A Comparative Study of Algorithms for Recommender Systems

[1] K.Mounika Krishna [2]A.Mary Sowjanya

[1]M.Tech, [2]Assistant Professor

[2]Dept of CSSE, Andhra University, College of Engineering(A),

Visakhapatnam, AP, India.

ABSTRACT: Recommender Systems have become a very useful tool for a large variety of domains. Researchers have been attempting to improve their algorithms in order to issue better predictions to the users. However, one of the current challenges in the area refers to how to properly evaluate the predictions generated by a recommender system. In the extent of offline evaluations, some traditional concepts of evaluation have been explored, such as accuracy, Root Mean Square Error and Mean Absolute Error for top-N recommendations. In recent years, more research have proposed some new concepts such as novelty, diversity and coverage. These concepts have been addressed with the goal to satisfy the users' requirements. Numerous definitions and metrics have been proposed in previous work. On the absence of a specific summarization on evaluations of recommendation combining traditional metrics and recent progresses, this paper surveys and organizes the main research that present definitions about concepts and propose metrics or strategies to evaluate recommendations. In addition, this survey also settles the relationship between the concepts, categorizes them according to their objectives and suggests potential future topics on user satisfaction. We compare five experimental methodologies, broadly covering the variants reported in the literature. In our experiments with three state-of-the-art recommenders, four of the evaluation methodologies are consistent with each other and differ from error metrics, in terms of the comparative recommenders' performance measurements. The other procedure aligns with RMSE, but shows a heavy bias towards known relevant items, considerably overestimating performance.

KEYWORDS*:* RMSE, MAE, *Matrix factorization*, Recommender System, Cosine Similarity Metric, ARHR.

I. INTRODUCTION

recommendation system is a type of information filtering system which is used to predict the "rating" or "preference" user would give to an item. A recommendation system collect data about the user's preferences either implicitly or explicitly on different items like movies, shopping, tourism, TV etc. An implicit acquisition in the development of movie recommendation system uses the user's behavior while watching the movies. On the other hand, a explicit acquisition in the development of movie recommendation system uses the user's previous ratings or history. Collaborative filtering is the technique to filter or calculate the items through the sentiments of other users. Collaborative filtering first collect the movie ratings or preference given by different users and then suggest movies to the different user based on similar tastes and interests in the past. The other supporting technique that are used in the development of recommendation system. The evaluation of Recommender Systems (RS) has been an explicit object of study in the field since its earliest days, and is still an area of active research, where open questions remain [3,10]. The dominant evaluation methodologies in off-line experimentation have been traditionally error-based. There is however an increasing realization that the quality of the ranking of recommended items can be more important in practice (in terms of the effective utility for users) than the accuracy in predicting specific preference values. As a result, precision-oriented metrics are being increasingly often considered in the field. Yet there is considerable divergence in the way these metrics are being applied by different authors, as a consequence of which, the results reported in different studies are difficult to put in context and compare. In the typical formulation of the recommendation problem, user interests for items are represented as numeric ratings, some of which are known. Based on this, the task of a recommendation algorithm consists of predicting unknown ratings based on the known ones and, in some methods, some additional available information about items and users.

With this formulation, the accuracy of recommendations has been evaluated by measuring the error between predicted and known ratings, by metrics such as the Mean Average Error (MAE), and the Root Mean Squared Error (RMSE). Although dominant in the literature, some authors have argued this evaluation methodology is detrimental to the field since the recommendations obtained in this way are not the most useful for users [9]. Acknowledging this, recent works evaluate top-N ranked recommendation lists with precision-based metrics [2,8,5,1], drawing from well-studied methodologies in the Information Retrieval (IR) field. Precision-oriented metrics measure the amount of relevant and non-relevant retrieved items. A solid body of metrics, methodologies, and datasets has been developed over the years in the IR field to measure this in different ways. There is however a major difference between the RS and IR experimental settings. In ad-hoc IR experiments, relevance knowledge is typically assumed to be (not far from) complete –mainly because in the presence of a search query, relevance is simplified to be a user-independent property. However, in RS it is impractical to gather complete preference information for each user in the system. In datasets containing thousands of users and items, only a fraction of the items that users like is generally known. The unknown rest are, for evaluation purposes, assumed to be non-relevant. This is a source of –potentially strong– bias in the measurements depending on how unknown

relevance is handled. To the best of our knowledge, a thorough analysis of precision oriented methodologies is still missing in the field. In fact, the detailed alternatives in the evaluation procedures are not clearly identified, and the extent to which they are equivalent or provide comparable results has not been studied in depth. Reported results differ in several orders of magnitude for the same metric on similar datasets and similar algorithms in different studies. In this paper, we present a general methodological framework for evaluating recommendation lists, covering different evaluation approaches, most of them documented in the literature, and some included here for the sake of the systematic consideration of alternatives. The considered approaches essentially differ from each other in the amount of unknown relevance that is added to the test set. We use Precision, Recall, and normalized Discounted Cumulative Gain (nDCG) [10] as representative precision oriented metrics. The purpose of our study is to assess the differences and potential equivalences resulting from the methodological variants, and to what extent precision-based results relate or differ from error-based metrics. We found that four of the methodologies are consistent with each other in terms of the observed recommenders' performance trend. The other methodology considerably overestimates performance, and suffers from a strong bias towards known relevant items.

II.RELATED WORKS

Predicting ratings and creating personalized recommendations for products like books, songs or movies online came a long way from Information Lense, the recommendation system using social filtering created by Malone, Grant, Turbak, Brobst, and Cohen more than 20 years ago. Today recommender systems are an accepted technology used by market leaders in several industries(e.g., by Amazon1, Netix2 and Pandora3). Recommender systems apply statistical and knowledge discovery techniques to the problem of making product recommendations based on previously recorded data (Sarwar, Karypis, Konstan, and Riedl). Such recommendations can help to improve the conversion rate by helping the customer to and products she/he wants to buy faster, promote cross-selling by suggesting additional products and can improve customer loyalty through creating a value-added relationship (Schafer, Konstan, and Riedl). The importance and the economic impact of research in this _eld is reected by the Netix Prize4, a challenge to improve the predictions of Netix's movie recommender system by more than 10% in terms of the root mean square error. The grand price of 1 million dollar was awarded in September 2009 to the Belcore Pragmatic Chaos team. Ansari, Essegaier, and Kohli categorizes recommender systems into content-based approaches and collaborative filtering. Content-based approaches are based on the idea that if we can elicit the preference structure of a customer (user) concerning product (item) attributes then we can recommend items which rank high for the user's most desirable attributes. Typically, the preference structure can be elicited by analysing which items the user prefers.For example, for movies the Internet Movie Database5 contains a wide range of attributes to describe movies including genre, director, write, cast, storyline, etc. For music, Pandora , a personalized online radio station, creates a stream of music via content-based recommendations based on a system of hundreds of attributes to describe the essence of music at the fundamental level including rhythm, feel, instruments and many more (John 2016).The idea is that given rating data by many users for many items (e.g.,1 to 5 stars for movies elicited directly from the users), one can predict a user's rating for an item not known to her or him (see, e.g., Goldberg, Nichols, Oki, and Terry) or create for a user a so called top-N lists of recommended items (see, e.g., Deshpande and Karypis). The premise is that users who agreed on the rating for some items typically also agree on the rating for other items.

1. EXPLICIT DATA

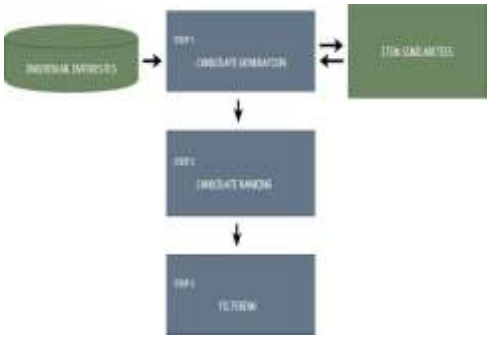
- It is collected with the help of feedback or a survey or ratings.
- Not everyone bothers to fill forms or take surveys. Moreover, One person's point of view may vary from another person's point of view depending on social-economic-political and geographical differences.

2. IMPLICIT DATA

- Understanding users through their 'click-data' or 'stream-data'
- Prone to frauds
- Amazon has so much implicit data that it doesn't need better algorithms! Even simple algorithms work like charm when we have a huge amount of data.

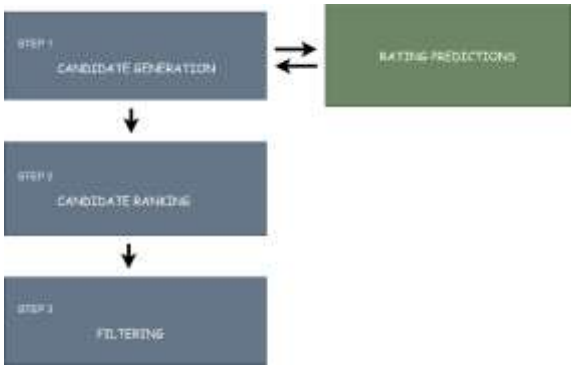
TOP-N RECOMMENDATION SYSTEM

"Top N Recommendation System". Top N Recommendation System tries to give user 'N' number of recommendations to the users that it has predicted using its algorithms.



TOP N RECOMMENDATION SYSTEM

STEP 1: If current user buys an 'item', recommendations are generated with the help of actual historical data of other people who bought 'later the same 'item' was bought by them.
STEP 2: Recommended items are ranked and sorted.
STEP 3: Already appeared recommendations and offensive ones are removed and they are shortlisted to 'N' number of items.
The above architecture is just one of the many that can be built for our recommendation System. For example, another architecture can be as shown below



IV. METHODOLOGY

Recommender Systems can be personalized, non-personalized, attribute based, item-to-item correlation, and people-to-people correlation. Different algorithms and techniques are used by recommender systems to generate recommendations. The most popular ones are collaborative filtering, content-based filtering and hybrid filtering.

CONTENT-BASED RECOMMENDATIONS

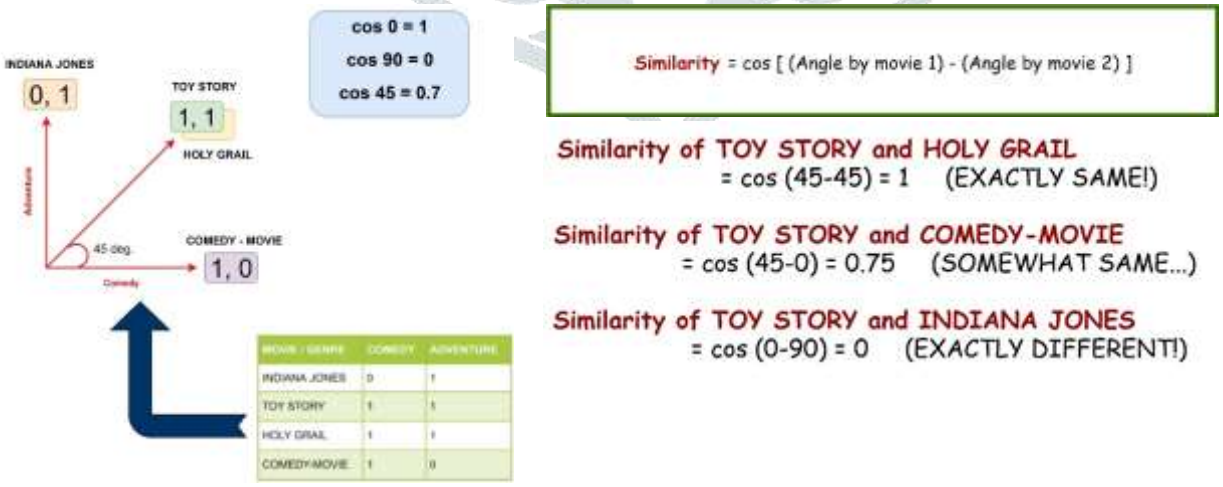
Content-based recommendations are the **simplest of all approaches**. The main idea is to recommend based on the properties of items instead of using aggregate user behavior .

COSINE SIMILARITY METRIC

To know how similar two items are with metrics such as **cosine similarity metrics.**

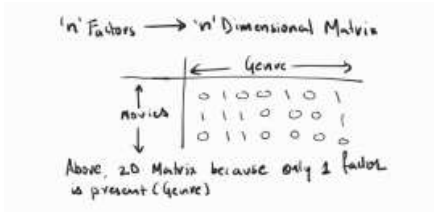
- A cosine similarity metric is ideal for content-based recommendations
- It is used to find similarity of any given pair of movies

To represent a movie-genre matrix in a two-dimensional space as shown below. The x-axis represents the discrete value for comedy attribute and the y-axis represents adventure attribute. Put discrete value '1' for comedy attribute if is movie is full of comedy and '0' if the movie isn't that funny; Same is done to adventure attribute.



Here, notice that the formula used for cosine similarity is a good but finding the angles from the kind of data is difficult. To solve this problem with the same approach but completely different formula! The below-mentioned formula can be used instead

$$\text{Cosine Similarity } (x,y) = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$



With the increase in the number of factors depending on which we are computing similarities, our matrix dimensions increase or decrease 2D Matrix Example Cosine similarity can be implemented to genre very easily. The code snippet below does the same.

K Nearest Neighbours [KNN] Algorithm: It selecting 'N' number of things that are close to the things you are interested in.

- **Step 1:** Find similarity scores between the movie and all the movies user has rated
- **Step 2:** Sort and choose top 40 nearest neighbours of the movie
- **Step 3:** Take the weighted average and predict the ratings

NEIGHBORHOOD BASED COLLABORATIVE FILTERING

Neighborhood Based Collaborative Filtering leverages the behavior of other users to know what our user might enjoy. It may find people similar to user and recommend stuff they liked or recommend stuff that other people bought after buying what user has bought.

PRODUCING SIMILAR ITEMS/PEOPLE

To measure the similarity between things or similarity between people by doing the following

- Using similarity metrics (Cosine Similarity)
- Sparsity — big challenge on measuring similarities based behaviour data.
- Quality and quantity of data are **more important** than the algorithm chose

There are many different ways to compute similarity. Some of them are listed below

- **Cosine Similarity Metric**
- **Adjusted Cosine Similarity:** Same as Pearson Similarity
- **Pearson Similarity:** Same as Adjusted Cosine Similarity
- **Spearman Rank Correlation:** Same as Pearson but with ranks
- **MSD Similarity:** Easier to understand but low performance
- **Jaccard Similarity:** Good for implicit data.

USER BASED COLLABORATIVE FILTERING

KNN — K Nearest Neighbours algorithm is used.

Even if Bob loved Star Wars and Ann hated it, In **SPARSE DATA SITUATION,** Both 100% similar

- Sparse data leads to problems with collaborative filtering in general

- Avoided weird stuffs, put thresholds and even pre-process data

SORTING

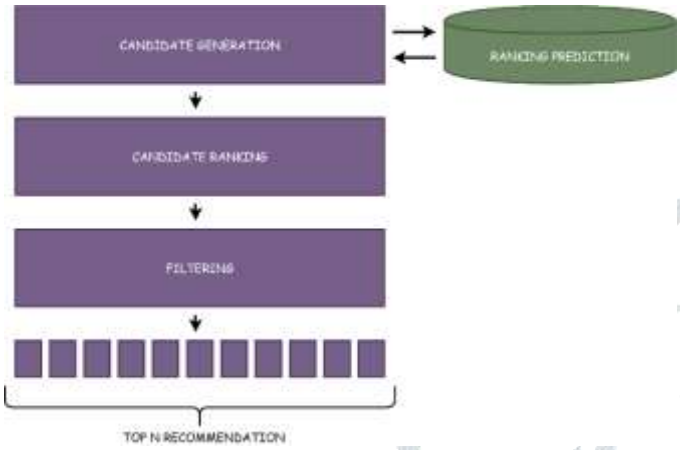
In this, try to choose and arrange our recommendations with best predictions

- Taking ratings under consideration
- Recommend similar things they love. Not similar things they hate. So, better normalize it
- Strengthen the relationship if 'items' appear in more than one neighbour

This step involves processes such as removing already seen items or the items that are offensive.
All the steps involved in this process is shown in block diagram

KNN Recommendations

The architecture below showcases collaborative filtering applied to the recommendation system for "Rating Predictions".



MATRIX FACTORISATION

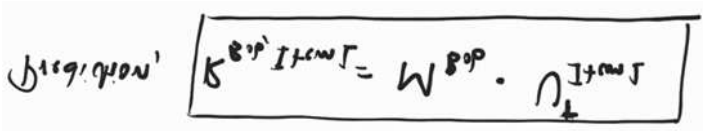
- It has many sub categories of techniques.
- Manages to find broader feature of users/items on their own . Math doesn't know what to call newly found features which are simply described by matrices.

Describe users and items as combinations of different amounts of each other.
For example, Bob = 50% Action +20% Comedy

SINGULAR VALUE DECOMPOSITION (SVD)

A way of computing M, Sigma, U-Transpose all at once efficiently and get all three factors in one shot

- Very accurate results
- Used widely during Netflix prize competition



Actually, predicting ratings and not computing them directly which is what SVD does. I am **not** doing real SVD Recommendations because SVD can't perform on missing data. Hence, it is **'SVD-Inspired-algorithm'** not SVD itself. Winner of Netflix prize was a variation of SVD called **SVD++**

SVD v/s SVD++

- Actual loss function used during SGD
- In SVD++, merely rating item at all is an indication of some sort of interest in the item.
- SVD++ outperformed SVD

THE BELOWLISTED ONES ARE JUST SOME OF THE ALGORITHMS THAT WERE DEVELOPED

I. Factorization Matrices

- Well suited for predicting ratings/clicks in Recommendation System
- More '**General Purposed'**
- **Handles sparse data with ease**. Unlike shoehorning itself into problem like SVD
- Available in Amazon Sage

II. TimeSVD++

- Used for predicting next item in series of events.

III. Factorized Personalized Markov Chains

- Same as above. Used for predicting next item in series of events

CONTENT-BASED RECOMMENDATIONS

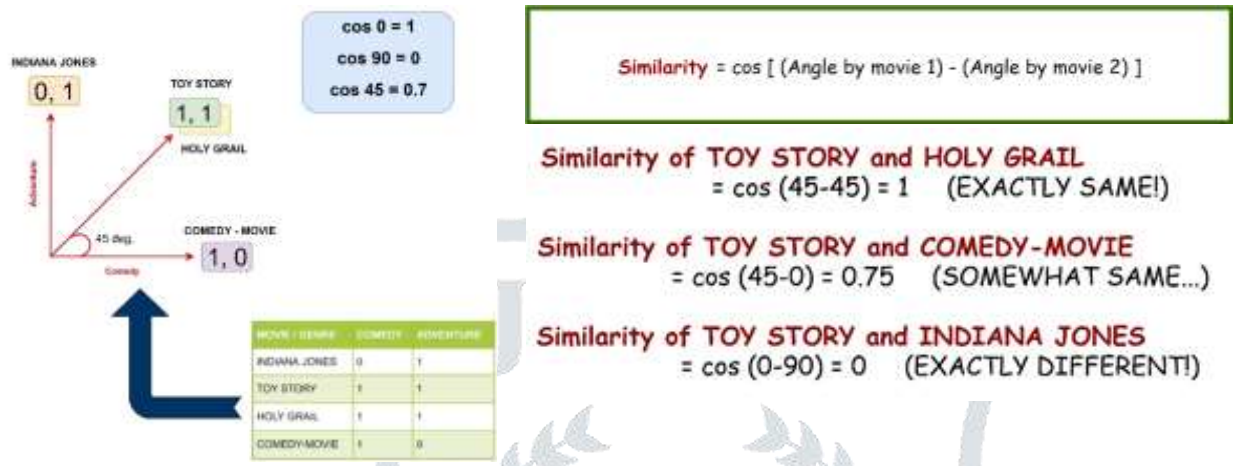
Content-based recommendations are the **simplest of all approaches**. The main idea is to recommend based on the properties of items instead of using aggregate user behavior .

COSINE SIMILARITY METRIC

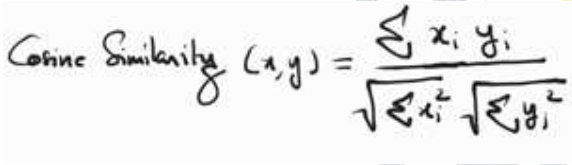
To know how similar two items are with metrics such as **cosine similarity metrics.**

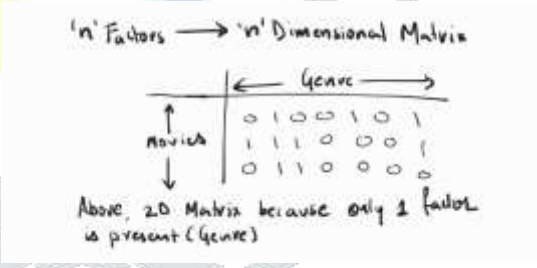
- A cosine similarity metric is ideal for content-based recommendations
- It is used to find similarity of any given pair of movies

To represent a movie-genre matrix in a two-dimensional space as shown below. The x-axis represents the discrete value for comedy attribute and the y-axis represents adventure attribute. Put discrete value '1' for comedy attribute if is movie is full of comedy and '0' if the movie isn't that funny; Same is done to adventure attribute.



Here, notice that the formula used for cosine similarity is a good but finding the angles from the kind of data is difficult. To solve this problem with the same approach but completely different formula! The below-mentioned formula can be used instead





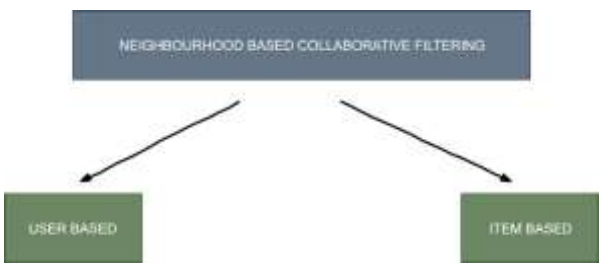
With the increase in the number of factors depending on which we are computing similarities, our matrix dimensions increase or decrease 2D Matrix Example Cosine similarity can be implemented to genre very easily. The code snippet below does the same.

K Nearest Neighbours [KNN] Algorithm: It selecting 'N' number of things that are close to the things you are interested in.

- **Step 1:** Find similarity scores between the movie and all the movies user has rated
- **Step 2:** Sort and choose top 40 nearest neighbours of the movie
- **Step 3:** Take the weighted average and predict the ratings

NEIGHBORHOOD BASED COLLABORATIVE FILTERING

Neighborhood Based Collaborative Filtering leverages the behavior of other users to know what our user might enjoy. It may find people similar to user and recommend stuff they liked or recommend stuff that other people bought after buying what user has bought.



PRODUCING SIMILAR ITEMS/PEOPLE

To measure the similarity between things or similarity between people by doing the following

- Using similarity metrics (Cosine Similarity)
- Sparsity — Big challenge on measuring similarities based behaviour data.
- Quality and quantity of data are **more important** than the algorithm chose

OTHER WAYS TO COMPUTE SIMILARITY

There are many different ways to compute similarity. Some of them are listed below

- **Cosine Similarity Metric**
- **Adjusted Cosine Similarity:** Same as Pearson Similarity
- **Pearson Similarity:** Same as Adjusted Cosine Similarity
- **Spearman Rank Correlation:** Same as Pearson but with ranks
- **MSD Similarity:** Easier to understand but low performance
- **Jaccard Similarity:** Good for implicit data.

V. EVALUATION

There are two different important sub-processes for evaluation —

1. **Evaluating Offline**
2. **Online A/B Testing**

There is a real challenge where often accuracy metrics tells which algorithm is great only to have it do HORRIBLE in online A/B Test. YouTube studies this and calls it "THE SURROGATE PROBLEM" online A/B tests are the only thing can be used for Recommendation System.

Evaluate recommender systems offline

Because they are the ONLY indicators you can look at while developing recommendation systems. Always preferring to go with online metrics to collect user behaviour and scoring the system is expensive and time-consuming. Moreover, when continuous feedback are asked from users, they might become more hesitant to use our platform and not use it at all. Good accuracies in offline metrics followed by good online A/B scores

THE TWO DIFFERENT WAYS TO TEST OFFLINE

1. **Train-Test Split**
2. **K-Fold Cross-Validation**

Accuracies in the above methods depend on historical data and try to predict what actual users have already seen. If the data collected is too old, however high the accuracies maybe, they won't mean anything as user interests a year back will not be as same as user interests a year from now

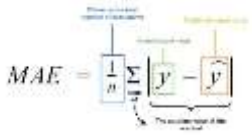
DIFFERENT OFFLINE METRICS

The different offline metrics and other measures that define our Recommendation System are mentioned below.

1. Mean Absolute Error (MAE)
2. Root Mean Square Error (RMSE)
3. Hit Rate (HR)
4. Average Reciprocal Hit Rate (ARHR)
5. Cumulative Hit Rate (cHR)
6. Rating Hit Rate (rHR)
7. Coverage
8. Diversity
9. Novelty
10. Churn
11. Responsiveness

1. MEAN ABSOLUTE ERROR (MAE)

It is the difference between the actual value (rating) and the predicted output value. Therefore, lower the MAE value is, better will be the model.



Source: Data quest

2. ROOT MEAN SQUARED ERROR (RMSE)

RMSE is similar to MAE but the only difference is that the absolute value of the residual is squared and the square root of the whole term is taken for comparison.The advantage of using RMSE over MAE is that it penalizes the term more when the error is high. (Note that RMSE is always greater than MAE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

The above two metrics are well known in the field of data science and machine learning which leaves me with nothing to talk about them. But one thing to note is that they are not complete withing themselves in case of Recommendation Systems i.e RMSE value of 0.8766 for an algorithm doesn't mean anything until there is another algorithm with another RMSE value with which we can compare our current RMSE value.

3. HIT RATE

HIT RATE = (HITS IN TEST) / (NUMBER OF USERS)

Hit Rate is a better alternative to MAE or RMSE. It does predicting on historical data, not future data which is possible only through online A/B tests

To measure a Hit Rate, first generate top N recommendations for all the users in our test data set**.** If generated top N recommendations contain something that users rated — 1 hit

Greater the Hit Rate better the Recommendation System

Hit Rate is easy to understand but measuring it is tricky. Best way to measure it is using Leave One Out Cross-Validation.

LEAVE ONE OUT CROSS VALIDATION:

compute the top N recommendation list for each user in training data and intentionally remove one of those items form user's training data. Then test the Recommender System's ability to recommend that intentionally removed an item in testing phase.

4. AVERAGE RECIPROCAL HIT RANK (ARHR)

- Hit Rate (HR)

$$HR = \frac{\#hits}{\#users}$$

- Average Reciprocal Hit Rank (ARHR)

$$ARHR = \frac{1}{\#users} \sum_{i=1}^{\#hits} \frac{1}{pos_i}$$

It is a variation of Hit Rate

- The difference is that we sum up reciprocal of the rank of each hit.
- It accounts for "where**"** in the top N lists our hits appear.

- It is more user-focused metric since users tend to focus more on the beginning of the list.

RANK	Reciprocal
3	1/3
2	1/2
1	1

If the user has to scroll down to see lower item in top N list, then it makes sense to panelize good recommendations that appeared too low in the top N list. Because, user had to work to find them - The ones he was interested on.

Rating	Hit Rate
5.0	0.001
0	0.004
3.0	0.030
2.0	0.001
1.0	0.0005

5. CUMULATIVE HIT RATE

- In this variation, throw away hits of predicted ratings if they are below some threshold.
- Confined to ratings above a certain threshold. Higher is better.

6. RATING HIT RATE

Another way to look at Hit Rate — Break it down by predicted rating score.

Small improvements in MSE leads to large improvements in Hit Rate.

7. COVERAGE

It is the percentage of possible recommendations (user-item pair) that system can predict

<user, item> pair

Coverage can be at odds with accuracy.

- It is a measure of how quickly new items will start to appear in our recommendation list. (similar to **churn** and **responsiveness** yet different)
- Measures the ability of the system to recommend long-tail items

8. DIVERSITY

It is a measure of how broad or variety of items our Recommender System is putting in front of people

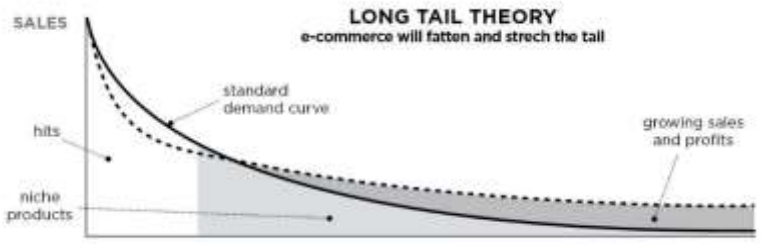
Diversity = 1 — Similarity

- Diversity and Similarity between recommendation pairs are opposite to each other
- high diversity leads to bad recommendations.

9. NOVELTY

It takes into consideration how popular are the items you are recommending.

The Long Tail: The x-axis in the below graph corresponds to products and y-axis corresponds to the popularity of the product. The shaded region is called “The long tail”. Leftmost products in the items are more popular compared to the rightmost items.



10. CHURN

It specifies how often recommendations change. If a user rates a new movie does it substantially change their recommendations if yes, your churn score is high.

11. RESPONSIVENESS

It measures “How quickly does new user behaviour influence recommendations”.

PROJECT PLAN

BUILDING A RECOMMENDATION SYSTEM FRAMEWORK

- **surpriselib** is a good library to work on Recommendation systems
- Even though **surpriselib** is a good package, it will be built on top of it so that algorithms can attain more flexibility.

VI. ANALYSIS OF RESULT

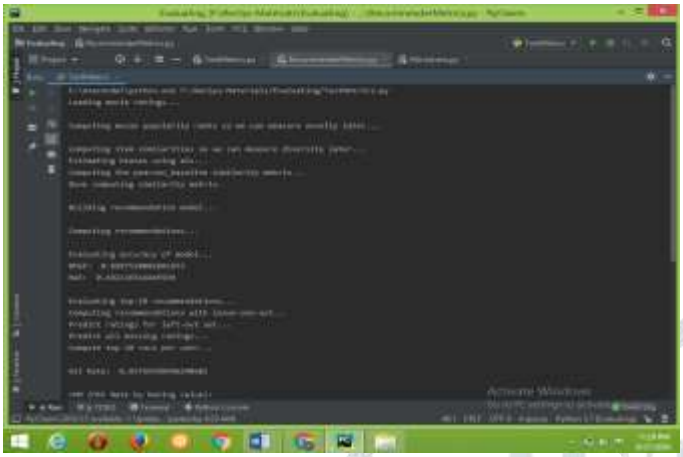
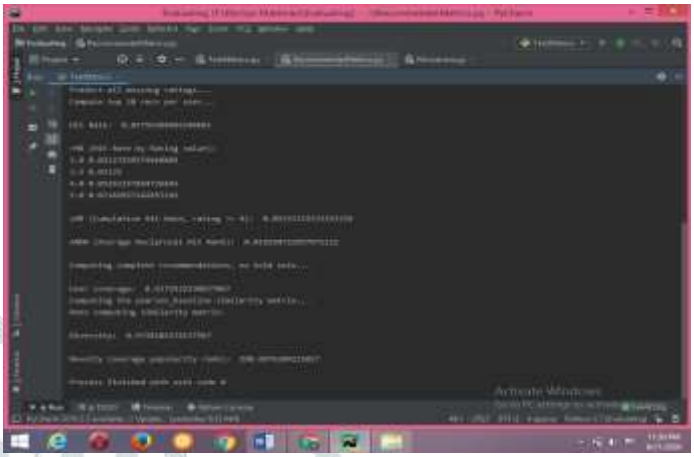



Fig 1 & Fig 2 specifies the evaluating accuracy of the predictions given by SVD algorithm

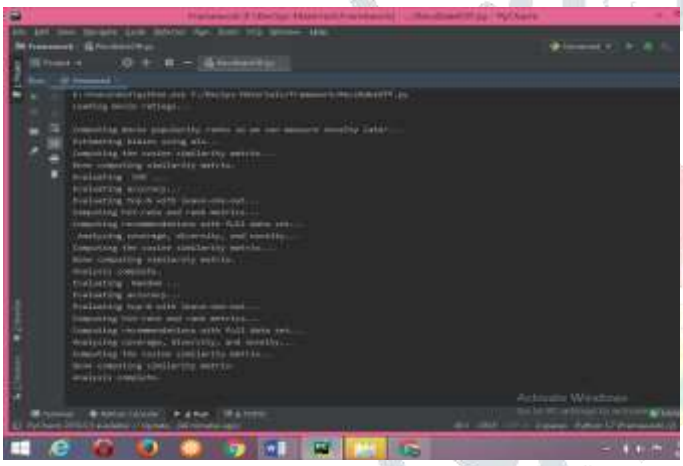
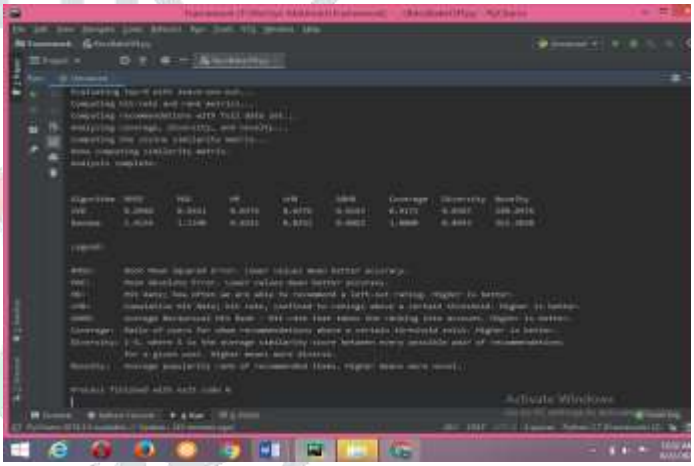



Fig 3 and Fig 4 comparing the accuracy of predictions between SVD and Random algorithm by using evaluation metrics

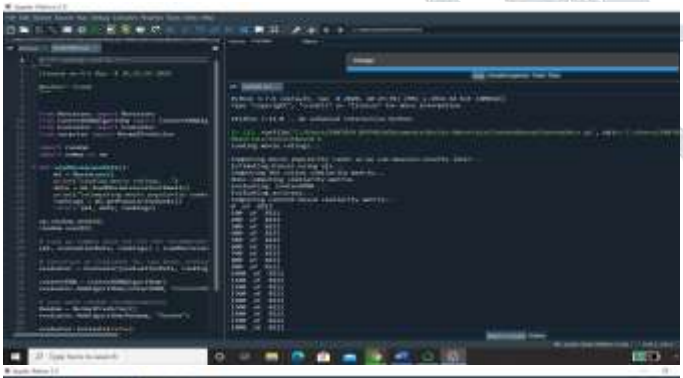
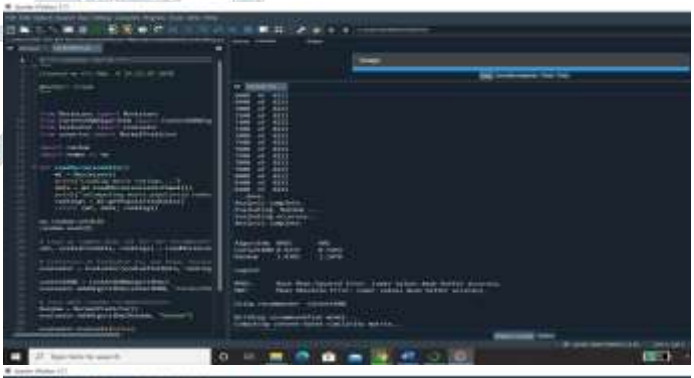
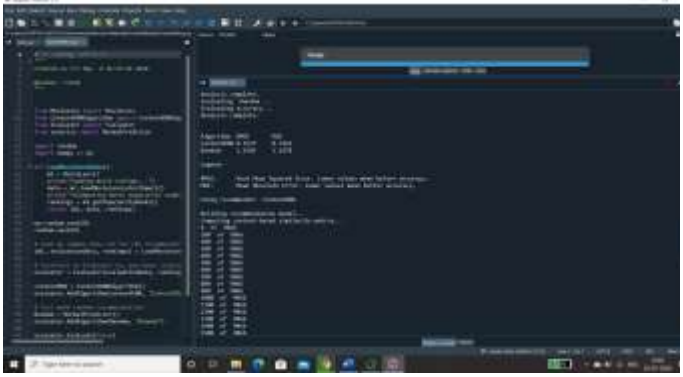
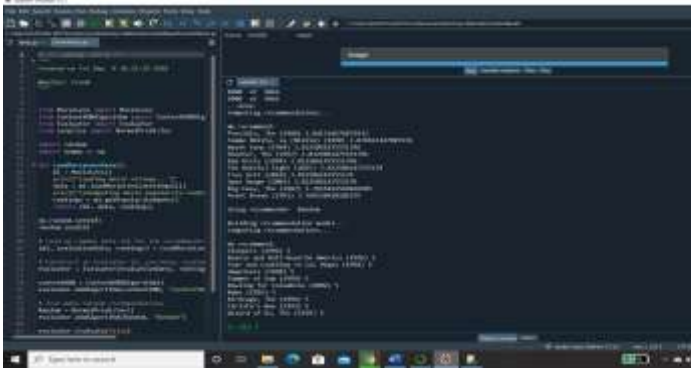
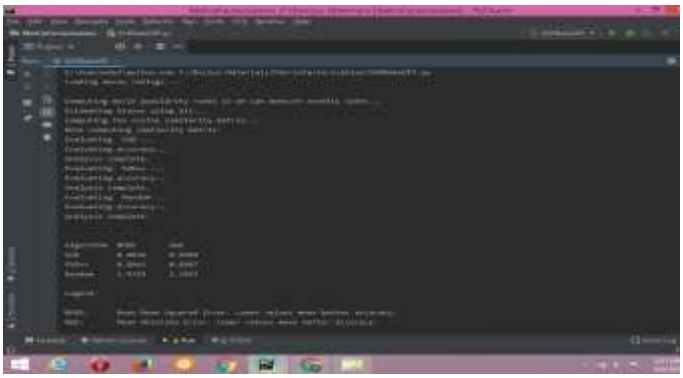





Fig 5 , 6, 7 & 8 are the outcomes of Producing and evaluating content based recommendations




Matrix factorization

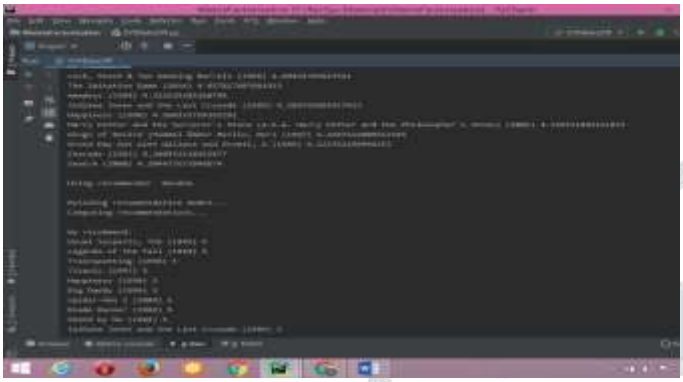
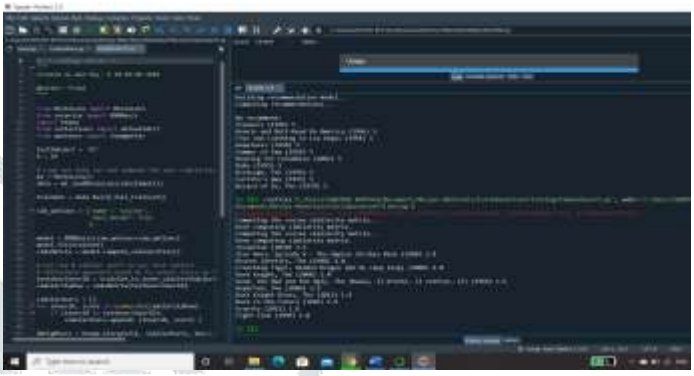




Fig 9, 10 &11 are the outcomes of calculating the movie ratings using SVD, SVD ++ and Random algorithms with matrix factorization technique

CONCLUSION

In this paper, specifies how recommendation algorithms should be evaluated in order to select the best algorithm for a specific task from a set of candidates. This is an important step in the research attempt to find better algorithms, as well as in application design where a designer chooses an existing algorithm for their application. Demonstrate that in some cases two algorithms can be ranked differently by two metrics over the same data set, emphasizing the importance of choosing the appropriate metric for the task, so as not to choose an inferior algorithm. And also describe the concerns that need to be addressed when designing offline and online experiments. We outline a few important measurements that one must take in addition to the score that the metric provides, as well as other considerations that should be taken into account when designing experiments for recommendation algorithms.

.

FUTURE SCOPE

In the future we will add more and more of these algorithms to the package and finally, more extended evaluation methods could be applied to the results, which will provide more detailed insights into the results. An example is the root mean square error (RMSE) that provides more insight into the incorrect predictions. Besides, this evaluation method was also used in the original competition from Netflix that came together with the dataset, which makes it possible to compare the achieved results with others.

REFERENCES

[1]M. Y. H. Al-Shamri, "User profiling approaches for demographic recommender systems," *Knowledge-Based Syst.*, vol. 100, pp. 175–187, 2016.

[2] J. Jooa, S. Bangb, and G. Parka, "Implementation of aRecommendation System Using Association Rules and Collaborative Filtering," *ProcediaComput.Sci.*, vol. 91, no.Itqm 2016, pp. 944–952, 2016.

[3] T. Osadchiy, I. Poliakov, P. Olivier, M. Rowland, and E. Foster, "Recommender system based on pairwise association rules," *Expert Syst. Appl.*, vol. 115, pp. 535–542, 2019.

[4] C. Binucci, F. De Luca, E. Di Giacomo, G. Liotta, and F. Montecchiani, "Designing the Content Analyzer of a Travel Recommender System," *Expert Syst. Appl.*, vol. 87, pp. 199–208, 2017.

[5] Z. Sun *et al.*, "Recommender systems based on social networks," *J. Syst. Softw.*, vol. 99, pp. 109–119, 2015. [6] C. A. Gomez-uribe and N. Hunt, "The Netflix Recommender System_Algorithms, Business Value.pdf," vol. 6, no. 4, 2015.

[7] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, pp. 109–132, 2013.

[8] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim, "Expert Systems with Applications A literature review and classification of recommender systems research," *Expert Syst. Appl.*, vol. 39, no. 11, pp. 10059–10072, 2012.

[9] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 261–273, 2015.

[10] L. Terán, A. O. Mensah, and A. Estorelli, "A literature review for recommender systems techniques used in microblogs," *Expert Syst. Appl.*, vol. 103, pp. 63–73, 2018.

[11] L. Ungar and D. Foster, "Clustering methods for collaborative filtering," *AAAI Work.Recomm.Syst.*, pp. 114–129, 1998.

[12] X. F. Meng and L. Chen, "The collaborative filtering recommendation mechanism based on Bayesian theory," *2009 1st Int. Conf. Inf. Sci. Eng. ICISE 2009*, pp. 3100–3103, 2009.

[13] M. Lu, Z. Qin, Y. Cao, Z. Liu, and M. Wang, "Scalable news recommendation using multi-dimensional similarity and Jaccard- Kmeans clustering," *J. Syst. Softw.*, vol. 95, pp. 242–251, 2014.

[14] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix Factorization model in Collaborative Filtering algorithms: A survey," *ProcediaComput. Sci.*, vol. 49, no. 1, pp. 136–146, 2015.

[15] X. Zhao, C. Zhu, and L. Cheng, "Coupled Bayesian matrix factorization in recommender systems," Proc. - 2017 Int. Conf. Data Sci. Adv. Anal.DSAA 2017, vol. 2018–Janua, pp. 522–528, 2018.