Evaluating the Accuracy and Coverage Performance of Collaborative Filtering, Content-Based, and Hybrid Recommender Systems

ANH HOANG, Davidson College, USA
MALAVIKA KALANI, Davidson College, USA
MIKE REMEZO, Davidson College, USA

Recommender Systems is a popular information filtering technology that lends itself to the e-commerce industry due to its usefulness in providing recommendations for consumers and increasing business revenue. There has been substantial research into different types of recommender systems and their implementations. In this paper, we present a comparative study of the three major paradigms of recommender systems: collaborative filtering, content-based, and hybrid methods. We used the MovieLens[9] data set for our analysis and based our results on two metrics, accuracy and coverage. Our findings reveal that collaborative filtering using matrix-factorization stochastic gradient descent is the best performing algorithm.

Additional Key Words and Phrases: Recommender Systems, Collaborative Filtering, Similarity and Prediction techniques

ACM Reference Format:
Anh Hoang, Malavika Kalani, and Mike Remezo. 2022. Evaluating the Accuracy and Coverage Performance of Collaborative Filtering, Content-Based, and Hybrid Recommender Systems. In *16th ACM Conference on Recommender Systems, Seattle, WA USA, Sept. 18 - 23, 2022*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Definition of Recommender Systems

Due to technological advancements, the e-commerce industry has witnessed drastic growth in recent decades. Individuals now spend the majority of their time on the Internet. However, given the overwhelming number of choices available online, there is a potential risk of information overload for many Internet users. Recommender systems offer solutions to this particular problem.

Recommender systems are personalized information filtering technology employed to make predictions on the potential likeness of users towards different products and provide them with recommendations for a set of items that are of interest to them. On the users' side, given the wide availability of products online, recommender systems solve the problem of how to filter, prioritize, and efficiently deliver suggestions on items that best fit consumers' needs, reducing the risks of information overload and increasing users' satisfaction. On the service provider's side, effective recommender systems can increase revenue and build consumers' loyalty to the business.

1.2 How Does A Recommender System Work?

In essence, a recommender system strives to predict the rating or potential preference that a user would give an item. Once the system outputs a prediction, recommendations will be given to users. Building a recommender system consists of three steps: retrieving users' data and formatting it for manipulation, computing similarity among users or items, and performing predictions of unknown ratings for users[4]. Users' data can be retrieved implicitly or explicitly[4]. Explicit data includes users' ratings and feedback, whereas implicit data can be congregated from users' purchase history, the number of clicks on certain items, and the likes[4]. Data will then be formatted; one common way to do so is to convert a user-item matrix into a rating matrix.

1.3 Types of Recommender Systems

Recommender systems can be categorized into three main types: Collaborative filtering, Content-based filtering, and Hybrid methods[4].

Collaborative filtering is an algorithm that generates recommendations for a user through a systematic analysis of data collected from other users sharing similar tastes or information needs[4]. Collaborative filtering can be broken down into memory-based methods and model-based methods[4].

Memory-based methods, also known as neighbourhood-based methods, are methods where the predictions of whether a user will like an item or not, can be computed using the information of their neighbours[4]. There are two main branches of memory-based methods: User-based collaborative filtering (UBCF) and item-based collaborative filtering (IBCF)[4]. UBCF focuses on finding people who like the same items as the active user and recommending items that those people like, while in IBCF, the similarities among items are calculated and the unrated items with high similarity will be recommended to users[4]. The core recommending algorithm can be broken down into three steps for both user-based and item-based methods:

(1) Apply different weightings to all users/ items to calculate the similarity among users and items
(2) Create a subset of users/items to predict the ratings for an item that the user has not rated
(3) Perform normalization of ratings and calculate the prediction using a weighted array of selected neighbours' ratings[1].

For model-based methods, the algorithms are built on the latent factor models[5]. Latent factor models seek to characterize both items and users on factors that can be inferred from the rating patterns[5]. One of the most successful latent factor models uses matrix factorization, which characterizes both items and users through vectors of factors that are computed from the item rating patterns[5]. Matrix factorization models map both users and items to a joint latent factor space, and user-item interactions are the inner products of corresponding vectors in the space[5]. To find out the factor vectors, the system minimizes the regularized squared error on the set of known ratings[5]. The system will try to learn the model by fitting the previously observed ratings.

Content-based filtering is a recommender algorithm that recommends items based on the attributes of the items themselves[4]. First, the algorithm will generate content information for items. Then, it will produce user-profiles and preferences with regard to features of the product[6]. Finally, it looks for items that are similar to the items that users have liked in the past by analyzing items' descriptions and features[6].

Hybrid methods are recommender algorithms that combine both content-based filtering and collaborative filtering methods to maximize their strengths and minimize their weaknesses[4]. There are multiple approaches to building a hybrid model. A notable example is combining separate recommenders, where we will use a linear combination of ratings or a voting scheme, or switch between the two after analyzing which one is better at any given moment[7]. Another popular approach is to add content-based characteristics to collaborative filtering methods, where we keep content-based profiles to aid with the collaborative filtering process, which helps with reducing sparsity and the algorithm will be able to recommend items directly similar to the user's taste[7].

1.4 Our Goal

In light of the substantial number of existing recommender systems models, we seek to explore which model performs the best. Therefore, our paper presents an empirical analysis of collaborative filtering, content-based, and hybrid recommender systems; ultimately, our goal is to propose the best recommendation algorithm based on metrics that will be defined in our thesis section.

2 RELATED WORK

In this section, we briefly present some of the research literature related to the implementations of the collaborative filtering system, , content-based and hybrid systems. Recommender systems have become a popular system that most companies and platforms rely on for offering a reliable and personalized experience to a user by providing the best recommendations.
User-based collaborative filtering systems predict the rating of an item by a user based on the ratings of other users who are similar to the active user. A common problem with user-based neighborhood recommendation systems is that it does not take into account that the similar users to the active users can have different levels of similarity.One way to address this problem is to apply similarity weighting to each contribution of each similar user [10] . Our research uses this idea of similarity weighting for our recommendations produced by the user-based collaborative filtering system.
There are two categories of collaborative filtering algorithms: memory-based which utilizes the entire user-item database to generate a prediction and model-based which provides item recommendation by developing a model of user ratings [2]. Matrix factorisation technique is another approach to producing recommendations that we have explored in our paper. It characterizes both items and users by vectors of factors inferred from item-rating patterns. In order to minimize the regularized squared error, there are two popular methods known as stochastic gradient descent and alternating least squares [5]. Stochastic gradient descent algorithm loops through all the ratings in the training set and alternating least squares rotates between fixing the item and user vectors which ensures that the squared error minimizes until it becomes really small. The accuracy and performance of both these methods have been detailed in our research.
Content-based recommendation system is another approach that analyzes item descriptions and features to identify items that might be worthy of a recommendation to a user [7] . Unstructured data such as unrestricted text items cannot be described by a set of attributes. A process of stemming is used to create a term that reflects the common meaning behind similar words [7] . The value of the variable related to the term is known as term-frequency times inverse document frequency (or tf-idf) which denotes the frequency of a term in a document, the number of documents that contain that term and the total number of documents in the collection. An important limitation of content-based systems using feature encoding is that the features of the recommended items must be easily parsed or assigned to items [7] . Additionally, hybrid approaches produce more accurate recommendations in comparison to pure approaches [7]. We have analyzed our results for such hybrid systems using item-based collaborative filtering approach combined with content-based TF-IDF technique. Lastly, our analysis of different hybrid recommender models is greatly aided by the comprehensive view of in-depth examples of different hybrid models and the discussion of benefits and drawbacks provided in [8].

3 THESIS

In this section, we will describe the research that we performed in this work and its main objective. We will also provide a brief description of how the research was conducted along with important details regarding the mathematical ideas and formulae used. Several algorithms and approaches have been explored to build a strong recommendation system for users to have a better experience while also managing the increasing load of user information. Our research aims to apply variations of parameters on the different algorithms implemented under collaborative filtering systems, model-based methods using matrix factorization, content-based filtering and hybrid recommendation systems. For each kind of recommender system, we will be analyzing the results produced by each algorithm using the best set of parameters. The goal of our research is to determine the best recommender system, one with the maximum coverage and highest accuracy. The metrics used in our experiment to evaluate the quality of each recommender system algorithm can be explained in the following way:

(1) Coverage: Coverage denotes the percentage of items for which the recommender system can compute predictions. In our experiment, coverage is shown as the number of items in place of a percentage value. It is desirable to have the maximum coverage as it implies that the prediction provided was computed based on almost all the item ratings existing in the dataset[1].
(2) Accuracy: In our experiment, the accuracy of the predictions provided by a recommender system algorithm is evaluated using certain error metrics. For all the error metrics used in our experiment, the lowest error indicates the highest accuracy.The formula for the main error metric used in this paper is shown below:
Mean Squared Error (MSE): The formula for calculating MSE where n denotes the number of ratings, Y_i denotes the actual rating and $\overline{Y_i}$ demotes the predicted rating is the following:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y_i})^2 \tag{1}$$

The two approaches used in our experiment to calculate our accuracy based on MSE are as follows:

(1) Leave-One-Out-Cross-Validation (LOOCV): For each rating in the user-item matrix, we remove one rating (which constitutes our test data) and use the train data predict our removed rating. Then, we store the squared error between our actual and predicted rating and restore the removed rating.
(2) Test/Train Holdout error: This approach divides the actual ratings matrix into test and train sets. For both SGD and ALS, a grid search of runtime parameters is conducted to determine the sets of parameters that best minimize the error between the predicted and actual ratings of the test set. [3]

3.1 User-based and Item-based Collaborative Filtering

For each of the user-based and item-based methods, we will be producing results using the following similarity weighting measures:

(1) *Euclidean distance for recommender systems* calculates the similarity between any two users across all the items they have rated in common. In recommender systems, the *euclidean distance* is a value ranging between 0 and 1 where 0 indicates no similarity and 1 indicates high similarity. The formula for calculating similarity between the ratings of two users (u_1, u_2) for two items (i_1, i_2) using euclidean distance is shown below:

$$Euclidian\ similarity\ distance = \frac{1}{1 + \sqrt{(u_1.i_1 - u_2.i_1)^2 + (u_1.i_2 - u_2.i_2)^2}} \tag{2}$$

(2) *Pearson Correlation* also calculates the similarity between two users across all the items they have rated in common. The *pearson correlation coefficient* is a value ranging between -1 and $+1$. A value of -1 indicates a negative correlation; a value of 1 indicates high similarity and 0 indicates no similarity. The formula for calculating similarity between ratings of two users X and Y,where $\overline{X}$ and $\overline{Y}$ denote the respective average of their ratings, is shown below:

$$\textit{Pearson Correlation between user X and user Y} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})}} \tag{3}$$

A common drawback of the collaborative filtering system is that an active user could have high correlation with other users based on very few commonly rated items between them. The computed correlation between two users does not accurately represent their true correlation if we only have a few data points to compare their item ratings. The accuracy of the predictions could be improved if we can apply a significance weighting measure in the following way: if the number of commonly rated items (n) between the two users is less than a weight w then we can apply a significance weight of n/w. If the number of commonly rated items (n) between two users is more than the weight, then we apply a weight of 1 [1]. The significance weights help us devalue the similarity weights that were based of fewer commonly-rated items as described in [1]. Ultimately, this enables us to produce a more reliable correlation between two users. For our experiment. we have applied the following significance weight variations to each of our user-based and item-based processes for each similarity measure (euclidean distance and Pearson):

(1) significance weight of 1
(2) significance weight of $n/25$ where n is the commonly rated items
(3) significance weight of $n/50$ where n is the commonly rated items

In addition to applying the significance weights, it is crucial to choose the specific users whose ratings can be used in computing the predicted rating of the active user. As explained in [1], it is more accurate to use a subset of other similar users instead of the entire database to generate a prediction for the active user. For large-scale collaborative filtering recommender systems, it is time-consuming to use the entire user database to generate predictions. This is why it is important to selectively choose the best set of users to compute a prediction for the active user. As explained in [1], we have implemented the absolute threshold technique in our experiment. According to this method, we create our subset of other users by selecting only those users whose absolute correlation with the active user is greater than our given threshold. For our experiment. we have applied the following threshold values to each of our user-based and item-based processes for each similarity measure (euclidian distance and pearson):

(1) threshold = 0
(2) threshold = 0.3
(3) threshold = 0.5

After choosing the best subset of other users, their ratings are utilised to compute a prediction for the active user. [3]

3.2 Collaborative Filtering using matrix factorisation

Matrix factorisation algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensional rectangular matrices.

Generally speaking, each item i is associated with a vector q_i and each user u is associated with a vector p_u. To learn the latent vectors, the matrix factorisation system minimises the MSE on the set of known ratings: $min_{q^*,p^*} = \sum_{(u,i)\in K}(r_{ui} - p_u q_i^T)^2$ where k is the set of the (u,i) pairs for which r_{ui} is known. The following mathematical equations were used for the system to learn the latent factors by minimizing the MSE: [3]

(1) usng SGD: $min_{q^*,p^*} = \sum_{(u,i)\in K}(r_{ui} - p_u q_i^T)^2 + \lambda(||q_i||^2 + ||p_i||^2 + b_u^2 + b_i^2)$ where b_u and b_i represent user and item bias components and λ is the regularisation term.
(2) using ALS: $min_{q^*,p^*} = \sum_{(u,i)\in K}(r_{ui} - p_u q_i^T)^2 + \lambda(||q_i||^2 + ||p_i||^2$ where λ is the regularisation term.

For our experiment, we have analysed the following parameters with SGD:

(1) number of latent features: 2, 20, 200
(2) learning rate: 0.02, 0.002, 0.0002
(3) regularization term: 0.2, 0.02, 0.002

We have analyzed the following parameters with ALS:

(1) number of latent features: 2, 20, 100, 200
(2) regularization term: 1.0, 0.1, 0.01, 0.001, 0.00001

3.3 Content-based methods: FE and TFIDF

Feature encoding turns categorical data in a dataset into numerical data. In case of our dataset ML-100K, feature encoding algorithm acquires the item-features' given domain. Then, we generate recommendation data by creating a preference matrix based on feature profiles of movies the user has seen and rated. The recommendations are computed for movies not yet rated by the user based on the feature profiles and the user preference matrix. The sum of the weighted preferences vectors are normalised to produce weights. A recommendation is computed by the sum of the product of the weights and user ratings: predicted rating = $\sum weights *$ (user ratings). It is important to note that there is no similarity threshold for the FE algorithm. [3]
Term frequency-inverse document frequency encodes text documents in multi-dimensional Euclidean space as weighted term vectors. In essence, the term frequency measures how often a term appears. It assumes that important terms should appear more often. Additionally the inverse document frequency aims to reduce the weight of terms that appear in all documents (common, may be high frequency). $TF(i, j)$ denotes the term frequency of keyword i in document j and $IDF(i)$ is calculated as $log\frac{N}{n(i)}$ where N is the number of documents and $n(i)$ is the number of documents from N in which keyword i appears. Cosine similarity metric is used to compare vectors by computing the angle between them. The algorithm used for our experiment can be explained in the following steps: [3]

(1) in the cosine similarity matrix, select the similarities between the item the active user has not rated and all the other items (i.e. similar items) the active user has rated.
(2) for every similar item, calculate the product of the rating and the similarity to them item that has not been rated.
(3) Compute the sum of the products and divide by the sum of all the similarities to the item not yet rated to generate a prediction. [3]

For our experiment, we use 0, 0.3, 0.6, 0.9 as our threshold variations for TF-IDF. Those threshold values were chosen based on the distribution of the cosine matrix that had a similar shape to a bell curve(fig 1).

3.4 Hybrid systems

For our experiment, we are using the TF-IDF system using the cosine similarity matrix as our base recommender system approach combined with an item-based collaborative filtering similarity matrix (using both euclidean distance and Pearson). When the TF-IDF similarity is zero, we look for the corresponding value in the item-item similarity matrix multiplied by a weighting factor of 25%, 50%, 75% or 100%. [3]

3.5 Hypothesis

Implementing the mathematical concepts and algorithms described above, we hypothesize that MF-SGD should be the best performing algorithm based on accuracy and coverage. Matrix factorization algorithm allows for the incorporation of a much more in-depth dataset of information: explicit feedback, implicit feedback, biases, dynamics along with confidence levels[5]. The prediction of calculation is based on more subtle and complex feature dimensions that user-based or item-based collaborative filtering methods won't be able to spot. Furthermore, we predict that SGD will outperform ALS since ALS is more equipped to deal with a dataset that contains substantial implicit feedback, which is not the case with ML-100k.

4 EXPERIEMENTAL DESIGN

4.1 Datasets

For our experiment, we used the ML100k data from MovieLens[9]. There are a total 943 users, 1664 movies, and 99693 ratings in this dataset[9]. Ratings are provided in the range of 1 to 5[9].

4.1.1 ***Layout of dataset statistics***. Descriptive analytics data/Chart for ml-100K:

Number of users: 943

Number of items: 1664

Number of ratings: 99693

Overall average rating: 3.53 out of 5, and std dev of 1.13

Average item rating: 3.08 out of 5, and std dev of 0.78

Average user rating: 3.59 out of 5, and std dev of 0.44

User-item Matrix Sparsity: 93.65%

Average number of ratings per users: 105.718982, and std dev of 100.567291

Min, Max, Median number of ratings per users: 20, 737, 65

Average number of ratings per items: 59.453032, and std dev of 80.359947

Min, Max, Median number of ratings per item: 1, 583, 27

4.2 Charts

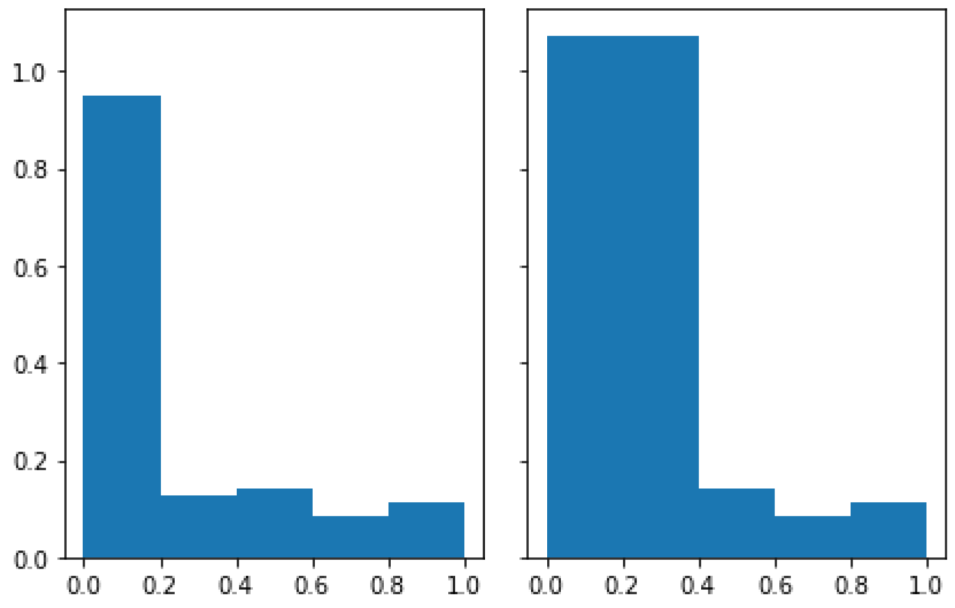


Fig. 1. cosine similarity distribution

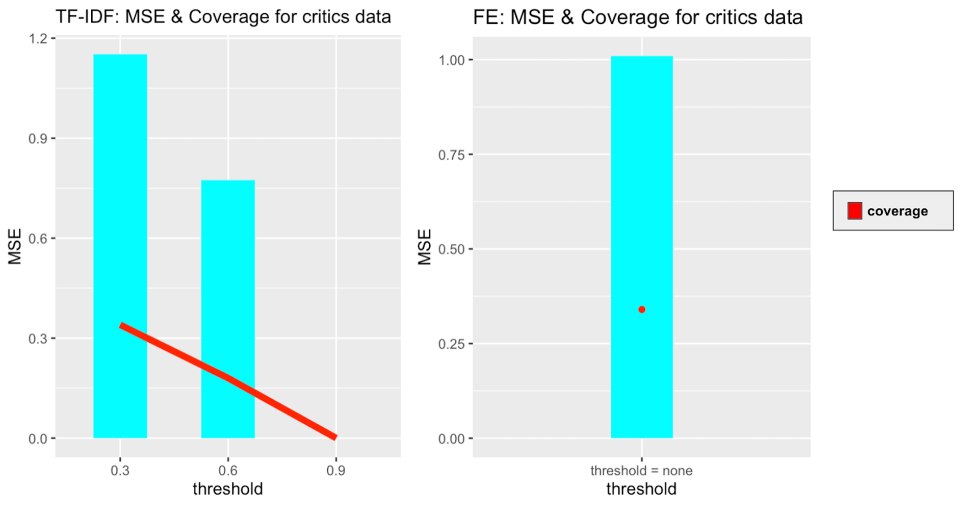

TF-IDF: MSE & Coverage for critics data
MSE
threshold
FE: MSE & Coverage for critics data
MSE
threshold = none
threshold
coverage

Fig. 2

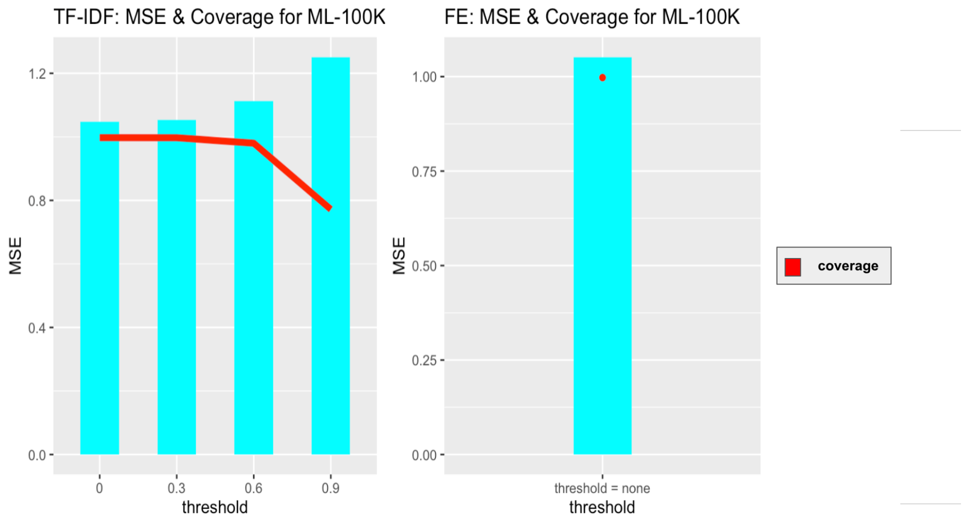

TF-IDF: MSE & Coverage for ML-100K
MSE
threshold
FE: MSE & Coverage for ML-100K
MSE
threshold = none
threshold
coverage

Fig. 3

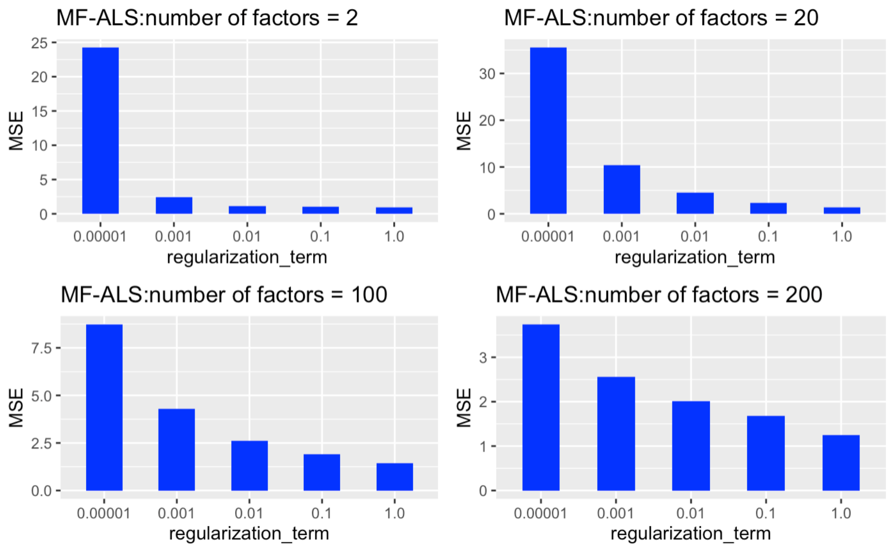

MF-ALS:number of factors = 2
MF-ALS:number of factors = 20
MF-ALS:number of factors = 100
MF-ALS:number of factors = 200
MSE
regularization_term
0.00001
0.001
0.01
0.1
1.0

Fig. 4

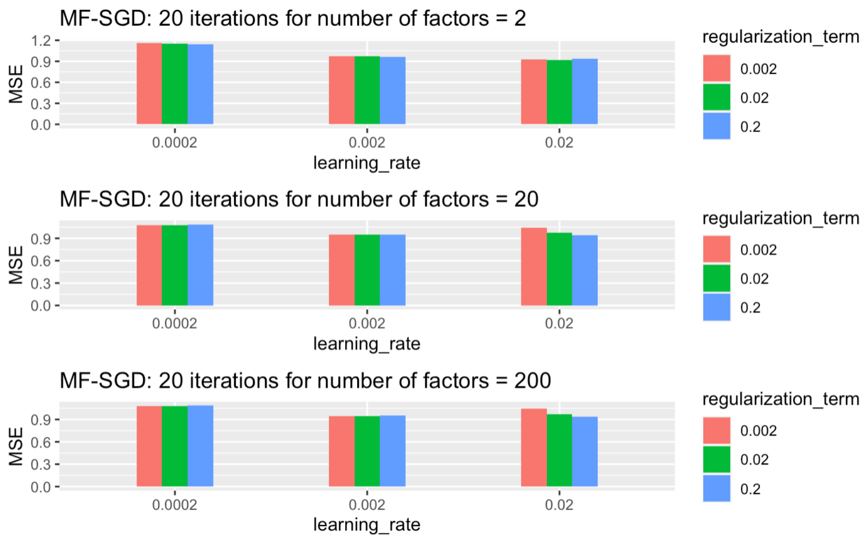

MF-SGD: 20 iterations for number of factors = 2
MF-SGD: 20 iterations for number of factors = 20
MF-SGD: 20 iterations for number of factors = 200
MSE
learning_rate
0.0002
0.002
0.02
regularization_term
0.002
0.02
0.2

Fig. 5

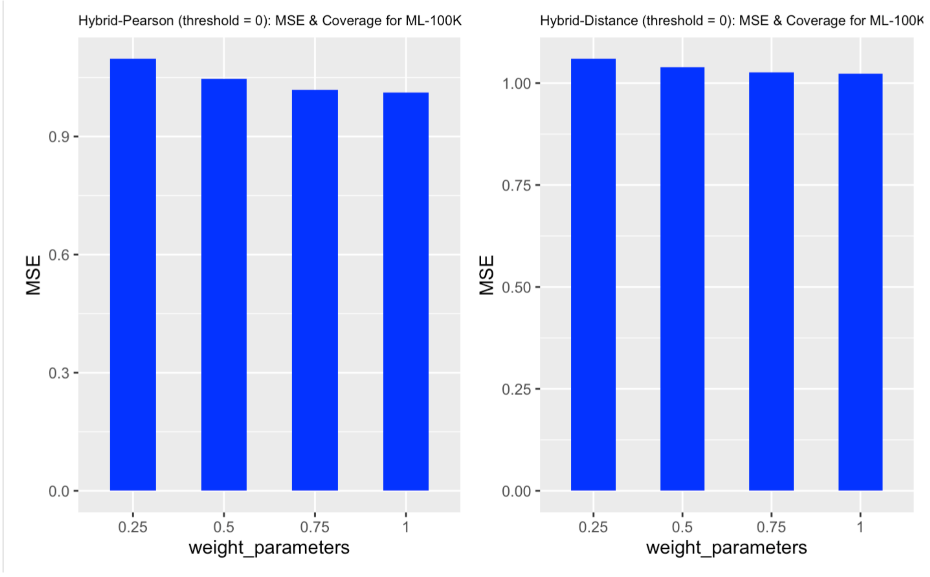


Fig. 6

4.3 Choosing The Best Set Of Parameters For Each Algorithm

With regards to user-based collaborative filtering algorithms, we decided that the best set of parameters would be the similarity threshold of 0 and the significant weighting of 25 for both Pearson and Distance algorithms. The findings in our earlier research highlight a strong negative correlation between the similarity threshold value and coverage[12]. Increasing the significant weighting from 0 to 0.3, and then to 0.5, dramatically decreases the coverage of our results, hence reducing the reliability of the MSE value[12]. Therefore, a similarity threshold of 0 would ensure a low MSE and a high coverage. In addition, higher significant weighting seems to increase coverage significantly while increasing MSE by a little bit, so we choose a significant weighting of 25 to balance out the increase in coverage and increase in MSE[12].

For item-based collaborative filtering algorithms, we notice the same patterns as user-based CF, where increasing the significant weighting from 0 to 0.3, and then to 0.6, dramatically decreases the coverage of our results, till the point of 0 for Distance[12]. The significant weighting increases coverage and decreases MSE[12]. Therefore, we decided to go with a similarity threshold of 0 and significant weighting of 25 for both Pearson and Distance algorithms.

In TFIDF algorithm, we notice that the higher threshold value is, the higher MSE (Figure 3). Therefore, we decided to go with the threshold value of 0 for the algorithm.

In terms of MF-SGD algoritm, across all three numbers of iterations, learning rate value alpha = 0.02 produces the lowest MSE results (Figure 5). As alpha decreases, MSE increases (Figure 5). Number of factors = 2 and alpha = 0.0002 produces the highest MSE (Figure 5). In addition, big values of factors also require a higher value of regularization. As regularization goes down, it is more likely that the algorithm will overfit data (Figure 5). Therefore, we decided that the best set of hyper-parameters for SGD is k = 2, alpha = 0.02, beta = 0.02 because it outputs considerably low MSE and does not overfit data.

For ALS algorithm, as regularization goes down, it is more likely that the algorithm will overfit data (Figure 4). In addition, big values of factors also require a higher value of regularization (Figure 4). Therefore, we think that the best set of hyper-parameters for ALS is k = 2, beta = 1.0. The set produces a low MSE of 0.9264 and does not overfit data.

For hybrid algorithms, we chose the TFIDF threshold of 0 for both Hybrid-Pearson and Hybrid-Distance because it is the best parameter for the algorithms as explained above. We tested our data on different weight parameter, and our findings shows that a higher value of weight parameter output higher MSE, which is the case for both Hybrid-Pearson and Hybrid-Distance (Figure 6). Therefore, we decided to go with a weight-parameter of 1 for both hybrid algorithms.

5 RESULTS

After selecting the best set of parameters for each of our algorithms as explained in our experimental design, we conducted a comparison of the accuracy and performance between each algorithm to determine the best recommender system approach. For the next step in our analysis, across the best parameters for each algorithm, we selected the best performing variation for each of the four approaches: user and item-based collaborative filtering, collaborative filtering using matrix factorisation, content-based filtering and hybrid method.

For collaborative filtering, our results demonstrate that item-Pearson and item-distance produce the lowest MSE (Fig. 6). It is also important to note that the coverage of both item-based algorithms does not differ significantly.

For model-based methods using matrix factorisation, our results demonstrate that both SGD and ALS perform very similarly but matrix factorisation using stochastic gradient descent algorithm produces a slightly lower MSE (Fig. 7).

For content-based methods,our results demonstrate that both FE and TFIDF perform very similarly but TFIDF algorithm shows better accuracy with a slightly lower MSE (Fig. 8).

For our hybrid approaches, hybrid method using item-pearson demonstrates better accuracy and a slightly higher coverage .(Fig. 9).

5.1 Results Charts

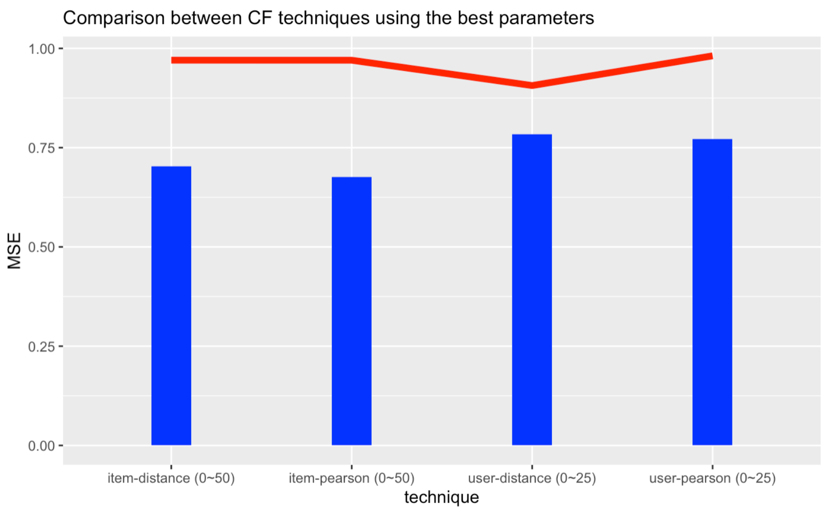


Fig. 7

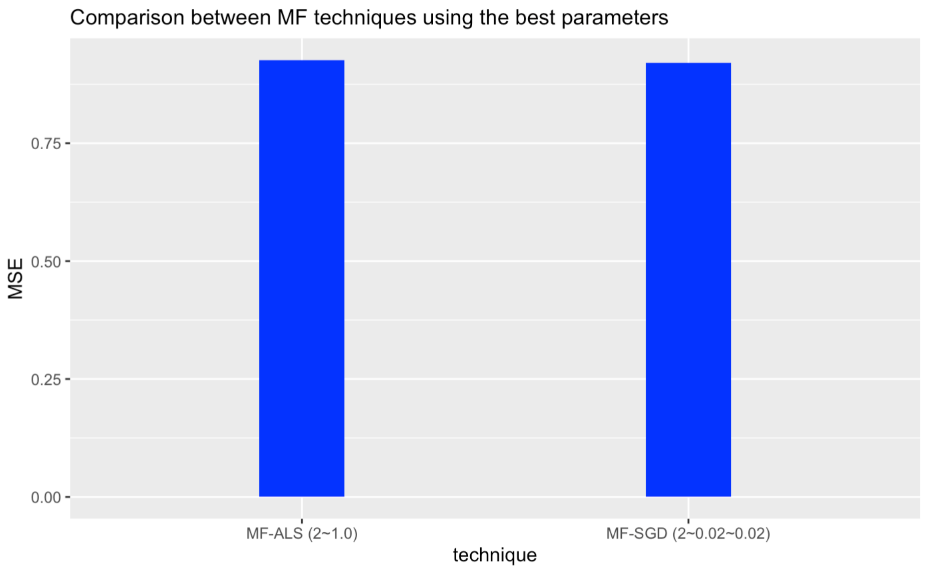

Comparison between MF techniques using the best parameters
MSE
0.00
0.25
0.50
0.75
MF-ALS (2~1.0)
MF-SGD (2~0.02~0.02)
technique

Fig. 8

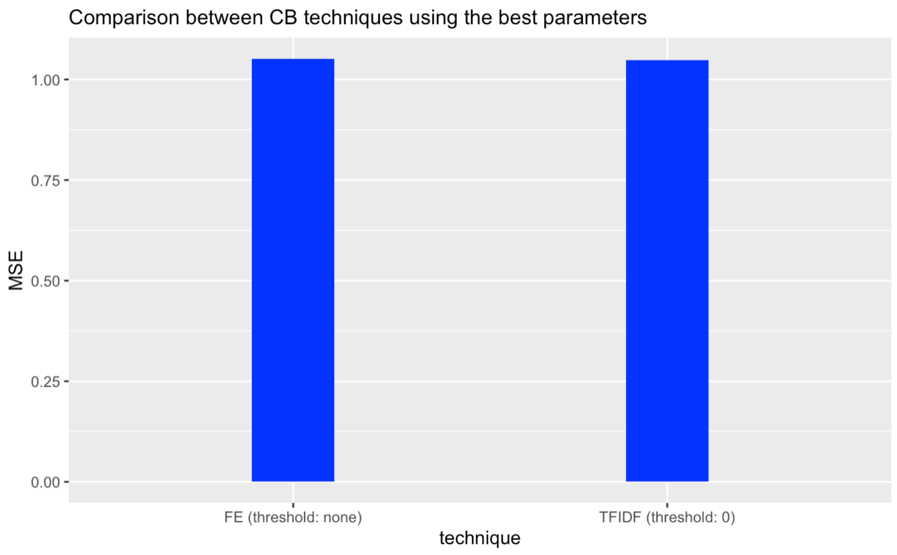

Comparison between CB techniques using the best parameters
MSE
0.00
0.25
0.50
0.75
1.00
FE (threshold: none)
TFIDF (threshold: 0)
technique

Fig. 9

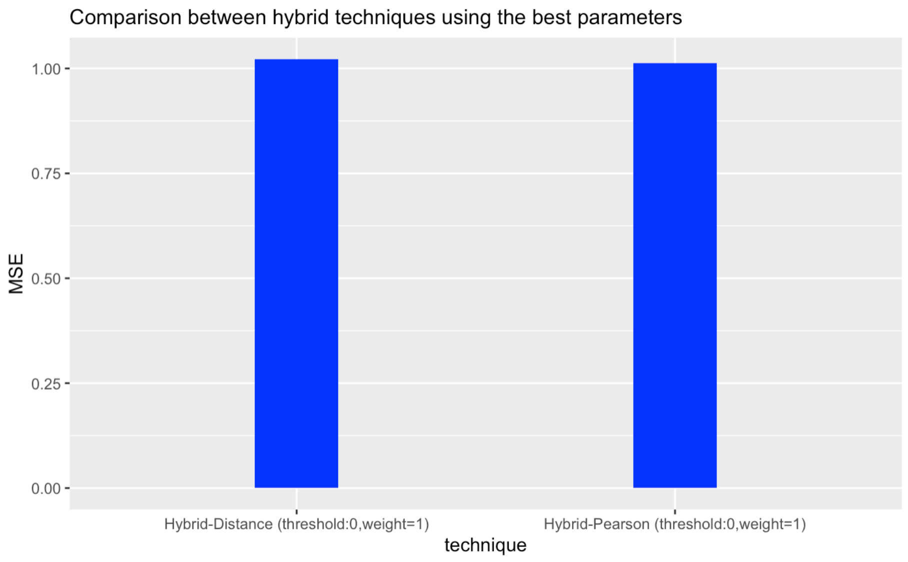

Comparison between hybrid techniques using the best parameters
MSE
1.00
0.75
0.50
0.25
0.00
Hybrid-Distance (threshold:0,weight=1)
Hybrid-Pearson (threshold:0,weight=1)
technique

Fig. 10

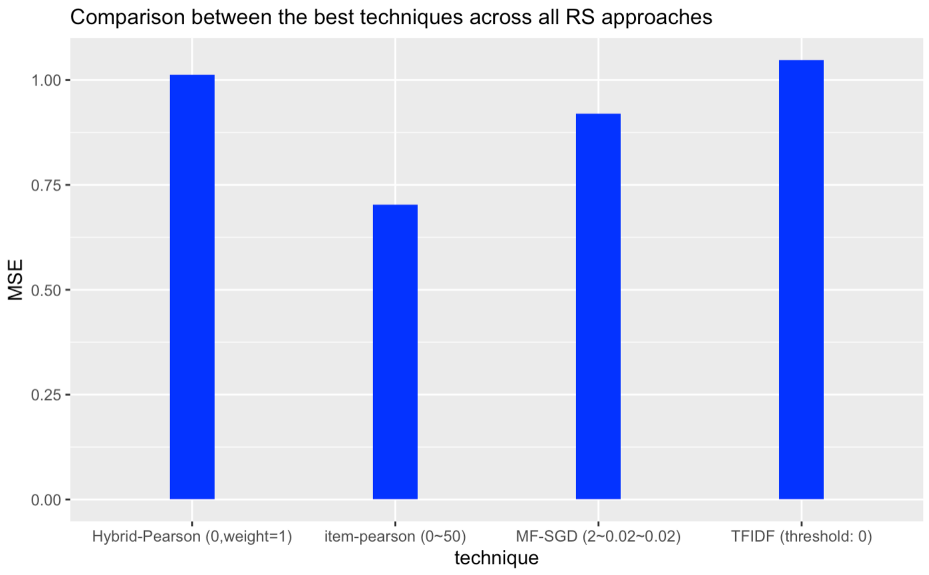
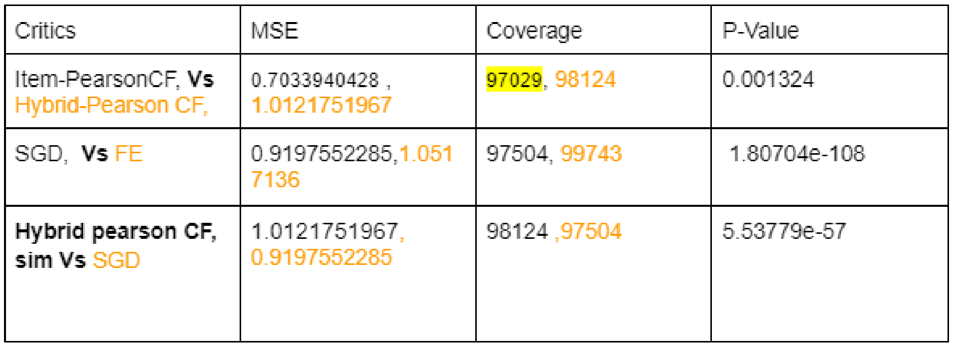

Comparison between the best techniques across all RS approaches
MSE
1.00
0.75
0.50
0.25
0.00
Hybrid-Pearson (0,weight=1)
item-pearson (0~50)
MF-SGD (2~0.02~0.02)
TFIDF (threshold: 0)
technique

Fig. 11



Critics	MSE	Coverage	P-Value
Item-PearsonCF, **Vs** Hybrid-Pearson CF,	0.7033940428 , 1.0121751967	97029, 98124	0.001324
SGD, **Vs** FE	0.9197552285,1.0517136	97504, 99743	1.80704e-108
Hybrid pearson CF, sim Vs SGD	1.0121751967, 0.9197552285	98124 ,97504	5.53779e-57

Fig. 12

6 DISCUSSION

According to our results in Fig. 6, item-based pearson showed better accuracy than item-based distance. Therefore, we selected item-pearson as the better memory-based collaborative filtering. Our selection corresponds with the findings in previous research conducted on different similarity weighting measures, where Pearson outperformed all variants tested (Herlocker et al ., 1999).

Fig. 7 indicates that model-based method SGD and AlS perform equally well in terms of accuracy. However, SGD is a better performing algorithm because it is generally easier and faster to implement. Moreover, ALS is more preferable when the system includes more implicit data which is not the case for our RML-100K dataset because it has lower sparsity.

Fig. 8 indicates that TF-IDF is a more accurate content-based algorithm. Feature analysis using feature encoding can be less accurate because every feature might not have similar importance (Seminario slides). Moreover, the features must be easily parsed or manually assigned to each item which proves to be a more time-consuming process (Adomavicius 2005). Fig.9 is comparing all the best techniques show that item-based distance and mf-sgd perform significantly better than hybrid-pearson and tfidf in terms of accuracy. Even though that item-based distance perform better than mf-sgd, we believe that mf-sgd will surpass item-based distance in the long run. Since mf-sgd allows for the incorporation of a much more in-depth dataset of information: explicit feedback, implicit feedback, biases, dynamics along with confidence levels. The prediction of calculation is based on more subtle and complex feature dimensions that Item-based CF methods won't be able to spot. As more data is collected, mf-sgd will definitely output better recommendations. Therefore, we choose SGD as our best recommender system.

Our final table compares the 4 techniques using t-test to determine if there is a significant difference between the means of two groups, which may be related in certain features. After comparing all the four elements as shown in fig 11. We can deduce that the mean squared errors between item-pearson and MF-SGD are different.

We select MF-SGD because it has a relatively low MSE and offers a compact memory-efficient model that systems can learn relatively easy.Additionally, it can integrate naturally many crucial aspects of the data such as multiple forms of feedback, temporal dynamics and confidence levels [5].

7 CONCLUSION

Overall, the findings confirmed our original hypothesis, with MF-SGD recommender systems outputting the best accuracy and coverage. We acknowledge that there are still many opportunities for further analysis following the footsteps of our work. First, there are potentially many recommender systems that have not been covered in our study, such as demographic, utility-based and knowledge-based techniques along with a substantial number of different hybrid models[8]. We plan to further extend our study by providing a comprehensive analysis of a wider range of recommender systems with the incorporation of the above-mentioned methods. In addition, one limitation of our study is that the dataset that we test our algorithms on is solely for movies. Each algorithm may perform differently when dealing with different domains of data. There are different domains that we look forward to performing our algorithms on, such as books and food data. Another important point is that our dataset is static, so we would also want to explore how algorithms deal with a dynamic dataset that has ratings constantly being updated. Finally, one of the metrics that we haven't touched on in our study is the runtime of our algorithm. When dealing with big data, the element of runtime requires utmost attention. Therefore, we want to further extend our study with our algorithms' runtime as one of the main metrics for our analysis.

8 ACKNOWLEDGMENTS

We are grateful to our CSC 381 professor Dr. Carlos Seminario who personally worked with us to help us navigate through the challenges in the project. We are also grateful to the authors of our cited related research works because of how their ideas have helped us shaped our own research.

REFERENCES

[1] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, John Riedl, *An Algorithmic Framework for Performing Collaborative Filtering*, 1999
[2] Badrul Sarwar, George Karypis, Jospeph Konstan, and John Riedl, *Item-Based Collaborative Filtering Recommendation Algorithms*, 2001
[3] Carlos Seminario, *Moodle Slides*, 2022
[4] Parul Pandey, *The Remarkable World of Recommender Systems*, 2019
[5] Koren et al, *Matrix Factorization Techniques for Recommnder Systems*, 2009
[6] Pazzani Billsus, *Content-Based Recommendation Systems*, 2007
[7] G. Adomavicius, *Toward the Next Generation of Recommender Systems*, 2005
[8] Burke, *Hybrid Recommender Systems: Survey and Experiments*, 2002
[9] F.M. Harper and J.A.Konstan, *The MovieLens Datasets: History and Context*, 2016
[10] Desrosiers/Karypis, *A comprehensive survey of neighborhood-based recommendation methods*, 2011
[11] Zhou, *Large-scale Parallel Collaborative Filtering for the Netflix Prize*, 2008
[12] Hoang, Wang, Remezo, Kalani, *Impact of Variations of Similarity and Prediction Techniques on User-based and Item-based Collaborative Filtering Recommendations*, 2022