The Performance of Total Variation Regularizer for User Collaborative Filtering

Karolis Mariunas
Supervisor(s): Elvin Isufi, Maosheng Yang, Bishwadeep Das
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Recommender systems (RS) assist users in making decisions by filtering content that the user would likely find relevant. Standard techniques like collaborative filtering exploit user similarities to find the recommendations assuming that similar users are likely to be interested in the same items. On the other hand, graph RS borrow techniques from the field of graph signal processing to predict ratings for items that the users have not seen and utilize a graph representation of user similarities and their corresponding ratings. Recent studies indicated that simple RS could outperform the state-of-the-art RS. Therefore our paper contributes by analyzing the performance of a novel approach to RS - collaborative filtering that uses a total variation graph regularizer. We show that total variation can outperform its predecessor collaborative filtering by reducing the root mean squared error by 4.68%. However, further experiments with top-n recommendations indicated that traditional collaborative filtering could recommend more relevant content than the total variation, which is supported by a 2.20% increase in precision. The results indicate that total variation alone does not add more information to the RS to show noteworthy improvements to existing traditional baselines. However, building on discoveries that even some of the state-of-the-art RS cannot outperform well-defined baselines, a 4.68% increase in accuracy reveals the potential of RS supported by graph regularization.

1 Introduction

The rising popularity and success of applications such as Netflix, YouTube, Amazon and Spotify can be explained by their ability to help users find the content they want, despite having countless options to choose from. Behind the curtains of user-centred applications, we find recommender systems [1] as one of the core functionalities that assist consumers in making choices. Without these systems, users would find it difficult, if not impossible, to explore the vast number of options and content these applications provide.

There are various recommender systems employed in the industry, however, one of the most popular approaches is so-called collaborative filtering (CF) [2]. User-based CF models exploit the similarities between users and how they interact with the items (by giving ratings to the items they like or dislike) to predict the ratings for items that the user has not seen [1, 3]. The underlying assumption of this process is that if some user group has similar tastes and interests, their ratings would likely be similar too. Nearest neighbour methods for CF systems encapsulate these assumptions by selecting only the most similar users for rating predictions [3].

The user similarity can be represented as a graph where the most similar users are connected by an edge. Consequently, the nodes of the graph - users, would contain information about their ratings for items as scalar values, usually ratings from 1 to 5 [3, 4]. Then the problem of predicting missing ratings becomes that of interpolation - using already known ratings and user graph connectivity to infer the values for unseen items. Simultaneously, the field of signal processing on graphs uses simple yet effective techniques called graph filters to interpolate missing values over a graph [5, 6]. One particular approach in this field is to use graph regularizers such as total variation [6] for signal regularization, denoising and other problems.

The rising popularity of recommender systems has led to novel solutions to the problem, such as neural network applications [7–9]. These complex approaches showed promising results compared to traditional solutions, such as collaborative filtering, which are standard baselines in the recommender system research. However, it has been shown that usually, simple approaches in recommender systems can perform just as good if not better than the state of the art that we have nowadays, putting the results of these new techniques in question [10–12]. On the other hand, using graph-based regularizers introduces new perspectives to existing simple recommender systems and has already shown some promising results [4, 13].

This study combines graph-based regularizers explored in graph signal processing [5, 6, 14] with the k-nearest neighbours collaborative filtering recommender system. Our main contribution is to measure and answer *"How does the total variation regularizer perform for user k-nearest neighbours collaborative filtering?"*. Specifically, the analysis is done by considering different design choices and their effects on the overall behaviour and performance of the system. In addition, the performance of our model is compared to the collaborative filtering recommender system.

This paper is organized as follows. Section 2 discusses the general design, components and processes of the graph recommender system pipeline. Section 3 describes the implementation of the total variation regularizer for collaborative filtering, followed by a description of metrics used to evaluate the performance. Section 4 analyses the performance of total variation compared to different baselines. In Section 5, we describe measures taken to ensure research integrity. In Section 6 we discuss the significance of our findings, compare results to other graph recommender systems, and discuss the similarity graph. Finally, in Section 7 we conclude the findings of this study.

2 Methodology

This section provides an overview of the methodology and design choices made. Furthermore, we introduce reproducibility and transparency to the process of the graph recommender pipeline to promote further research and in-depth discussions about the combination of graph-regularizers and recommender systems. Finally, we constructed a graph recommender pipeline to measure the performance of user CF combined with a total variation regularizer, as shown in Figure 1.

2.1 Data

The first step of our pipeline is to collect data on users and their corresponding ratings for some items. This project uses

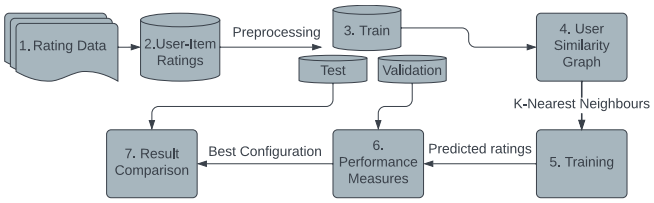


Figure 1: Graph Recommender System Pipeline. The data is firstly preprocessed and sampled into Train, Test and Validation data sets (1,2,3). Afterwards, the Train set is used to construct a User Similarity Graph (4), which is then used to predict unknown ratings (5). Finally, the performance of the predicted ratings is measured (6), and the best-performing model is selected for final measures and result comparison (7).

a benchmark data set MovieLens, containing 100,000 ratings (1-5) from 943 users on 1682 movies [15]. This data set ensures that our recommender system deals with the sparsity problem of rating data, having only 6% of the potential number of ratings while maintaining a relatively small scale compared to other data sets that MovieLens provides (e.g. 1,000,000 ratings). In addition, a smaller data set allows us to run more experiments and get a better performance overview.

The data set is split into random mutually exclusive Train, Validation and Test sets with a ratio of 75/5/20. This is done to ensure that we have enough data to predict the ratings and test whether the algorithm generalises well when predicting the missing ratings found in the validation and test sets. Bellow, we provide a more extensive description of these data sets.

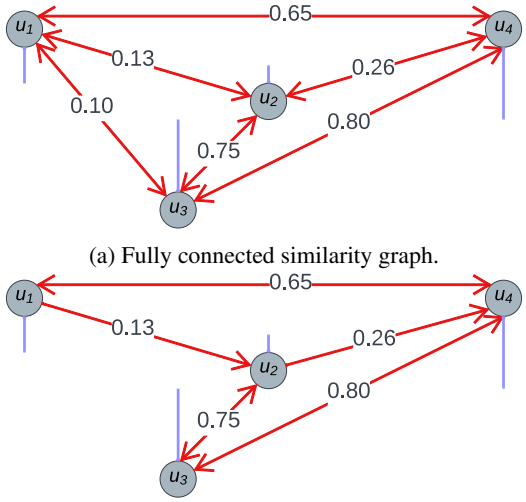
Train Set. The training set is used to calculate user similarities and build their corresponding graph that is needed to predict the ratings for unseen movies. We take 75% of the original data set by sampling random user-item ratings. Furthermore, it ensures that each user has at least 10 ratings, compared to 20 in the original data set.

Validation Set. The validation set is used to measure and compare unbiased results of different configurations of our recommender system. It allows us to select the best performing configuration after model hyper-parameter tuning. This data set contains five randomly sampled ratings per user and corresponds to 5% of the original data set.

Test Set. The test set is used to gather unbiased performance measurements of the final model fitted on the train-validation sets. We use these measurements to compare the results of our model to baselines and other graph-based models [16–18]. The Test set corresponds to 20% of the original data.

2.2 Rating Predictions

After preparing the data, a graph recommender system pipeline is constructed to predict ratings. The pipeline consists of building a user similarity graph, which is further processed into a k-nearest neighbours graph, and finally, predicting ratings.



(a) Fully connected similarity graph.

(b) Filtered k-nearest neighbours directed similarity graph.

Figure 2: Examples of fully connected and filtered similarity graphs. The ratings users give for some item can be visualized as graph signals, indicated by the blue line comming out of the user nodes. Each user in the fully connected graph (a) is connected to everyone else. The simnilarities between these users are respresented as scalar values on the edges. Then, the graph is filtered to contain only k most similar users as seen in (b) where k is 2. The direction of the arrows indicates the direction of the similarity, for example, in graph (b) user u_1 is connected to user u_2, but not the other way around.

We construct a user similarity graph using the user ratings in the train set to calculate the similarity. All the users are connected with a weighted edge, where the weight corresponds to the similarity of the connected users. However, we do not connect users to themselves. An example of the similarity graph is displayed in Figure 2 (a).

Since graph regularization is combined with the k-nearest neighbours (KNN) collaborative filtering, the graph is further processed into a KNN graph, where only the top k most similar neighbours are connected. KNN graphs can be either directed or undirected [14]. Directed graphs allow for directed relationships between two neighbour users, meaning that user u can be a neighbour of user v, but not the other way around, as shown in Figure 2 (b). On the other hand, undirected graphs have bidirectional relationships, meaning that if user u is a neighbour of user v, user v is also a neighbour of user u. In this study, we experiment with both directed and undirected KNN graphs. More information about similarity measures used in this study can be found in Section 3.4.

Finally, the KNN graph and user ratings are used to calculate the predictions. The predictions are then used to reconstruct the user-item prediction matrix, where the missing ratings are filled in with the predicted ratings. This matrix is then used to measure the performance of our method.

2.3 Performance Measurements and Comparisons

Once the rating predictions are gathered, we measure the performance of our graph recommender system configuration using the metrics in Section 3.6. Here, the predicted ratings are compared against actual ratings found in the validation

and test sets. Predictions are further used to construct top-n recommendation lists for each user that contain movies with the highest predicted ratings. Finally, the performance measurements assess to what extent the recommender system can predict unseen ratings and whether the recommendations provided contain movies rated highly by the user.

We then repeat the measurements for different configurations of our recommender system (e.g. number of neighbours, directed/undirected graphs) and select our top-performing configuration for comparison with other recommender systems. Finally, our best-performing recommender system is evaluated using the test set's ratings to measure the overall performance and generalization capabilities. More information about the evaluation procedure can be found in Section 4.

To better understand the results, we want to put them in a broader context. Therefore, we compare the results of our final models with other graph-based models [16–18] and our baselines which are further discussed in Section 4. In our experiments with graph-based recommender systems, we try to align our data, pipelines and processes as closely as possible to ensure that the comparisons and the results are objective and reveal the true nature of the methods [16–18]. It is important to note, however, that although the metrics and data are aligned, the process of measuring the performance might differ.

3 Collaborative Filtering with Total Variation

Since the collaborative filtering problem can be represented as an interpolation of missing ratings over a graph, it allows us to combine it with total variation, a regularization method from the field of graph signal processing [5, 6, 14]. In this study, we use user k-nearest neighbours collaborative filtering. However, it has to be modified before we can integrate total variation. Therefore, we start by explaining total variation, followed by its integration into collaborative filtering. Finally, we end by explaining all of the integral processes and modules of the system.

3.1 Total Variation

Total variation graph regularizer is concerned with measuring the variation of a node signal and its neighbour signals connected to it [5]. The graph $\mathcal{G} = (\mathcal{V}, \mathbf{A})$ can be represented as a set of nodes $\mathcal{V}$ and a weighted adjacency matrix $\mathbf{A}$ that defines the connections between the nodes. Then the graph signal $\mathbf{x}_i$ is a vector that maps some node i to their corresponding signal data. Finally, the total variation of a signal $\mathbf{x}_i$ is defined in [5, eq. (18)]:

$$TV(\mathbf{x}_i) = \|\mathbf{x}_i - \mathbf{A}^{norm}\mathbf{x}_i\|_1 . \tag{1}$$

where the adjacency matrix is normalized by dividing it with its largest eigenvalue as seen in [5, eq. (6)]:

$$\mathbf{A}^{norm} = \frac{1}{|\lambda_{max}|}\mathbf{A}. \tag{2}$$

Total variation in Equation (1) outputs a scalar value corresponding to the 1-norm - the sum of absolute values of the signal variation vector. It can be interpreted as the smoothness of the graph. Low total variation means that the signals of the neighbouring vertices are similar to the signals on the nodes themselves. In contrast, high total variation indicates that the signal is more uneven with more considerable differences between the connected nodes.

3.2 Graph Collaborative Filtering

Traditional user neighbourhood-based collaborative filtering algorithms predict unknown ratings for some *target* user by taking into account the ratings of their nearest neighbouring users [1, 3]. The neighbourhood of the *target* user is specified by their similarity to other users in the system, where the most similar users are thought to be neighbours. However, users tend to rate items differently, and some might be hesitant to assign a high rating to the item they like. Therefore, we have to normalise users' ratings before calculating the similarity. Thus, the components of collaborative filtering are rating normalisation, similarity computation, and neighbourhood selection [3].

Graph collaborative filtering also has these components, however, their representation differs. For example, in a graph $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, users are a set of nodes $\mathcal{V}$ that are connected to their neighbours by a weighted adjacency matrix $\mathbf{A}$, where a value at a position (i, j) corresponds to the similarity of users i and j (0 if the users are not connected). Furthermore, the graph nodes have a signal vector $\mathbf{x}_i$ that contains the ratings that users assign to the movie i, 0 if the rating is unknown. Once these essential components are defined, there are multiple ways to interpolate missing values of the graph signals [1, 3–5, 14].

Our approach to predicting the unknown rating matrix $\mathbf{X}$, where entry $\mathbf{X}_{ui}$ is the rating user u gave to item i, uses graph regularization, commonly utilised to retrieve the true graph signal from noisy measurements [14]. In the case of recommender systems, the observed rating matrix $\mathbf{Y}$ is sparse - contains a lot of missing values, which in this context is noise since we are trying to find the true ratings (signal) that the users would assign to all of the movies. Under the assumption that the observed ratings $\mathbf{y}_i$ for some movie i contain noise, we can model $\mathbf{y}_i$ as in [14, eq. (2.15)]:

$$\mathbf{y}_i = \mathbf{C}\mathbf{x}_i + \mathbf{n}. \tag{3}$$

where $\mathbf{C}$ is a binary selection matrix which masks the unknown values, $\mathbf{n}$ is noise and $\mathbf{x}_i$ is the true signal for some movie i. Following Equation (3), we can estimate the ratings $\mathbf{x}_i$ for some movie i as done in [14, eq. (2.16)]:

$$\hat{\mathbf{x}}_i = \min_{\mathbf{x}_i \in \mathbb{R}^U} \|\mathbf{y}_i - \mathbf{C}\mathbf{x}_i\|_2^2 + \mu f(\mathbf{x}_i; \mathbf{A}). \tag{4}$$

The first term in Equation (4) $\|\mathbf{y}_i - \mathbf{C}\mathbf{x}_i\|_2^2$ measures the difference between our predicted observed rating values $\mathbf{C}\mathbf{x}_i$ and the observed ratings $\mathbf{y}_i$. The second term - regularization, $f(\mathbf{x}_i; \mathbf{A})$, models our assumed prior knowledge about $\mathbf{x}_i$ and the neighborhood of our graph $\mathbf{A}$, and penalizes the predictions which do not follow the assumptions. The trade-off factor $\mu > 0$ controls the importance of our assumptions for the predictions.

This study is concerned with testing the total variation regularizer. We assume that for some movie i, users and their most similar neighbours would have similar ratings. Since total variation measures how smooth the signals are over the graph, we can substitute the regularization term of the Equation (4) $f(\mathbf{x}_i; \mathbf{A})$ with total variation in Equation (1), to enforce that the predicted ratings $\hat{\mathbf{x}}$ follow this assumption:

$$\hat{\mathbf{x}}_i = \min_{\mathbf{x}_i \in \mathbb{R}^U} \|\mathbf{y}_i - \mathbf{C}\mathbf{x}_i\|_2^2 + \mu \|\mathbf{x}_i - \mathbf{A}^{norm}\mathbf{x}_i\|_1 . \quad (5)$$

Equation (5) displays how collaborative filtering interacts with total variation. We use precomputed user-user similarities $\mathbf{A}^{norm}$ as seen in Equation (2), to measure the variation of our predicted ratings $\mathbf{x}_i$. Since we are looking for predictions which minimize Equation (5), we can control how important user-neighbour rating similarities are with the help of μ. For more information on how we find the predicted ratings that minimize Equation (5), consult Section 3.5.

3.3 Data Preprocessing

Since users might use the same ratings to indicate different preferences (4 might seem like a high score to one and mediocre to another), they have to be normalized. We use mean-centering to obtain a more objective insight into whether the ratings are positive or negative [3]. Mean-centering works by subtracting the average rating of some user from all of their observed ratings. Thus, if a rating was below the mean, it would now be negative, indicating that the user did not enjoy the movie.

3.4 Similarity Graph

To construct the adjacency matrix $\mathbf{A}$, we need to calculate the similarity of each user pair. The similarity is crucial since it allows us to indicate how significant are the ratings of some neighbouring users based on their similarity [3]. Furthermore, we can filter the adjacency matrix to contain only the most similar neighbours of each user, which corresponds to the nearest neighbour methods used for collaborative filtering [1, 3].

To calculate the similarities, we use a standard approach called Pearson correlation coefficient [1, 3]. Pearson correlation takes the ratings of movies that both users have rated and indicates how similar the users are on a scale from -1 to 1. However, it is relatively common for users to have only a few rated movies in common, and if the ratings for those movies are very close or even the same, their similarity would be close to one. This is undesirable since it would indicate that the users are highly similar, whereas, in reality, we do not have enough information to conclude that. Therefore, we set the number of minimum movie ratings in common to four and say that for anything below that, the similarity is unknown. After calculating the similarity of all user pairs, we construct the weighted adjacency matrix $\mathbf{A}$, where the value at position (u, v) indicates how similar user u is to user v.

Finally, we process the weighted adjacency matrix to contain only the top k neighbours for every user. For undirected adjacency matrix, we ensure that it is symmetric, meaning that values at positions (u, v) and (v, u) are identical for users u and v. Since some of the user-pair similarities are unknown, we only ensure that the number of connections for each user is not above K. On the other hand, for directed graphs, we connect each user to their top-k most similar users with no restrictions. The similarities of users who are not connected are set to 0. Finally, we normalize our filtered adjacency matrix as seen in Equation (2).

3.5 Rating Interpolation

To get the rating predictions, we iteratively solve Equation (5) for each movie. We use CVXPY, a tool for convex problem optimization, to help us select ratings that minimize Equation (5) [19]. For each movie i, we select the observed ratings $\mathbf{y}_i$ as our initial guess and find the predicted rating vector $\mathbf{x}_i$ for all of the users. However, it is impossible to predict meaningful ratings without using any meta-data for movies that are in the test set but not in the training set (strict cold starters). In such cases, we use the train set to take an average rating of each user as our predictions. Finally, once we have rating vectors for all movies, we construct the user-movie rating matrix $\mathbf{X}$ and add the user rating means to their corresponding ratings.

3.6 Performance Evaluation

To measure how our algorithm performs on unseen ratings hidden during data-splitting, we use root mean squared error (RMSE) for accuracy and precision@n and recall@n to measure our recommendations' effectiveness [1].

RMSE measures how close the predicted ratings $\hat{\mathbf{x}}$ are to the true ratings $\mathbf{y}$ found in the test set $\mathbf{Y}_{ts}$:

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in\mathbf{Y}_{ts}} |\mathbf{y}_{ui} - \hat{\mathbf{x}}_{ui}|^2}{|\mathbf{Y}_{ts}|}}. \quad (6)$$

where $\mathbf{y}_{ui}$ is a true rating given by user u for some movie i, and $\hat{\mathbf{x}}_{ui}$ is the predicted rating.

Precision@n is concerned with the proportion of recommended items that are relevant in a recommendation list of length n [20]. Since users rate differently, we say a movie is relevant if its rating for some user is above their rounded mean rating. Furthermore, for our prediction list, we select the movies with the top-n highest predicted ratings that are also in the test set.

Very similarly, **Recall@n** measures the fraction of relevant items found out of all relevant items in a recommendation list of length n [20]. The score indicates the proportion of all movies retrieved that had positive feedback from some user.

4 Experimental Setup and Results

To assess the performance of our recommender system (RS), we benchmark our model against a set of baselines and other simultaneously implemented graphs RS [16–18]. The baseline models are run in the same environment as the total variation model to ensure that the results are fair and objective. As for other graph RS, the procedure of testing and validating

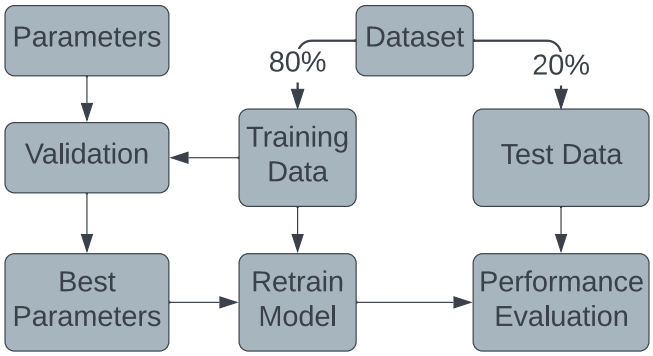


Figure 3: Evaluation Procedure. The full data set is split into training and testing data with a ratio of 80/20. Training data is further split into a validation set used to tune the parameters. Then, the model is retrained on the full training set with the best-performing parameters. Finally, the performance of our model is evaluated using test data.

might differ. However, we test our models on multiple 80/20 train-test data splits and compare the averaged-out results.

Baselines. We use two baselines as a benchmark to assess whether our algorithm can outperform its predecessor or techniques that abuse certain properties to get one of the metrics down:

UserKNN: A traditional collaborative filtering algorithm that uses user neighbour similarities to predict the unknown ratings. As with our model, we use Pearson Correlation to calculate similarities. The model only has one parameter, k - the number of neighbours considered when predicting [1, 3].

User Mean Predictions(UserMP): A simple algorithm that outputs the mean rating of a user for all of their unseen movies. This method tries to artificially reduce the RMSE score based on the assumption that the unseen ratings are close to the mean rating.

4.1 Experimental Setup

We split the data set into 5 80/20 random train-test splits to evaluate our model. For each of the five splits, we select the best-performing set of parameters validated on the validation set (for more information about the data splits, see Section 2.1). The model is then retrained with the best parameters on the entire training set, and the performance is measured using techniques discussed in Section 3.6. Finally, the performance metrics of all the five folds are averaged out to get the overall performance of our model. Figure 3 illustrates an overview of the evaluation procedure.

Since the authors and creators of the MovieLens 100k dataset cannot guarantee the correctness of the data [15], we decided to measure the performance of our RS by taking the average performance of 5 random 80/20 train-test splits over other splitting techniques [21]. Moreover, to account for irregularities in the distribution of data, we ensure that there is no overlap between the test data sets.

For each of the five folds, we tune the parameters on a validation set sampled from the training data. Firstly, the validation data set is used for all hyper-parameter combinations. Secondly, we use grid search to select the parameter combination with the smallest RMSE or precision on the validation set. We use the following parameter sets for grid search:

- **trade-off factor μ:** [0, 0.25, 0.5, 0.75, 1]
- **number of neighbours K:** [5, 10, 15, 20, 25, 30, 35, 40]

Finally, we retrain our model using the best-performing parameters on the whole training data set and measure the performance. This procedure is done individually for every data split. Therefore, the parameters might differ for every fold since they are specifically tuned for that data split. The performance of our algorithm is measured by calculating RMSE as well as precision and recall. For top-n recommendations, we constructed recommendation lists of lengths 5, 10 and 20 to measure how well the methods perform for a low and large number of recommendations. The results retrieved from each fold are averaged and compared to the baselines, and other graph regularization techniques [16–18].

4.2 Results

Accuracy Performance

The results of total variation with collaborative filtering tuned for RMSE (UserTV$^{RMSE}_{undir}$ and UserTV$^{RMSE}_{dir}$ for undirected and directed graphs) are compared against the baselines in Table 1. The results indicate that UserTV^{RMSE} does not act like UserMP and can beat UserMP on all of the given metrics. Therefore, we only compare the results of UserTV^{RMSE} to a UserKNN^{RMSE} baseline.

Table 1 shows that the total variation method with both directed and undirected graphs improves the traditional user collaborative filtering algorithm by 4.68% on the RMSE metric and 1.37% on recall@5. However, it shows no significant improvements for other top-n recommendation metrics. Moreover, all algorithms show stable and consistent results across all five folds, indicated by the low RMSE standard deviation (std). Finally, since both methods were tuned on RMSE, a 4.68% improvement indicates that total variation performs better on RMSE than the traditional algorithms on a relatively small data set with high sparsity.

Furthermore, directed graphs show no substantial increase in any metrics compared to undirected graphs. Although directed graphs contain more connections between users, the additional connections do not add more meaningful information to our predictions resulting in almost identical results. This suggests that undirected graphs can capture all the necessary similarity information while maintaining symmetricity.

To further analyse the performance of total variation, we analyse the performance of different parameters on the validation set. As shown in Figure 4, the RMSE value drops significantly when μ value is 0.25. However, further increasing μ results in minimal changes in RMSE, which smoothly converges to 0.99 when approaching $\mu = 1$. It seems that the total variation value (the second term in Equation (5) is significantly larger than the error-fit (first term in Equation (5)) when $\mu >= 0.25$, which outweighs the error-fit rate and therefore causes the algorithm to focus on minimizing total variation. Consequently, this leads to convergence since no

	RMSE	PREC@5	REC@5	PREC@10	REC@10	PREC@20	REC@20	RMSE std	PREC@5 std
UserMP	1.042	0.542	0.445	0.542	0.649	0.541	0.815	**0.005**	**0.011**
UserKNN^{RMSE}	1.005	0.679	0.511	0.631	0.709	**0.590**	**0.860**	**0.005**	**0.010**
UserKNN^{PREC}	1.018	**0.698**	**0.522**	**0.643**	**0.718**	**0.595**	**0.865**	**0.005**	**0.010**
UserTV$^{RMSE}_{undir}$	**0.960**	0.683	**0.518**	0.634	**0.713**	**0.591**	**0.862**	**0.007**	**0.010**
UserTV$^{RMSE}_{dir}$	**0.958**	0.681	0.516	0.633	**0.713**	**0.591**	**0.862**	**0.007**	**0.010**
UserTV$^{PREC}_{undir}$	**0.961**	0.683	**0.518**	0.634	**0.713**	**0.591**	**0.862**	**0.008**	**0.010**
UserTV$^{PREC}_{dir}$	**0.960**	0.681	**0.517**	0.632	**0.712**	**0.590**	**0.862**	**0.006**	**0.011**

Table 1: Total variation user collaborative filtering (UserTV) average result comparison against baselines (user collaborative filtering UserKNN and user mean predictions UserMP) on the MovieLens 100k data set. Hyperparameters are optimized for Root Mean Squared Error RMSE (UserTV^{RMSE}) and precision@5 (UserTV^{PREC}). Results for undirected (UserTV$_{undir}$) and directed graphs (UserTV$_{dir}$) are also displayed. Results within 1% difference of the best performance for each metric are marked in bold.

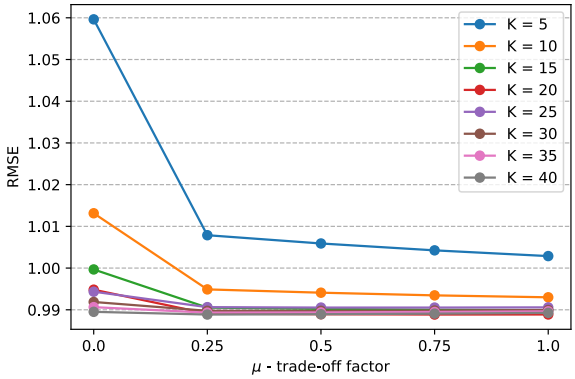


Figure 4: Average RMSE of different number of neighbours K and trade-off factor μ combinations during validation. Y-axis displays RMSE and x-axis μ that is used in Equation (5). The color of the lines indicate the K value, as displayed in the legend.

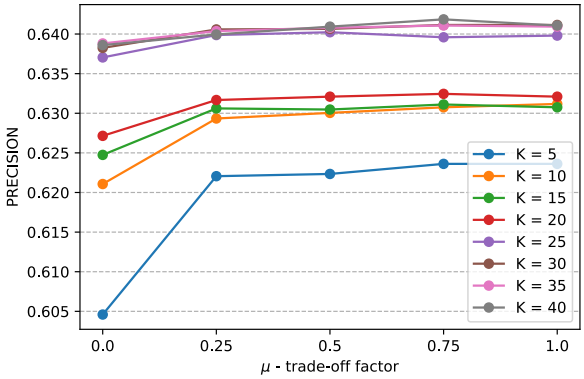


Figure 5: Average precision of different number of neighbours K and trade-off factor μ combinations during validation. Y-axis displays precision and x-axis μ that is used in Equation (5). The colour of the lines indicates the K value, as displayed in the legend.

new information is added, and the total variation score is amplified with $\mu > 0.25$.

We also see a notable performance gain if the number of neighbours k is increased to 10. However, similarly to μ, further adding neighbours seems to converge to 0.99. Moreover, all k values follow the same trend, suggesting that the total variation method is stable, further supported by a low RMSE standard deviation (std).

Top-n Recommendation Performance

We applied the same tests to our models, which were tuned for top-n recommendations. Both directed and undirected graph models were explicitly tuned for precision (UserTV^{PREC}), which measures the proportion of recommended items that are relevant. The results can be seen in Table 1.

Table 1 indicates that UserKNN^{PREC} outperforms UserTV$^{PREC}_{undir}$ and UserTV$^{PREC}_{dir}$ on precision@5 and precision@10 by 2.20% and 1.42% respectively. However, as the size of recommendation lists increases, both methods tend to perform evenly, indicating that UserTV performance increases when the recommendation lists get larger. This shows that UserTV does not outperform the traditional baseline for top-n recommendations. However, UserTV^{PREC} shows a better performance for RMSE, beating UserKNN^{PREC} by 5.70%.

Interestingly, tuning the parameters on precision seems not to affect the precision or any other metrics when compared to UserTV^{RMSE} models that were trained for RMSE. Moreover, directed graphs again show no increase in any metrics, supporting the claim that directed graphs do not add additional meaningful information.

The same convergence patterns that are visible in Figure 4 are also present in Figure 5, where precision starts to converge when $\mu > 0.25$ and $k > 10$. Since higher μ and k tend to increase precision, the parameters chosen for measurements on the test are nearly identical (with slight changes to μ) to those chosen during hyper-parameter tuning for UserTV^{RMSE}. Therefore, total variation performance indicates a balance between accuracy and top-n recommendations, compared to the trade-off in RMSE for better precision in collaborative filtering.

5 Responsible Research

This section discusses the efforts made to ensure that the study is ethical, reproducible and follows the best practices for research integrity [22].

To test and measure the performance of our recommender system, we use the MovieLens data set [15]. The data collection includes demographic data about the users who took part in the study. However, our recommender only requires user-movie ratings. Therefore we removed this information.

In a 2016 study about reproducibility, Baker reports that over 70% of research across multiple science fields have failed to be reproduced [23]. This reproducibility 'crisis' is present in the field of recommender systems as well. For example, a 2019 study about state-of-the-art recommender systems found that only 39% of the work was reproducible [10]. As a result, we think it is our responsibility to give replicable results on which future graph recommender system research can be built.

As our project is part of computational science, we follow some recommendations to provide the necessary tool for reproduction [23, 24]. Firstly, we describe the entire pipeline of our model, introducing any design choices, decisions and other processes that affect the results. Secondly, the source code of this project and the data used is publicly available on GitHub[1] to allow for easy inspection and further work. Finally, the experimental setup and findings are detailed, as this appeared to be one of the issues with recommender system reproducibility [10].

We acknowledge negative results as equally crucial as positive when promoting understanding, research, and more robust progress [25]. To ensure that our results were objective and realistic, we ran multiple experiments for our baselines and total variation model in the same environment, ensuring that each model was trained before evaluation. Although our model could not show significant improvements to a simple collaborative filtering baseline model, which might be viewed as an unfavourable result, we view these findings as essential to publish to promote research transparency and integrity.

We consider transparency and reproducibility essential to scientific integrity and robust, meaningful research. Therefore, we hope that by applying best practices, we can promote further research into the field of graph recommender systems, critical analysis of the methods used and improvements of already existing models through a deeper understanding of the underlying methods.

6 Discussion

6.1 Result Significance

This study aimed to measure the performance of the total variation regularizer in a collaborative filtering recommender system. A graph recommender system pipeline was constructed to predict ratings for movies that the users did not see. The overall performance of our model was measured by calculating the root mean square error (RMSE), precision and recall.

The results indicated that our model outperformed the traditional collaborative filtering recommender on RMSE by 4.68% when both models were tuned for this specific metric. Although both collaborative filtering and total variation models perform well on more extensive recommendation lists, collaborative filtering displayed better results for smaller lists, showing a 2.20% and 1.42% increase in precision@5 and precision@10, respectively. Therefore, we concluded that total variation does not significantly improve the already established baselines.

Recent findings reveal that well-defined baselines, such as user collaborative filtering, outperform reproducible state-of-the-art recommender systems on some metrics [10]. However, both models perform well for more extensive recommendation lists, yet total variation can maintain a lower RMSE. These findings motivate total variation as a more balanced baseline for future research in recommender system graph-regularisation.

However, recommender systems are driven by the context in which they are applied. Our paper applied total variation to a recommender system to measure its general capabilities for accuracy and top-n recommendations. It serves as base research that paves the way for research into more context-driven uses of the total variation. Hopefully, RMSE improvements motivate that total variation has the potential to beat the state-of-the-art recommender systems in some application contexts. Furthermore, more research is needed to establish whether graph regularization is appropriate for other metrics. One such metric is diversity, which assesses the novelty and serendipity of the offered recommendations [1].

6.2 Related Work

This study was split into four projects, where each student measured the performance of the following graph recommender systems:

- **Total variation** regularizer with **user** k-nearest neighbours collaborative filtering (UserTV).
- **Total variation** regularizer with **item** k-nearest neighbours collaborative filtering (ItemTV) [16].
- **Tikhonov** regularizer with **user** k-nearest neighbours collaborative filtering (UserTikhonov) [17].
- **Tikhonov** regularizer with **item** k-nearest Neighbours collaborative filtering (ItemTikhonov) [18].

Here we compare the performance of each of the graph recommender systems mentioned above to our method and baselines. To get an accurate indication of which method performs the best, we decided to use Pearson Correlation with the same parameters (see Section 3.4) together with the same metrics (Section 3.6) and 5 80/20 random train-test splits. The best results for all metrics are displayed in Table 2.

Table 2 indicates that UserTV performs better for RMSE than any other algorithm. However, our baseline UserKNN can beat or be relatively even with the other methods regarding the top-n recommendation performance. Further observing the results in Table 2, we can see that UserTV performs very similarly to the ItemTikhonov method. However, although precision and recall are similar, ItemTikhonov removes all items with less than five neighbours after the adjacency matrix construction, which artificially increases the score for all metrics since bad cases are removed before the performance evaluation. Finally, total variation for item similarity (ItemTV) shows odd behaviour, unable to perform well on any metrics compared to UserTV.

[1] https://github.com/kmariunas/TURP_total_variation_CF/tree/user_user_recommender/user_user_recommender

	RMSE	PREC@5	REC@5	PREC@10	REC@10	PREC@20	REC@20
UserKNN	1.005	**0.698**	**0.522**	**0.643**	**0.718**	**0.595**	**0.865**
ItemTV	1.025	0.582	0.468	0.580	0.676	0.561	0.832
UserTikhonov	1.020	0.682	0.516	0.636	**0.711**	**0.593**	**0.861**
ItemTikhonov	1.006	0.688	**0.521**	0.635	**0.713**	–	–
UserTV	**0.958**	0.683	**0.518**	0.634	**0.713**	**0.591**	**0.862**

Table 2: User collaborative filtering with total variation (UserTV) comparison against other graph recommender systems (ItemTV, UserTikhonov, ItemTikhonov) and collaborative filtering UserKNN (MovieLens 100k data set). The results displayed are the best measurements for each of the metrics. Results within 1% difference of the best performance for each metric are marked in bold.

6.3 Similarity Graph

The similarity graph construction is one possible explanation for the total variation recommender system results. The similarity graph is one of the critical aspects of this recommender system since it is directly used in the prediction Equation (5). We build the graph by first calculating the similarity of each user pair, then connecting only the k most similar users and finally normalizing the graphs' adjacency matrix. We expect significant changes in the performance if the process of constructing the graph is changed to represent the relations between users better.

In this study, we chose Pearson Correlation as our similarity measure since it is one of the most common measures in recommender systems. However, Pearson Correlation depends on the ratings for items that both users have rated. In this study, we say that if users have < 4 ratings in common, their similarity is unknown. Since the user-item rating matrix is sparse (6% density), it is expected that many users would not have enough rated items in common.

There are multiple ways to solve this problem. First, we could try using different similarity computation techniques which calculate the similarity based on all of the user ratings, e.g. Adjusted Cosine Similarity [3]. Another approach would be to consider the significance of the similarity weight, which reduces the similarity if users have only a few items in common [3]. With these changes, we expect an increase in performance since total variation is very much dependent on the quality of the similarity graph.

Another critical aspect of the similarity graph is filtering, where only the top k neighbours are connected in the adjacency matrix. Currently, we ensure that the users have at most k connections for undirected adjacency matrices. However, some users end up with less since we do not allow them to be connected to users who already have k connections, and other similarities are unknown. One possible solution could be to ensure that users have at least k connections, allowing for more if users are dependent on other users who already have k connections. Furthermore, by allowing the connections to be more dynamic, the adjacency graph could contain more meaningful information, which would lead to better results.

Finally, similarity graphs could be constructed for each item by filtering the entire adjacency matrix to contain only the most similar neighbours who have also rated some item i [4, 14]. Collaborative filtering uses the same principle of selecting only the neighbours that have rated some item i when making predictions for that item. Since collaborative filtering produced significant results, beating total variation on the top-n recommendation performance, adding additional item-specific information to our similarity graphs is expected to increase the performance.

7 Conclusion

This paper proposed a graph recommender system that uses existing graph regularization approaches. Specifically, we used total variation with user k-nearest neighbours collaborative filtering to measure the effects on the performance of graph regularization. Firstly, we defined the recommendation problem as a graph interpolation problem. Then, in our proposed recommender pipeline, we constructed a user similarity graph, where only the k most similar users are connected. This graph was then used to interpolate the missing user-movie ratings over a graph. Finally, we used multiple metrics to assess the overall performance of our recommender system. RMSE for accuracy and precision and recall to measure the performance in the context of top-n recommendations.

This project aimed to answer the research question *"How does the total variation regularizer perform for user k-nearest neighbours collaborative filtering?"*. To accurately assess the performance of our method, the recommender system was trained and validated on five different random train-test splits of a MovieLens dataset. The average performance of these five experiments was then compared to two baselines, a traditional user collaborative filtering (UserCF) algorithm and a heuristic approach to minimize RMSE. The results showed that total variation performs similarly to UserCF, with RMSE improvements of 4.68% and a decrease in precision@5 and precision@10 by 2.20% and 1.42%, respectively. In addition, we found that the total variation model can find a balance between all metrics compared to trade-offs between RMSE and precision seen in UserCF performance. Therefore, we conclude that total variation performs as good as the traditional UserCF baseline, with improvements in RMSE.

References

[1] C. C. Aggarwal, *Recommender Systems*, 1st ed. Springer, 2016.

[2] S. Sivapalan, A. Sadeghian, H. Rahnama, and A. M. Madni, "Recommender systems in e-commerce," in *2014 World Automation Congress (WAC)*, 2014, pp. 179–184.

[3] A. N. Nikolakopoulos, X. Ning, C. Desrosiers, and G. Karypis, *Trust Your Neighbors: A Comprehensive Survey of Neighborhood-Based Methods for Recommender Systems*, F. Ricci, L. Rokach, and B. Shapira, Eds. New York, NY: Springer US, 2022. [Online]. Available: https://doi.org/10.1007/978-1-0716-2197-4_2

[4] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Transactions on Signal Processing*, vol. 66, no. 19, pp. 5066–5081, 2018.

[5] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[7] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, ser. SIGIR '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 515–524. [Online]. Available: https://doi.org/10.1145/3209978.3209991

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[9] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015. [Online]. Available: https://arxiv.org/abs/1511.06939

[10] M. Ferrari Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 101–109. [Online]. Available: https://doi.org/10.1145/3298689.3347058

[11] J. Lin, "The neural hype and comparisons against weak baselines," *SIGIR Forum*, vol. 52, no. 2, pp. 40–51, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3308774.3308781

[12] M. Ludewig, N. Mauro, S. Latifi, and D. Jannach, "Performance comparison of neural and non-neural approaches to session-based recommendation," ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 462–466. [Online]. Available: https://doi.org/10.1145/3298689.3347041

[13] K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst, "Song recommendation with non-negative matrix factorization and graph total variation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2439–2443.

[14] E. Isufi, B. Das, A. Natali, M. Yang, and M. Sabbaqi, "Graph filters for processing and learning from network data," Delft University of Technology, pp. 1–19, Sep. 2021.

[15] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, Dec. 2015. [Online]. Available: https://doi.org/10.1145/2827872

[16] L. van Blokland, "Total variation regularisation for item knn collaborative filtering: Performance analysis," unpublished.

[17] S. Monté, "Tikhonov and sobolev regularisers compared to user-based knn collaborative filtering," unpublished.

[18] M. Koper ook geschreven Jansen, "Item-item collaborative filtering via graph regularization," unpublished.

[19] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[20] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, Jan. 2004. [Online]. Available: https://doi.org/10.1145/963770.963772

[21] Z. Meng, R. McCreadie, C. Macdonald, and I. Ounis, *Exploring Data Splitting Strategies for the Evaluation of Recommendation Models*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 681–686. [Online]. Available: https://doi.org/10.1145/3383313.3418479

[22] K. Algra, L. Bouter, A. Hol, J. van Kreveld, D. Andriessen, C. Bijleveld, R. D'Alessandro, J. Dankelman, and P. Werkhoven, "Netherlands code of conduct for research integrity," Sep. 2018.

[23] M. Baker, "1,500 scientists lift the lid on reproducibility," *Nature*, vol. 533, no. 7604, pp. 452–454, May. 2016.

[24] W. I. Matters, "Reproducible research," *Computing in Science Engineering*, vol. 12, no. 5, pp. 8–13, 2010.

[25] D. Mehta, "Highlight negative results to improve science," *Nature*, 2019. [Online]. Available: https://www.nature.com/articles/d41586-019-02960-3