Collaborative Filtering System Leveraging Social Network-Based Community Detection

Natural Language Processing and Recommender Systems
Computing for Data Science - UniBz

Julian Senoner 20509
Matteo Zorzi 20156

Abstract

A community enriched collaborative filtering approach has been proposed by Li et al. [1] to overcome typical collaborative filtering recommender sparsity and scalability shortcomings. Our project re-implements Li and Li (2019)'s approach of combining overlapping community detection and a hybrid similarity measure with traditional collaborative filtering. We test our implementation of this approach on the MovieLens 100K dataset and evaluate it for different number of neighbors and train-test split ratios. We confront our approach against user-based collaborative filtering baselines that utilize Pearson or cosine similarity. Even though the community variants achieve similar results for low k, the highest overall correctness is obtained by the Pearson baseline for nearly every setting. This does not let us confirm the results that have been found by Li. Our code and experiment work are included for full reproducibility and for convenience of follow-up work on community enriched recommenders (Github repository).

1 Introduction

In recent years, the rapid growth of online content and user-generated data has significantly increased the demand for accurate and efficient recommendation systems. Among various recommendation techniques, collaborative filtering (CF) has emerged as one of the most widely used approaches due to its simplicity and effectiveness in modeling user preferences based on historical interactions [1].

This report presents the design, implementation, and evaluation of a recommender system based on the algorithmic framework proposed by Xiaofeng Li and Dong Li in their paper on improved collaborative filtering using community detection [1]. The objective of this project is to reconstruct the key components of the algorithm, verify its effectiveness on a benchmark dataset, and assess its performance relative to traditional collaborative filtering methods such as basic user-to-user cosine and Pearson correlation-based collaborative filtering.
Our implementation follows the exact same steps described in the paper and is evaluated using the same dataset. We start by implementing two community detection strategies described in the paper: one based on central nodes and another based on k-factions (maximal cliques). These community structures are then used to reduce the search space for neighbor selection in a user-based collaborative filtering context. In addition, we incorporate an enhanced similarity metric that combines rating-based similarity with category-based similarity, as described in the original work.

The system is evaluated using the MovieLens dataset, focusing on prediction accuracy using standard error metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). We moreover also incorporated evaluation using Precision- and Recall@K. We conduct experiments to observe how performance varies for different train-test split sizes and different neighborhood sizes. Like in the paper, our implementation results are then compared against conventional collaborative filtering methods that use cosine similarity and Pearson correlation to assess the benefits and limitations of the proposed approach.

The remainder of this report is structured as follows: the Background section provides an overview of the original algorithm and its theoretical foundation. The Methodology section details our implementation approach and key design decisions. The Experiments section outlines our experimental setup and presents the results. Finally, we conclude with a discussion of findings, limitations, and potential directions for future improvement.

2 Background

2.1 Collaborative Filtering in Recommender Systems

Recommender systems aim to help users navigate large information spaces by predicting items of potential interest. Among the most popular approaches is *collaborative filtering* (CF), which generates recommendations based on the behavior of similar users. The core principle is that users with similar preferences in the past will likely prefer similar items in the future [2].

Collaborative filtering methods are typically divided into two categories: *memory-based* and *model-based* approaches. Memory-based techniques include *user-based* and *item-based* methods. In user-based CF, predictions are made by aggregating the preferences of similar users. Item-based CF recommends items that are similar to those the user has already liked [2].

User-based collaborative filtering systems works by firstly finding similar users to the target user based on their item preferences. To do this, similarity metrics like cosine similarity or Pearson correlation can be adopted. We then predict a rating for items the user has not interacted yet and recommend the top items with the highest predictions. A standard formula for computing the prediction is for example

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u)} \mathrm{sim}(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} |\mathrm{sim}(u,v)|}$$

Where:

- $\hat{r}_{u,i}$ is the predicted rating that user u would give to item i.
- $\bar{r}_u$ and $\bar{r}_v$ are the average rating of users u and v across all items.
- $N(u)$ is the set of users that are most similar to user u.
- $\mathrm{sim}(u,v)$ is the similarity between user u and user v, which depends on the similarity measure used.

Intuitively, we start by the average rating of user u and adjust it by looking how much similar users liked item i, relative to their average rating. Deviations are weighted by user similarity and normalized by similarity weight. This is a commonly used formula, but there are other formulas that can be used for prediction.

Item-based collaborative way is an approach where instead of searching for similar users, similar items are considered. We then aggregate similar items across all liked items to generate recommendations. It is more scalable than user-based collaborative filtering, as item similarity is more stable over time, but it is also less personalized.

Model-based approaches, such as matrix factorization, project users and items into a shared latent space to capture hidden preferences. Techniques like Probabilistic Matrix Factorization (PMF) [3] and neural collaborative filtering [4] have improved scalability and accuracy in sparse settings.

2.2 Limitations of Collaborative Filtering

Despite its effectiveness, collaborative filtering faces several limitations. Data sparsity arises when users interact with only a small subset of items, making it difficult to compute reliable similarities. The cold start problem occurs for new users or items with limited interaction history [2]. Finally, scalability remains a challenge in large-scale systems where real-time computation is required.

To address these issues, hybrid approaches have been developed that combine CF with content-based methods, demographic data, or social information [5, 6]. One promising direction is the use of community detection from social network analysis.

2.3 Community Detection for Collaborative Filtering

Community detection has emerged as a powerful technique to enhance collaborative filtering (CF) by leveraging structural patterns in user-item interactions or social networks. The underlying intuition is that users often form communities based on shared preferences or behaviors. By identifying these communities and limiting CF operations within them, recommender systems can reduce computational complexity, mitigate sparsity, and improve recommendation accuracy.

This approach has been explored in several papers. One of the earliest and most influential works in this area is by Massa and Avesani [7], who introduced trust-aware recommender systems. Their approach incorporated trust propagation in a user graph to restrict neighborhood selection to a trusted subnetwork, thereby improving robustness and relevance in sparse settings. Other approaches leveraging social community information have been explored in research of the past 15 years (eg. [8] [9] [10]).

These studies collectively show that incorporating community structure can address several inherent weaknesses of collaborative filtering, including sparsity and scalability. Community detection offers an efficient way to reduce the search space for neighborhood-based methods while enabling more expressive user models through group-based reasoning.

2.4 Contribution of Xiaofeng Li and Dong Li (2019)

Building on this foundation, Li and Li [1] proposed an enhanced CF algorithm that integrates overlapping community detection into user-based collaborative filtering.

Two community detection methods are introduced: one based on high-degree *central nodes*, and another using *k-factions* (i.e., maximal cliques).
These communities help constrain the neighbor search space. They also introduce a hybrid similarity measure that combines traditional rating-based similarity with user-category similarity using a weighted linear combination of the two.

The system is evaluated on the MovieLens dataset using MAE and RMSE and compared to traditional baseline CF algorithms. Results show that their approach achieves improved accuracy and efficiency.

3 Methodology

As anticipated, the goal of this report is to describe how we have tried to implement the improved collaborative filtering recommender system proposed by Li et al. [1]. As the paper does not include many technical implementation details, we have tried to best recreate what was described by the authors. We have used the same dataset as in the paper and followed the structure step by step, facing challenges in some parts of the implementation which have been left open for interpretation by the authors. In this section, we are going to describe in detail the algorithmical components and steps of our implementation, along with explanations for our decisions.

3.1 Dataset description

The experiments in this project are based on the MovieLens 100K dataset, a widely-used benchmark dataset for evaluating recommender systems. It is provided by the GroupLens research group at the University of Minnesota [11] and contains 100,000 user ratings applied to 1,682 movies by 943 users.
Each user has rated at least 20 movies, and ratings are integers from 1 to 5, with higher scores indicating stronger preference. The data is collected from the MovieLens website and reflects user preferences based on actual movie interactions.
Most of the ratings are distributed in the higher range (3 to 5) and lower ratings are less frequent.

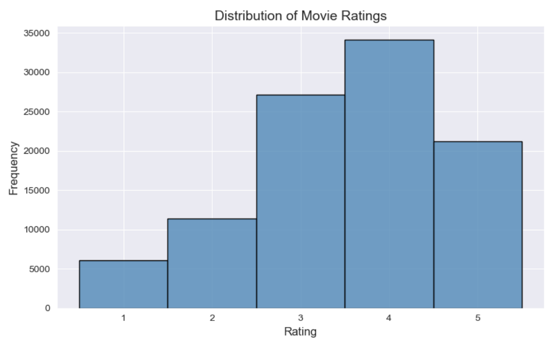

Figure 1: Rating distribution Movielens 100k

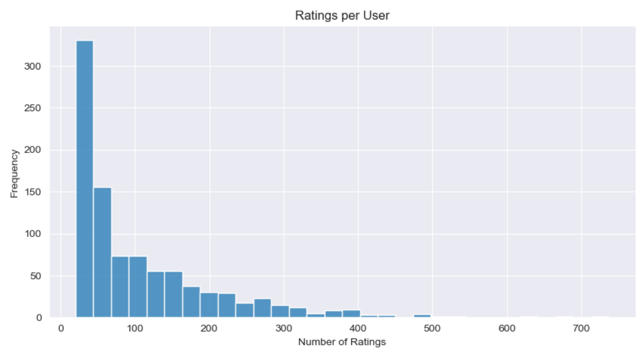

Figure 2: Distribution of ratings per user

The three main data files in the MovieLens datasets are `u.rating`, `u.user` and `u.item`. The ratings file contains all the 100.000 ratings along with the user involved, the movie that was rated and the explicit rating given. The `u.user` file provides demographic information about each user, while the `u.item` file includes metadata for each movie such as genres, titles, and so on.

The dataset is ideal for evaluating collaborative filtering algorithms because it offers a manageable size for experimentation while still presenting real-world challenges such as data sparsity and cold-start issues.

3.2 General Framework

The general framework as proposed by Li[1] and adopted in our project consists of the following main components:

- Community detection from a user network
- Community-aware neighborhood selection
- Hybrid similarity measure
- Prediction and recommendation

We start with the first implementation detail, namely the community detection part.

3.3 Community detection and User Network construction

Traditional user–based collaborative–filtering chooses neighbours from the entire user set, which is brittle under sparsity and noise. We instead extract overlapping user communities from an auxiliary user–user network and restrict neighbour selection to these extracted communities.

The first step in this is nonetheless the construction of a user network from the data at hand. As the paper [1] is not very specific on how the user network is constructed and limits explanations by stating that they "project the user-user network from the user-item rating matrix", we have explored two kinds of networks:

- A similarity graph where an edge links users whose rating-based Pearson correlation similarity exceeds a threshold. This network is meaningful because users who rate items similarly often share interests, allowing the graph to capture implicit social or behavioral connections. It enables effective community detection and improves recommendation accuracy by focusing on like-minded user groups. It is not a simple pre-computation of similarities, as users who are not linked may still be grouped in the same community due to common "friends".
- A demographic network constructed by weighting age, gender and occupation similarity. This type of networks focuses on user attributes and links users based on them.

Both produce a network with around 30 k edges.

As a next step, we need to apply the community detection algorithms on these networks. The algorithms implemented are the ones proposed in the paper, namely one based on central nodes and one based on k-factions.

Central-Node Based Algorithm

The central-node method seeds every community with the highest-degree unlabeled node and greedily adds the neighbor that maximizes the local contribution.

$$q(C \cup \{j\}) = \frac{L_{\text{in}}}{L_{\text{in}} + L_{\text{out}}},$$

where L_{in} is the number of internal edges and L_{out} the number of edges leaving the community. Growth stops once no neighbour can further increase the best q reached so far (Q). A post-processing stage merges any two communities whose Jaccard overlap $S(C_i, C_j) = |C_i \cap C_j|/|C_i \cup C_j| > 0.7$, iterating until convergence.

K-Faction Based Algorithm

The k-Faction algorithm first enumerates all maximal cliques of size $\geq k$ in the user–user graph. Cliques whose overlap (intersection divided by the size of the smaller clique) exceeds a threshold T are merged. Communities are then iteratively fused when their cross-connectivity

$$\frac{\text{edges}(C_1, C_2)}{|C_1|\,|C_2|}$$

exceeds a certain connectivity threshold. Finally, unassigned nodes attach to the community with which they share the most neighbours; isolated nodes form singletons.

We verified that both algorithms retrieve sensible structures on three classical networks—Zachary's Karate Club, American Football, and the Dolphin social graph—by colouring nodes with their detected memberships and highlighting overlaps.

Figure 3 and 4 show the communities detected for the Zachary Karate Club dataset by our implementation. The communities detected are similar to the ones detected by Li [1], with the central node algorithm detecting 3 and the k-faction 2 communities.

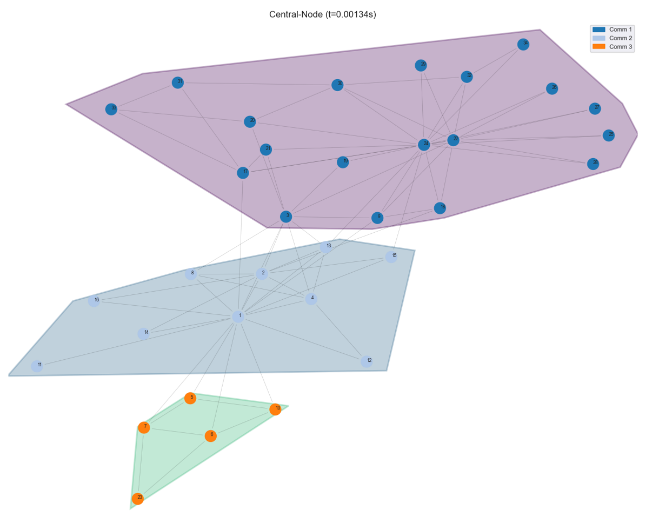

Figure 3: Central node based communitieson the Zachary's Karate Club dataset

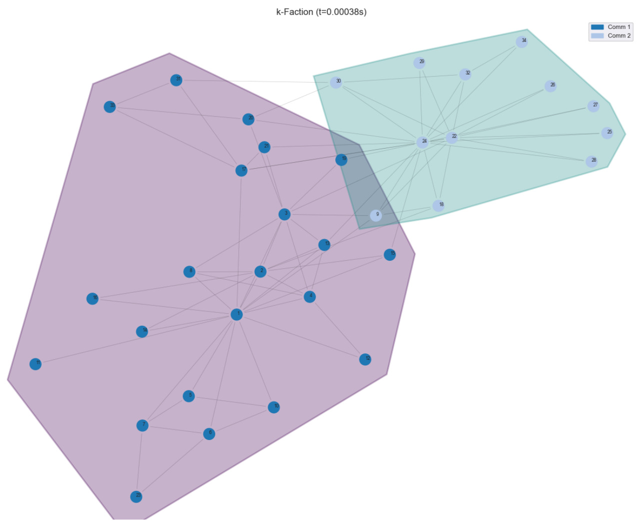

Figure 4: k-faction based communities on the Zachary's Karate Club dataset

The k-Faction variant is two orders of magnitude faster on these toy graphs, yet the central-node strategy tends to return slightly denser covers.

These quick and simple experimental results showed us that the algorithms work and can be applied to our user-user network. As the community detection algorithms are very parameter sensitive and the paper gives no reference regarding either the parameters or the communities obtained, we experimented with parameters in order to end up with more or less balanced communities. For the similarity-based network, we detected a total of 47 communities with the k-faction based algorithm and 53 communities with the central-node based algorithm. For the demographic network, we found 21 and 62 accordingly.

3.4 Hybrid similarity measure

As proposed by Li [1], we adopt a hybrid similarity measure to compute similarities between users. This measure combines two types of similarities between two users u and v.

1. The **rating similarity** $\text{sim}_R(u, v)$: It is computed using the corrected cosine similarity on the user's co-rated movie ratings.

2. The **category similarity** $\text{sim}_{\text{cate}}(u, v)$: Is computed as the cosine similarity between the users' binary genre–preference vectors ob-

tained by thresholding their average rating per genre at 3.5.

These two similarities are then combined by linearly weighting the two terms using a parameter $\lambda \in [0, 1]$.

$$\text{sim}_{\text{hybrid}}(u, v) = \lambda \cdot \text{simR}(u, v) + (1-\lambda) \cdot \text{sim}_{\text{cate}}(u, v)$$

When searching for a user's neighbours, we limit ourselves to community members and adopt this kind of similarity to find the k most similar users.

3.5 Rating Prediction

This section explains how the final rating predictions are made. As explained earlier, given a target pair of user and item (u, i) for which we want to predict the rating, we first collect the set $N(u, i)$ of community neighbors of u that have rated item i.

Keeping the top–k most similar users by $|\text{sim}_{\text{hyb}}|$, the final estimate is predicted by

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u,i)} \text{sim}_{\text{hyb}}(u, v)\,(r_{vi} - \bar{r}_v)}{\sum_{v \in N(u,i)} |\text{sim}_{\text{hyb}}(u, v)|},$$

where $\bar{r}_u$ and $\bar{r}_v$ are the overall mean ratings of user u and user v.

As there were no indications on the specific formula to use for final rating prediction, we have chosen this one as it combines mean-centered collaborative filtering with a hybrid similarity measure that blends rating-based and category-based similarity. This approach adjusts for individual user rating biases, improves robustness in sparse data, and leverages both behavioral and semantic information.

4 Experimental settings

As introduced, the goal is to recreate the experiments from the paper. To do so, we compare our implementation to the results obtained using traditional user-based collaborative filtering techniques based on the cosine and Pearson correlation similarity. These approaches have been implemented using the `surprise` library, leveraging its `KNNWithMeans` function, which uses the same prediction formula as in our implementation, but without the community aspect.

As using the similarity-based user-user network produces better results than with the demographic one, we compare its variants with the traditional collaborative filtering approaches. So in total we compare four different approaches: community-based collaborative filtering once using the central node based communities and once using the k-faction based communities, and traditional collaborative filtering once using cosine similarity and once using Pearson correlation.

The four approaches are compared across different perspectives in order to be able to retrieve interesting behaviors in it. We compare them across different values of k, the number of neighbors considered, and across different train-test splits. Moreover, we also analyze how the results behave for different values of λ, the weighting factor for the hybrid similarity formula.

The parameters considered are the following:

1. Number of neighbors k:
   {5, 10, 20, 30, 40, 50}
2. Train/test split:
   {90/10, 80/20, 70/30, 60/40, 50/50}
3. Weighting factor λ:
   {0.1, 0.2, 0.3, 0.4, 0.5}

This results in a total of $6 \times 5 \times 5 = 150$ configurations. Each configuration was trained and evaluated under identical computational conditions to ensure a fair comparison between community detection and collaborative filtering.

4.1 Evaluation metrics

We assess both rating accuracy and top-K ranking quality:

- **MAE**: average absolute error
$$\frac{1}{N}\sum_{i=1}^{N} |\hat{r}_i - r_i|$$
- **RMSE**: root mean square error - penalizes large mistakes
$$\sqrt{\frac{1}{N}\sum_{i=1}^{N} (\hat{r}_i - r_i)^2}$$

- **Precision@K**: proportion of the K recommended items that are truly relevant ($rating \geq 4$).

$$\text{Precision@K} = \frac{|\text{Recommended}_K \cap \text{Relevant}|}{K}$$

- **Recall@K**: fraction of all relevant items that appear in the top K.

$$\text{Recall@K} = \frac{|\text{Recommended}_K \cap \text{Relevant}|}{|\text{Relevant}|}$$

These metrics enable a fair comparison between plain KNN baselines and the two community-aware variants.

5 Experimental Results

This section is dedicated to carefully analyzing and commenting on the final results obtained from our experiments.

The results do not match the results that Li et al. [1] obtained in their evaluation. For our implementation, the community-based variants were not able to outperform the traditional approaches. Nonetheless, the result of our approach more or less matches the results that are reported in the paper. With our best community-based approach (central-node based communities, k=40, split=80/20), we achieve a MAE of 0.78 would be a good average metric when compared to what has been reported by LI et al. [1]. Nonetheless, using traditional collaborative filtering approaches, we have obtained far better results than the ones reported in the paper. In the paper, they claim to achieve the best MAE at around 0.77-0.78 for traditional approaches, while we get to as low as 0.75. This deviation makes us critically rethink the experimental results obtained by Li [1].

5.1 Neighborhood size

What regards the size of the neighborhood, we were able to gather some interesting insights for our community-based approaches.
Figure 5 underlines that the community-based variants underperform in relation to the traditional variants. For the two community-based variants, the one based on the central nodes performs much better than the one based on k-factions.
KNN-Pearson and KNN-Cosine kept improving as the neighborhood grew: their lowest MAE/RMSE and their best ranking metrics all occurred at $K \approx 50$. In other words, they need a deep neighborhood to stabilize the similarity signal. Central-node based and k-Faction based CF, instead, peaked much earlier. The community-based models reached their MAE/RMSE minima at $K \approx 20$ and achieved their best Precision/Recall already around $K \approx 10$–20. After that point, extra neighbors added noise rather than information.

What is interesting is that for the first 5 to 10 neighbors, the community-based version is able to detect more meaningful candidates than when looking at the whole set of users: This suggests that the approach can benefit from restricting on communities. Nonetheless, when k gets larger, the community-based versions cannot compete anymore.
This trend and pattern can be found not only for this specific 80/20 split depicted in Figure 5, but for the other train-test splits as well.

5.2 Effect of train set ratio

What regards different train-test split ratios, we can see in Figure 6 that, as expected, the error metrics decrease as the training share grows. All four approaches present similar behaviors when faced with less training data, and thus more sparsity. Increasing the test set (reducing training data) consistently harms performance, more noticeably for error-based metrics. Also, here KNN-based methods consistently outperform community-based variants. The same trend is notable for different values of k too.

5.3 Hybrid similarity weighting analysis

We have also tried to analyze whether the hybrid similarity weighting has a significant impact on results. The analysis has yielded us the insight that increasing the λ parameter generally also increases the error rate and that the approach works best when it is kept low at around 0.2. This indicates that the genre-based similarity is less meaningful than the traditional rating-based similarity between users.

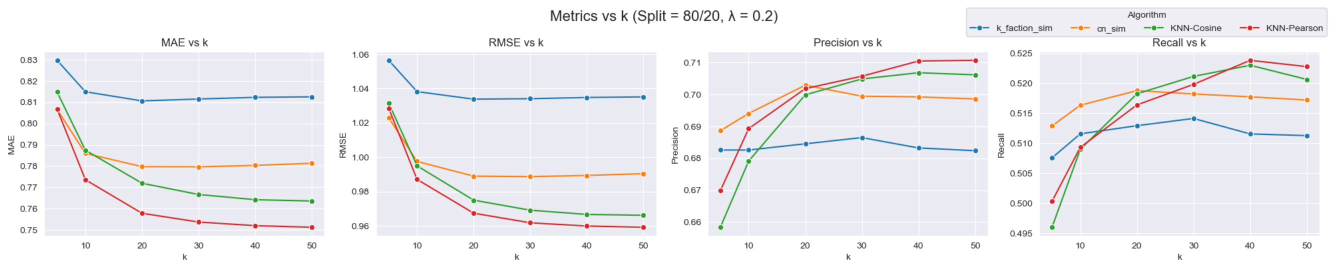

Figure 5: Metrics vs k on an 80/20 split

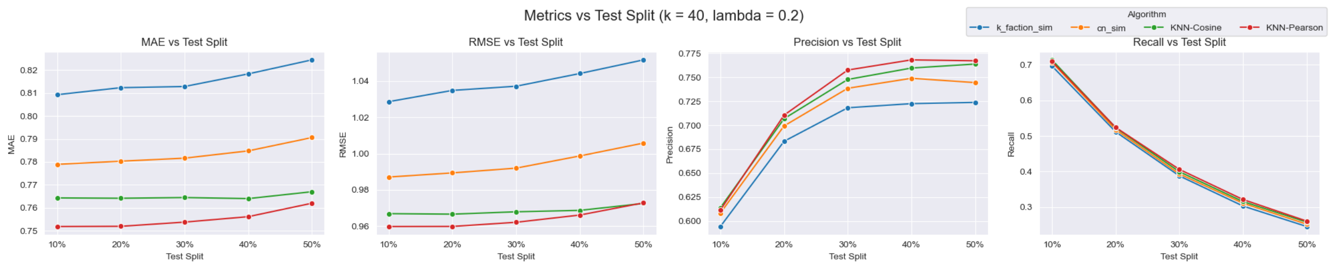

Figure 6: Metrics vs Split ration with k = 40

5.4 Algorithm Comparison

On our test of algorithms, KNN–Pearson narrowly outperforms all other approaches: it had the minimum average RMSE and MAE and the highest overall Precision, although only after being provided a higher number of neighbours (k = 50) and a large 80/20 or 90/10 train-test split. KNN–Cosine performed nearly as good, being only slightly less expressive on this data set. Central Node Based CF was still somewhat competitive, especially when the neighborhood size was small. Nonetheless, it was still outperformed by the traditional variants when this size grew. Finally, k-faction based CF had significantly higher error metrics than the three other approaches. This might indicate that the communities retrieved with this approach are not very meaningful and do not have a positive impact on results.

6 Conclusions and Future Work

The goal of this report is to describe how we have tried to integrate social network community detection with collaborative filtering as proposed by Li et al. [1]. We recreated their approach step by step, following the main structure given by their paper. During the implementation, we faced some open issues that were not explained in full detail in the paper, such as how the network should be constructed and which parameters to use for community detection. Building our project on the MovieLens dataset, we explore two types of user networks, namely one based on similarities between users and one based on users' demographic attributes.

Our attempt to reproduce the community-based collaborative filtering approach gave mixed results. Restricting the neighbor search to the detected communities can help when the neighborhood is small: with $k \leq 10$, the central-node based variant attains results that are better than the plain traditional collaborative filtering baselines. Once the neighborhood grows, however, the classic collaborative filtering approaches outperform the community-based variants. Pearson-correlation based kNN delivers the best overall performance, showing the lowest errors (MAE $\downarrow 4-6$ %, RMSE $\downarrow 5-8$ %) and the highest hit-rate metrics. Our analysis on the weighting factor for the hybrid similarity formula suggests us to focus most of the similarity weighting on the classical user-item rating similarity.

Summarizing, we cannot confirm the results obtained by Li et al [1], even though our implementation achieved comparable results to the results reported in their paper. Nonetheless, in our implementation, the baseline traditional CF methods

produced way lower errors than what has been reported by Li. This makes us critically rethink the experimental evaluation they conducted.
Open issues still are wether we were able to correctly construct a meaningful user-user network and wether the extracted communities are meaningful. This would be the base for such an approach to work, but cannot easily be checked without having any externally supplied information.

This is why we do not want to say in a concluding statement that the approach is not applicable and useful, as we have seen that it can detect relevant neighbours for low values of k. Going forward, future research could explore how this approach works when we have a well-defined user network from which we can extract communities that are certainly meaningful. Moreover, future work could focus on making the community detection algorithms less parameter sensitive or giving suggestions on parameter values for different settings.

All code and experiment logs are publicly available in the github repository, enabling full replication and further improvements of our community-aware models..

References

[1] Xiaofeng Li and Dong Li. An improved collaborative filtering recommendation algorithm and recommendation strategy. *Mobile Information Systems*, 2019(1):3560968, 2019.

[2] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2010.

[3] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. ACM, 2017.

[5] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[6] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[7] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.

[8] Zhoubao Sun, Lixin Han, Wenliang Huang, Xueting Wang, Xiaoqin Zeng, Min Wang, and Hong Yan. Recommender systems based on social networks. *Journal of Systems and Software*, 99:109–119, 2015.

[9] Mengting Gao, Zhongyuan Li, Depeng Jin, Xiaofei Cao, Yong Zhang, and Yucheng Chen. Joint community detection and recommendation with graph regularized matrix factorization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3394–3400, 2018.

[10] Lesly Alejandra Gonzalez Camacho and Solange Nice Alves-Souza. Social network data to alleviate cold-start in recommender system: A systematic review. *Information Processing & Management*, 54(4):529–544, 2018.

[11] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. `https://grouplens.org/datasets/movielens/`, 2015. Accessed: 2025-06-23.