms and combination as hybrid to ensure best outcome.

Following are some parameters along with their outcome on which the system is tested.

	Random	RBM	KNN	SVD	Hybrid
RMSE	1.8102	1.4391	1.4346	1.4373	1.4365
MAE	1.4932	1.2269	1.2152	1.2235	1.2233
HR	0.0001	1.0000	1.0001	1.0000	1.0000
CHR	0.0001	2.0000	2.0001	2.0000	2.0000
ARHR	0.0001	2.0000	2.0001	2.0000	2.0000
Coverage	1.0000	1.0000	0.4065	0.0093	0.0000
Diversity	0.4896	1.0000	0.4368	0.9754	1.0000
Novelty	1.3700	6.0000	1.1966	9.3053	7.0010

Table 1: Performance Metrics

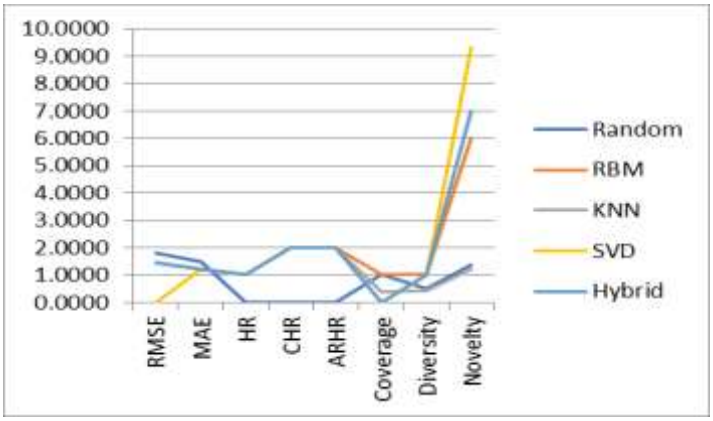

Chart 1: Performance of different algorithms

RMSE (Root Mean Squared Error): Root Mean Square Error (RMSE) puts more emphasis on larger absolute error and the lower the RMSE is, the better the recommendation accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2}$$

Where P is the predicted rating for user $P_{u,i}$ is the predicted rating for user u on item i, $r_{u,i}$, i is the actual rating and N is the total number of ratings on the item set. RMSE is comparatively lowest in RBM hence accuracy is more.

MAE (Mean Absolute Error): MAE is the most popular and commonly used; it is a measure of deviation of recommendation from user's specific value. It is computed as follows

$$MAE = \frac{1}{N}\sum_{u,i}|p_{u,i} - r_{u,i}|$$

Where P is the predicted rating for user $P_{u,i}$ is the predicted rating for user u on item i, $r_{u,i}$, i is the actual rating and N is the total number of ratings on the item set. The lower the

MAE, the more accurately the recommendation engine predicts user ratings. MAE is comparatively lowest in Hybrid; hence accuracy is more.

HR (Hit Rate): Hit rate means how often we are able to recommend a left-out rating. The whole hit rate of the system is the count of hits, divided by the test user count. It measures how often we are able to recommend a removed rating, higher is better.

A very low hit rate simply means we do not have enough data to work with. As we have less data hit rate is equal for all algorithms.

CHR (Cumulative Hit Rate): Cumulative hit rate, confined to ratings above a certain threshold. Because we care about higher ratings, we can ignore the predicted ratings lower than 4, to compute hit rate for the ratings > = 4. Higher is better. As we have less data CHR is equal for all algorithms.

ARHR: Average Reciprocal Hit Rank - Hit rate that takes the ranking into account. Commonly used metric for ranking evaluation of Top-N recommender systems, that only takes into account where the first relevant result occurs. We get more credit for recommending an item in which user rated on the top of the rank than on the bottom of the rank. Higher is better. As we have less data ARHR is equal for all algorithms.

Coverage: Ratio of users for whom recommendations above a certain threshold exist. We try to find good recommendations in our top n list by setting a threshold which allows only good recommendations for each user in the top-n list and summing them and dividing by the number of users. Higher is better. As we have a smaller number of users, coverage is comparatively good in RBM.

Diversity: 1-S, where S is the average similarity score between every possible pair of recommendations for a given user. This is calculated by first finding similarity for a set of users and then subtracting it from 1 to find the diversity. We calculate similarity for all the combinations of users and sum them and then divide by the number of combinations. To calculate similarity, we need the inner ids of the users to find the similarity between the users, as the surprise library in python uses them for indexing similarity scores. Higher means more diverse. Hybrid system gives more diverse results.

Novelty: Novelty determines how unknown recommended items are to a user. It is average popularity rank of recommended items. Higher means more novel. Hybrid system gives more novel results.

6. CONCLUSION

Recommendation system is widely used from the last decades. Book recommendation system is recommending books to the buyers that suits according to their interest and stores recommendations in the buyer's web profile. This system will ensure highly personalized and accurate recommendations eliminating the drawbacks of rating-based recommendation systems while reducing the cold start problem alongside. Apart from just the traditional Collaborative and Content based filtering techniques, many modern techniques are being exploited in this application. The hybrid algorithm we are using is a combination of content based and collaborative approaches. Demographic filtering which helps give more personalized recommendations is also used. The system is using multiple algorithms as explained in the paper to enhance the quality and accuracy of personalized results and recommendations.

REFERENCES

[1] Robin Burke, "Hybrid Web Recommender Systems", January 2007.

[2] Salil Kanetkar, Akshay Nayak, Sridhar Swamy, Gresha Bhatia, "Web-based Personalized Hybrid Book Recommendation System" IEEE International Conference on Advances in Engineering & Technology Research (ICAETR-2014), August 01-02, 2014, Dr. Virendra Swarup Group of Institutions, Unnao, India.

[3] Jihane KARIM, "Hybrid System for Personalized Recommendations" 978-1-4799-2393-9/14 ©2014 IEEE

[4] F.O.Isinkaye, Y.O.Folajimi, B.A.Ojokoh "Recommendation systems: Principles, methods and evaluation" Eg yptian Informatics Journal Volume 16, Issue 3, November 2015, Pages 261-273

[5] Ziegler CN, McNee SM, Konstan JA, Lausen G. Improving recommendation lists through topic diversification. In:Proceedings of the 14th international conference on World WideWeb; 2005. p. 22–32.

[6] Min SH, Han I. Detection of the customer time-variant patternfor improving recommender system. Exp Syst Applicat2010;37(4):2911–22.

[7] Celma O, Serra X. FOAFing the Music: bridging the semantic gapin music recommendation. Web Semant: Sci Serv Agents WorldWide Web 2008; 16(4):250–6.

[8] Manisha Chandaka, Sheetal Giraseb, Debajyoti Mukhopadhyay "Introducing Hybrid Technique for Optimization of Book Recommender System"

[9] Ms. Praveena Mathew, Ms. Bincy Kuriakose, Mr.Vinayak Hegde "Book Recommendation System through Content Based and Collaborative Filtering Method"

[10] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi, "Semantic Cosine Similarity"

[11] Osman Nurg Osmanli,"A singular value decomposition approach for recommendation system", Thesis

[12] Hua-Ming Wang, Ge Yu," Personalized recommendation system K- neighbor algorithm optimization" International Conference on Information Technologies in Education and Learning (ICITEL 2015)

[13] Ruslan Salakhutdinov,Andriy Mnih, Geoffrey Hinton, "Restricted Boltzmann Machines for Collaborative Filtering" University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada