Improving the Scalability of Collaborative Filtering Recommendation with Clustering Techniques

Teng Moh
Department of Computer Science
San Jose State University
San Jose, CA
teng.moh@sjsu.edu

Nikunj Rana
Department of Computer Science
San Jose State University
San Jose, CA
nikunj.rana@sjsu.edu

***Abstract*—Nowadays, we need to recommend a lot to users, whether it's about the food items they might like or the movies they might watch. Collaborative filtering (CF) is a popular method for generating these recommendations, but a major issue it struggles with is scalability when dealing with large datasets. In this study, we explore different clustering techniques combined with traditional CF methods to enhance scalability and performance. By dividing the users into clusters and then applying CF, this method helps reduce computational complexity. This paper details our methodology, the results of our experiments, and the implications for future recommendation systems.**

***Index Terms*—Collaborative Filtering, Clustering Techniques, Recommendation Systems, Scalability**

I. Introduction

Nowadays, we are living in an era where we have a lot of data, and using this data, we can recommend different items to users. Among various approaches for recommendation systems, one of the famous approaches is collaborative filtering. By using user-to-item interaction data, CF methods can identify patterns and similarities, thus providing new recommendations to the user. However, as the scale of data grows, the computational demands for processing this large data also grow. This happens as we need to compare millions or billions of users to millions or billions of items, which can be both time-consuming and resource-intensive. As a result, there is a major need for developing methods that maintain accuracy as well as reduce computational costs. One approach for handling this scalability issue is to use clustering with collaborative filtering. By clustering similar users together and then applying collaborative filtering on it, we can improve the computational efficiency of CF. In this study, we explore the impact of using clustering alongside CF methods. We aim to see whether clustering alongside CF methods would be a good approach and whether it is possible to further enhance the result from the base implementation [1].

II. Related Work

Most of the previous research on improving scalability discusses the use of clustering methods. Authors in papers [1] and [2] have employed clustering to enhance scalability but have used different similarity matrices. This project uses the research conducted by [1] as the baseline for experimentation and discusses the methodology and implementation in the next section.

In 2010, Jia Rongfei et al. [2] conducted an experiment to improve the scalability of collaborative filtering. A major takeaway from this study is that they used adjusted DBSCAN as one of the clustering methods, which appeared to perform better than K-means. However, when V. Botti-Cebriá et al. [1] experimented with the DBSCAN clustering method, it seemed to perform poorly compared to K-means. In this study, we will check if adjusted DBSCAN could be a could alternative for the Kmean clustering method.

In the study performed by Young-dul et al. [3], the authors discussed creating a genre preference vector that could help reduce clustering costs. They used a rating cutoff of 4 for forming clusters, while authors in [1] used a cutoff of 3.5 for the same MovieLens dataset.The major drawback of this approach would be that we will not be assigning the users that have not rated above 3 to any cluster and thus those users would not be getting recommended. In this study we will be exploring with different rating cutoff to identify the impact of the cutoff on the result and try to see if we can use any other cutoff method that ensures all users get assigned to a cluster.

III. Baseline

In Fig. 1 we can see the architecture of the authors approach. The authors in [1] have divided the process for improving the scalability of collabarative filtering in three major steps which will be discussed after discussing about the dataset.:

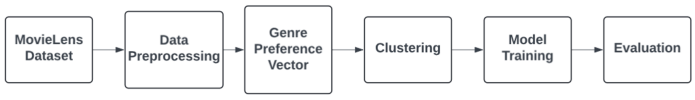

Fig. 1. Architecture

A. Dataset:

The dataset used by the authors [1] is the Movielens dataset. They have done the experiment on 1M dataset and 10M dataset.1M dataset has 4000 users with 38000 movies and all movies are divided into 18 genres.Each user has rated atleast 20 movies and a total of 1,000,000 user ratings are present in the dataset.

B. Creating genre Vector:

The first step authors[1] have performed is creating genre vector. Instead of using the usual user-item matrix for creating clusters, the authors have decided to use genre-user matrix which they call as 'genre vector'. The main advantage of using the genre vector is that in the MovieLens dataset, a user-item matrix would result in a 4000x38000 matrix, which is considerably large. In contrast, the genre vector is a much more manageable 4000x20 matrix.

C. Dividing user in cluster:

For dividing users in clusters. the authors have used different clustering techniques. First author have used clustering techniques that does not reuire the number of clusters to be pre determined such as DBSCAN,OPTICS and mean shift clustering. The drawback of these clustering methods was that they generated clusters of 10 users which are low for conducting collabarative filtering on to give better recommendations. Secondly, authors have used clustering techniques like Kmeans,MiniBatch Kmeans and Birch which require the user to assign the number of cluster in the start.For assigning the number of clusters, the authors have used multiple clusters size of 5,10,25,50,100,150 and 200.They found that if they assign more than 100 number of clusters, then the number of users in a clusters come down to 4 which is very low for doing a collabarative filtering on it so they decided to test these methods using 5,10,25,50 and 100 clusters. After performing the clustering the results of the clusters were evaluated based on these metrics: Silhouette, Calinski and Davis scores and results of the evaluation is shown in Fig 2

SILHOUETTE, CALINSKI AND DAVIES SCORES FOR DIFFERENT CLUSTERING TECHNIQUES.

Techniques	Nº Clusters	Silhouette	Calinski	Davies
KMeans	10	***0.174***	***1008.4***	***1.423***
	50	**0.105**	**409.8**	**1.679**
	100	**0.092**	**257.4**	**1.741**
Mini Batch KMeans	10	0.138	874.6	1.714
	50	0.093	380.8	1.828
	100	0.082	246	1.864
BIRCH	10	0.118	838.3	1.65
	50	0.062	347.7	1.845
	100	0.053	226.1	1.927

Fig. 2. Clustering results

Based on the results we can see that Kmean performed the best as compared to other clustering techniques which could be because KMeans optimizes the within-cluster sum of squares , which generally leads to compact and well-separated clusters. Another thing to notice is that as we increase the number of clusters, the Silhoutte and Calinski score decreases while the Davies score increase.This gives us the idea that the less the number of cluster, the better the clustering result.

D. Recommendation Experiments:

The authors found that Kmean is producing the best results and using the clusters found by Kmeans they have performed different CF techniques like K-NN, SVD and SVD++. In each cluster they have split the data in 75% for training and 25% for testing.They have used the following parameters for each methods: K-NN (k=40, mink=1), SVD (n_factors=100, n_epochs=20, lr=0.005), and SVD++ (n_factors=20, n_epochs=20, lr=0.007) provided by the Microsoft Recommenders library [4].

E. Evaluating Results:

For evalauting the results, the authors have used the RMSE and MAE metrics. The results for the evaluation metrics are shown in Table I.Based on the results, we can see that the best results are seen when the number of clusters is 0 and as we keep on increasing the number of clusters, the error keep on increasing for RMSE and MAE. So, from this we get the idea that the best results come when we do not perform any clustering but this result was expected by the authors.Now we need to check the reduction in the temporal data to see whether the trade off between time and accuracy is good enough.

TABLE I
ERROR METRICS FOR DIFFERENT CLUSTERING TECHNIQUES AND NUMBER OF CLUSTERS

Metrics/Clusters		**Cluster number**					
Technique	**Metric**	**0**	**5**	**10**	**15**	**20**	**25**
KNN	**MAE**	0.732	0.753	0.794	0.802	0.808	0.818
	RMSE	0.928	0.955	1.008	1.019	1.028	1.041
SVD	**MAE**	0.691	0.726	0.755	0.759	0.762	0.767
	RMSE	0.880	0.920	0.952	0.956	0.959	0.965
SVD++	**MAE**	0.681	0.713	0.744	0.747	0.751	0.756
	RMSE	0.869	0.908	0.942	0.946	0.951	0.957

F. Temporal cost:

The results for temporal cost are shown in Fig 3. The results are shown as percentages of time improvements adding clustering against the non-clustering experiments. Based on the temporal results, we can see that, for KNN the improvement was by a big margin for training and testing phase.This is mainly because instead of finding distance of each user to other user we are just doing it for subsets of the user which reduces the time complexity.But incase of SVD and SVD++ the change is not that big but definitely improvement that can be seen.

	Clusters	KNN		SVD		SVD++	
		Train	Test	Train	Test	Train	Test
1M	5	80%	67%	-6%	10%	0.3%	6%
	10	86%	77%	1%	18%	1%	5%
	15	91%	84%	3%	21%	0.2%	10%
	20	93%	88%	3%	25%	0%	18%
	25	94%	89%	2%	26%	-0.4%	19%

Fig. 3. Temporal costs

IV. REPRODUCTION RESULTS:

In his experiments, authors [1] have performed experiment on 1M and 10M Movielens dataset. The results below are just for the 1M dataset as while performing the experiment on 10M dataset more than 8Gb system RAM is required and when performing on online sources like google colab, multiple

time out and memory issues were seen because of which the experiment results discussed in this section are only for 1M dataset.

A. Dividing user in Cluster:

The results are calculated using Kmeans clustering.As can be seen in the results Fig. **??**, the reproduced results are almost similar as compared to the original baseline results. There are difference in results and that is majorly because of randomness of initial point in Kmeans. 100 different initial states were tested and these are the best results found.

	Sillhoutte(Repro/Orig)	Calinski(Repro/Orig)	Davies(Repro/Orig)
10	0.174/0.174	1013.9/1008.4	1.419/1.423
50	0.114/0.105	414.5/409.8	1.629/1.679
100	0.092/0.092	261.8/257.4	1.73/1.741

Fig. 4. Comparison of Kmean evaluation metrics

B. Evaluation results:

After performing clustering KNN,SVD and SVD++ was applied and the results for KNN and SVD are showin in Fig 5and result for SVD++ are shown in Fig 6.Based on the results, we were able to get a little better results as compared to original paper and that could be mainly because of the slight better cluster results we got.

		RMSE(Repro/Orig)	MAE(Repro/Orig)
K-NN	0	0.925/0.928	0.73/0.732
	5	0.944/0.955	0.744/0.753
	10	0.964/1.008	0.76/0.794
	15	0.982/1.019	0.773/0.802
	25	1.001/1.041	0.788/0.818
SVD	0	0.883/0.880	0.691/0.691
	5	0.887/0.920	0.687/0.726
	10	0.903/0.952	0.7/0.755
	15	0.914/0.959	0.709/0.759
	25	0.923/0.965	0.719/0.767

Fig. 5. Comparison of KNN and SVD evaluation metrics

		MAE(Repro/Orig)	RMSE(Repro/Orig)
SVD++	0	0.681/0.681	0.854/0.869
	5	0.702/0.713	0.895/0.908
	10	0.723/0.744	0.935/0.942
	15	0.732/0.747	0.941/0.946
	25	0.744/0.756	0.948/0.957

Fig. 6. Comparison of SVD++ evaluation metrics

C. Temporal Cost:

Although the authors [1] have given the temporal costs as percentage, I have used the exact seconds value for showing the results. The major reason is that in percentage mode we can only compare the clustering to non clustering method for each CF method but in exact time we can compare the methods time with each other too. Also, in calculation author is comparing the training and testing times but we also need the clustering time to get the whole picture.The temporal costs published by authors [1] are shown in Fig. 7. The result from the reproduction experiment are shown in Fig. 8.

		Clustering time	Train time	Test Time	Total Time
K-NN	0	0	46.94	170.185	217.125
	5	11.83	6.152	64.07	82.052
	10	17.03	3.876	48.95	69.856
	15	17.079	2.421	34.897	54.397
	25	22.44	1.426	24.878	48.744
SVD	0	0	19.483	2.607	22.09
	5	11.83	11.92	1.594	25.344
	10	17.03	11.91	1.66	30.6
	15	17.079	11.998	1.57	30.647
	25	22.44	11.912	1.603	35.955

Fig. 7. Time in seconds for KNN and SVD

		Clustering Time	Train Time	Test Time	Total Time
SVD++	0	0	505	108.17	613.17
	5	11.83	473.59	104.8	590.22
	10	17.03	478.56	102.78	598.37
	15	17.079	484.24	107.45	608.769
	25	22.44	483.8	106.6	612.84

Fig. 8. Time in seconds for SVD++

Based on the results, it can be seen that although SVD++ was giving good results with accuracy but the amount of time taken is too much which cannot be seen in the percentage format.

V. New Approach

In Fig. 9 we can see the architecture of the new approach which is different from the author approach by: creating genre vector in a different format and second using other clustering methods like dbscan and hierarchical clustering. The need and discussion for these changes is covered in next sections.

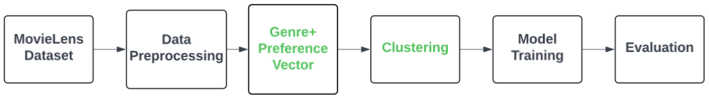

Fig. 9. Proposed Architecture

A. Genre+ Preference Vector:

The author has used only those rating where the user has given more than 3.5 rating. Issue with this approach is that the users that have not rated any movie more than 3.5 will not get selected and thus would not get recommended. For overcoming this issue, three approaches were applied:

1) No Filtering:: In this method we do not filter out any users based on ratings and take all users.Using this approach the results for accuracy can be seen in Fig. 10. The results are roughly the same.The major advantage of this method is that now all the users will get recommended. The reason this method is better because now we are getting the genre preference of all the authors and based on that we are recommending movies to them.

B. No filtering with more negative users:

The major issue is that in the 1M dataset there are only 2 such users that rated all users less than 3.5. So coming to the conclusion if the last proposed method of no filtering is

		RMSE(Orig/Prop)	MAE(Orig/Prop)
KNN	5	0.944/0.942	0.744/0.742
	10	0.964/0.96	0.76/0.75
	15	0.982/0.979	0.773/0.772
	25	1.001/0.996	0.788/0.785
SVD	5	0.887/0.884	0.687/0.684
	10	0.903/0.898	0.7/0.755
	15	0.914/0.91	0.709/0.706
	25	0.923/0.922	0.719/0.718

Fig. 10. Genre+ Preference vector results

better than the author method is not possible. For getting a better idea I have added 500 more users with only negative ratings i.e all ratings below 3.5. This was tested for only 2 cluster values of 10 and 25 and the results are shown in Fig. 12.Although this comparison is not proper as it is done on different datasets as the author method does not contain negative users in recommendation system, but it can be used as a reference.

		RMSE(prop/Repro)	MAE(Prop/Repro)
KNN	10	0.959/0.964	0.756/0.76
	25	0.992/1.001	0.782/0.788
SVD	10	0.920/0.903	0.724/0.7
	25	0.940/0.923	0.742/0.719
SVD++	10	0.904/0.935	0.710/0.723
	25	0.927/0.957	0.730/0.756

Fig. 11. Evaluation of No filtering with more negative users

C. Double Filtering:

In this method, we perform clustering twice.In first phase, we do clustering of user ratings ¿ 3.5 as suggested by author.In second phase, we cluster only the rating that are ¡ 2. After this, if a user was not present in any cluster of phase1 lets say user1 did not got clustered in phase1, then we check that user1 belongs to which cluster in phase2 and which other user is most similar to him.Once we find such a user(user2) we put user1 in the cluster of phase1 of user2. The main idea behind this approach is that we are trying to see in phase 2 which all users dislikes same kind of movies and then cluster them in phase1 with those users. The results for this approach is shown in Fig. 12 and Fig. 13.

		RMSE(Orig/Prop)	MAE(Orig/Prop)
KNN	5	0.944/0.945	0.744/0.744
	10	0.964/0.962	0.76/0.767
	15	0.982/0.983	0.773/0.774
	25	1.001/0.998	0.788/0.789
SVD	5	0.887/0.885	0.687/0.686
	10	0.903/0.907	0.7/0.7
	15	0.914/0.92	0.709/0.716
	25	0.923/0.931	0.719/0.721

Fig. 12. Evaluation of double filtering with KNN and SVD

		MAE(Repro/Prop)	RMSE(Repro/Prop)
SVD++	0	0.681/0.684	0.854/0.869
	5	0.702/0.713	0.895/0.895
	10	0.723/0.744	0.935/0.935
	15	0.732/0.732	0.941/0.953
	25	0.744/0.749	0.948/0.952

Fig. 13. Evaluation of double filtering with SVD++

D. Clustering

For clustering I have tried Hierarchical clustering,adjusted DBSCAN clustering and Gaussian Mixture clustering. For adjusted DBSCAN and GMM clustering, issues were faced for coming up with proper set of parameters.Different parameter values and grid search were applied but no good results were seen. So both of these methods were not used and Hierarchical clustering was performed. For Hierarchical clustering, we can assign the number of clusters.For finding the best number of clusters, dendogram was constructed as shown in Fig 14. Based on the dendogram the optimal number of clusters seems to be 4 or 10. The number of clusters at both the values were evaluated and it was found that Silhouette, Calinski and Davis scores were better for cluster with count 4. So after using 4 clusters,we get the results as in FIg. 15.Hierarchical clustering may have performed better than K-means in the MovieLens dataset because it can more accurately merge users into meaningful clusters based on actual data hierarchies, which can adapt more dynamically to the dataset structure compared to the fixed partitioning approach of K-means.

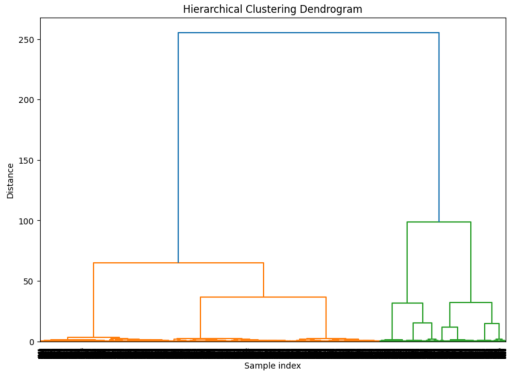

Fig. 14. Dendogram

		RMSE(Kmean/Heir)	MAE(Kmean/Heir)
KNN	5	0.944/0.942	0.744/0.743
	10	0.964/0.96	0.76/0.757
	15	0.982/0.979	0.773/0.772
SVD	5	0.887/0.883	0.687/0.691
	10	0.903/0.902	0.7/0.699
	15	0.914/0.911	0.7090.709

Fig. 15. Hierarchical clustering

VI. Conclusion:

For conclusion, when it comes to clustering it seems that Hierarchical clustering do perform better than Kmeans clustering when it comes for accuracy but Hierarchical clustering has a time complexity of $O(n^3)$ and Kmeans have time complexity of $O(n^2)$. So, Hierarchical Clustering takes time but does give better results.Similarly when it comes to collabarative filtering methods, SVD++ perform better than SVD and Kmeans in terms of accuracy but it takes alot more time as compared to other methods.So ultimately, it will be a trade off between

time and accuracy. If the requirement is just speed, then using kMeans followed by SVD will be giving the best results but if need is for accuracy with some time improvement, then performing Hierarchical clustering followed by SVD++ seems to be a better approach.

For the creating the genre+ vector, there is still not a proper solution that was proposed that has no drawbacks. The proposed solution with no filtering is good as it allows all users to get recommendations but if a user dislikes all horror movies specifically then again this method will suggest horror movies which will be a major drawback.

REFERENCES

[1] V. Botti-Cebriá et al., "Improving the Scalability of Collaborative Filtering Recommendation with Clustering Techniques," 2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology

[2] Jia Rongfei et al., "A new clustering method for collaborative filtering," 2010.

[3] Young Duk, Seo & Kim, Young-Gab & Lee, Euijong & Kim, Hyungjin. (2021). Group recommender system based on genre preference focusing on reducing the clustering cost. Expert Systems with Applications. 183. 115396. 10.1016/j.eswa.2021.115396.

[4] Scott Graham, Jun-Ki Min, and Tao Wu. Microsoft recommenders: Tools to accelerate developing recommender systems. In Proceedings of the 13th ACM Conference on Recommender Systems, pages 542–543, 2019.