Tikhonov and Sobolev regularisers compared to user-based KNN collaborative filtering

Sérénic Monté
Supervisor: Elvin Isufi and Maosheng Yang
EEMCS, Delft University of Technology, The Netherlands

June 19, 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

Collaborative filtering is used to predict the preference or rating of a user for a certain item. Collaborative filtering is based on the notion that similar users rate similarly. A lot of research is done on how to improve this algorithm, mostly with deep learning. A less investigated field for recommender systems is graph signal processing. Graph signal processing is used to reconstruct a graph signal based on the surrounding nodes. The item ratings of a user can be represented as a graph signal. So it is possible to use graph signal processing as a recommender system. In this paper we investigate how the Tikhonov and Sobolev graph regularisers perform for user-based KNN collaborative filtering. We investigated this by comparing the performance of the collaborative filtering algorithm with the two graph regularisers. We found that the Tikhonov regulariser and the Sobolev regulariser performed very similar to user-based KNN collaborative filtering. This means that the added complexity of the graph regularisers did not increase the quality of predictions we can already make with collaborative filtering.

1 Introduction

We encounter recommender systems on a daily basis, like the advertisements Google suggests when we enter a query, or the videos YouTube recommends after we watch a video [1]. A recommender system is therefore useful since it helps the user with the processing of large quantities of data by suggesting to the user the items with a high rating. For instance there are thousands of movies on Netflix but based on the profile of the user the system tries to recommend the content the user is most likely to consume and thus improving user experience.

Collaborative filtering methods are widely used as recommender systems. These methods are based on the idea that similar users rate similarly [1, 5]. One of the simplest collaborative filtering algorithm is the user-based k-nearest neighbours (KNN) collaborative filtering. This method predicts an unobserved rating for a user u based on the n most similar users who have already rated the item [3]. Please note that in this paper we use n instead of k to indicate the amount of neighbours. A lot of research is conducted on improving this algorithm, for instance with deep learning [7, 9]. A methodology that has not been widely investigated in the field of collaborative filtering is the use of graph signal processing and specifically graph regularisation.

We can convert the data of a recommender system into a graph if we represent a user as a node in a graph, the edges are the similarity between users and the graph signal becomes the measurements of a specific node. Normally a user only rates a few items of the total, in other words the graph signal is noisy. In recommender systems we want to predict the ratings for all the unrated items. Therefore we use graph regularisation to reconstruct the noisy graph signal to predict the unrated items of a user [11]. Reconstructing a graph signal is done by combining the signals from the neighbouring nodes [8]. The Tikhonov regulariser is a special form of a graph regulariser [10] and an advanced form of the Tikhonov regulariser is the Sobolev regulariser [2].

The purpose of this paper is to investigate how the Tikhonov regulariser performs for user-based KNN collaborative filtering. The experiments will consist of comparing the two models on different metrics for the same data set. More specifically we will investigate if the Tikhonov regulariser outperforms collaborative filtering on rating prediction. We will investigate the influence of different parameters for the algorithms. A recent development with the Sobolev regulariser [2] shows promising results so we want to extend the experiment by also testing this regulariser.

The paper is structured as follows. In the next section (2) we describe the research method. The theoretical background will be explained in section 3. The results of the experiments are

provided in section 4. Section 5 contains the discussion on the ethical implications of this research. Finally section 6 contains the discussion on our findings and future work.

2 Methodology

In the experiment we want to compare user-based KNN collaborative filtering, which is our baseline, with the graph regularisers. The experiment is broken down in of three phases:

- Data preprocessing phase
  The preprocessing phase for the baseline consists of calculating the similarity between different users. For the graph regularisers we have to take three steps. The first step is data normalisation, the second step is calculating the similarity between different users and the last step is building the graph. The different steps are explained in the next section.
- Predicting phase
  In the second phase we use the preprocessed data to predict the unseen ratings from the test set. The next section will explain how each algorithm will reach its prediction.
- Evaluation phase
  This phase is used to calculate the performance of the algorithms with different metrics. The metrics we use are the Root Mean Squared Error(RMSE), Recall@k, Precision@k and Normalized Discounted Cumulative Gain (NDCG@k) which will be explained in section 3.3.

The experiments will use the MovieLens100k data set [4]. This data set is collection of 100000 ratings (1 to 5), from 943 users, on 1682 movies. The research group that collected the data preprocessed in away that every user in the set has at least 20 ratings. The data set is randomly split into a training and test set which contain 80% and 20% of the data respectively. The training set is used as input for the algorithms to base their predictions on. The test set is used to evaluate the performance of the algorithms.

3 User-based KNN collaborative filtering and graph regularisers

3.1 Phase one: Data preprocessing

This phase consists of three steps the graph regularisers use all three of them. The user-based KNN collaborative filtering only performs the second step: similarity measure.

Step 1: Rating normalisation

The rating normalisation we use is mean centering. With mean centering we subtract the mean of a user from all their ratings. What this accomplices is that we now show how much a user appreciates an item instead of an absolute number [5]. This is useful since a rating has not the same meaning for every user. Take for instance two imaginary users Alice with an average rating of 4 and Bob with an average rating of 2. They have one movie in common which is movie i. They both rated this movie with a 3. The appreciation of movie i for Alice would be -1 and for Bob +1. In other words Bob likes the movie and Alice dislikes the movie even tough the absolute ratings are the same.

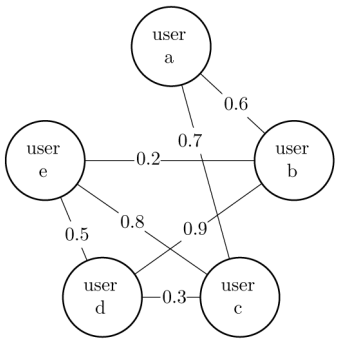
Step 2: Similarity measure

The similarity between users is calculated with Pearson correlation coefficient (equation (1)) [11].This similarity measure compares two users based on items they mutually rated. The set P_{uv} is the set of items both users u and v rated and X_{ui} is the rating of user u for item i.

$$\mu_u = \frac{1}{|P_{uv}|} \sum_{i \in P_{uv}} X_{ui}$$

$$B_{uv} = \frac{\sum_{i \in P_{uv}} (X_{ui} - \mu_u)(X_{vi} - \mu_v)}{\sqrt{\sum_{i \in P_{uv}} (X_{ui} - \mu_u)^2 \sum_{i \in P_{uv}} (X_{vi} - \mu_v)^2}} \tag{1}$$

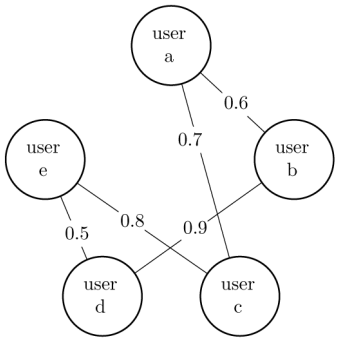
Step 3: Graph building



$$A = \begin{bmatrix} 0 & 0.6 & 0.7 & 0 & 0 \\ 0.6 & 0 & 0 & 0.9 & 0.2 \\ 0.7 & 0 & 0 & 0.3 & 0.8 \\ 0 & 0.9 & 0.3 & 0 & 0.5 \\ 0 & 0.2 & 0.8 & 0.5 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 1.3 & 0 & 0 & 0 & 0 \\ 0 & 1.7 & 0 & 0 & 0 \\ 0 & 0 & 1.8 & 0 & 0 \\ 0 & 0 & 0 & 1.7 & 0 \\ 0 & 0 & 0 & 0 & 1.5 \end{bmatrix}$$

Figure 1: Left an undirected graph with five nodes, the values on the edges are the weights. The missing edges have weight 0. Top right is the weighted adjacency matrix of this graph with the edge weights as entries. Bottom right the degree matrix, the diagonal entries are the row sums of the subsequent rows in the adjacency matrix.



$$A = \begin{bmatrix} 0 & 0.6 & 0.7 & 0 & 0 \\ 0.6 & 0 & 0 & 0.9 & 0 \\ 0.7 & 0 & 0 & 0 & 0.8 \\ 0 & 0.9 & 0 & 0 & 0.5 \\ 0 & 0 & 0.8 & 0.5 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 1.3 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & 0 & 0 \\ 0 & 0 & 0 & 1.4 & 0 \\ 0 & 0 & 0 & 0 & 1.3 \end{bmatrix}$$

Figure 2: Filtered version of graph from figure 1, where every node has at most 2 edges.

The graph presented in figure 1 can be expressed as a Laplacian matrix [11]. To do this we use the weighted adjacency matrix A and the degree matrix D of this graph see figure 1.

Matrix A is a U by U matrix where U is the number of users. If the edge (u, v) exists in the graph then the entry (u, v) of A follows with value w_{uv}. w_{uv} is the weight of edge from node u to v. The Laplacian matrix is constructed by subtracting matrix A from D [11].

For the graph regularisers to work as a recommender system we need a special form of the Laplacian matrix. Because we assume that similar users rate similarly, we need to select the n highest edges per node i.e., the most similar users. To do this we only select the n edges with the highest value per node in the adjacency matrix. In figure 2 we performed this selection for $n = 2$.

3.2 Phase two: Predicting

User-based KNN collaborative filtering

As Aggarwal explained "The basic idea of collaborative filtering methods is that these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items."[1, p. 8]. We measure the correlation between users with the Pearson correlation as explained in phase one. The correlation is not the same for every user, therefore for the prediction we only want to use the n most similar users who have already rated the item. We calculate the prediction by taking the weighted average of the selected users see equation (2).

$$\hat{r}_{ui} = \frac{\sum_{v \in N_{in}(u)} r_{vi} w_{uv}}{\sum_{v \in N_{in}(u)} |w_{uv}|} \tag{2}$$

The nominator consists of the sum of the weights times the ratings. The denominator consists of the sum of the absolute weights. $N_{in}(u)$ are the n most similar users to u who have rated item i. The variable r_{vi} is the rating of user v for item i. The weight w_{uv} is the similarity score of user u to v.

Graph regulariser

The general equation for graph regulariser (equation (3)) consists of two parts. The first part is the error fitting term which minimizes the difference between the unknown graph signal $\mathbf{X}$ and the measurements $\mathbf{Y}$. The second part, $f(\mathbf{X}; \mathbf{S})$, is the regulariser term. Different regulariser can be chosen, like Tikhonov and Sobolev. Each regulariser interprets the surrounding graph measurements in a different way which leads to different graph signal estimations [11].

$$\hat{\mathbf{X}} = \underset{\mathbf{X} \in \mathbb{R}^{U \times I}}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{X}||_F^2 + \mu f(\mathbf{X}; \mathbf{S}) \tag{3}$$

The parameter μ shifts weight from or to this regularisation term. If μ is low the prediction relies more on the measurements (the first term). If the μ is high the prediction depends more on the regularisation (the second term) [10].

$$\hat{\mathbf{X}} = \underset{\mathbf{X} \in \mathbb{R}^{U \times I}}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{X}||_F^2 + \mu tr(\mathbf{X}^\top \mathbf{L} \mathbf{X}) \tag{4}$$

The Tikhonov regulariser, see equation (4), uses $tr(\mathbf{X}^\top \mathbf{L} \mathbf{X})$ as a regularisation term. This term is the signal smoothness measure, which assumes the graph has a smooth signal [11].

$$\hat{\mathbf{X}} = (\mathbf{I} + \mu \mathbf{L})^{-1} \mathbf{Y} \tag{5}$$

The equation (4) is a convex problem and therefore we can derive closed form (5) [11].

$$\hat{\mathbf{X}} = \underset{\mathbf{X} \in \mathbb{R}^{U \times I}}{\operatorname{argmin}} ||\mathbf{Y} - \mathbf{X}||_F^2 + \mu tr(\mathbf{X}^\top (\mathbf{L} + \epsilon \mathbf{I})^\beta \mathbf{X}) \tag{6}$$

The Sobolev regulariser, see equation (6), uses a different regularisation term, namely $\mathbf{X}^\top (\mathbf{L} + \epsilon \mathbf{I})^\beta \mathbf{X}$ [2]. The influence for any value of ϵ, other then 0, is that it adds for every node a self loop with weight ϵ [2]. The effect of the self loop is that the measured signal, of a node we want to predict, is added to the prediction. The parameter β influences how much a similarity score is taken into account [2]. For a value of $\beta > 1$ we give more weight to the nodes who have the highest similarity score. If the value of $\beta < 1$ we give more weight to the nodes who have a lower similarity score. Note that if $\beta = 1$ and $\epsilon = 0$ then the Sobolev regulariser is identical to the Tikhonov regulariser. The Sobolev regulariser has a closed form as well, see equation (7).

$$\hat{\mathbf{X}} = \left(\mathbf{I} + \mu (\mathbf{L} + \epsilon \mathbf{I})^\beta\right)^{-1} \mathbf{Y} \tag{7}$$

We used the closed form of the Tikhonov and Sobolev regularisers in this experiment, equation (5) and (7) respectively.

3.3 Phase three: Evaluation

There are many evaluation metrics available. We want to evaluate the algorithms on different aspects, therefore we use the RMSE, Recall@k, Precision@k and NDCG@k metrics. The RMSE tests the total accuracy of the predictions. The Recall@k tests how many of the relevant items are actually recommended. The Precision@k checks how many of the recommended items are relevant. And finally the NDCG@k metric tests if the items with the highest ratings still have the highest predicted ratings.

The RMSE gives an indication of the average distance between the true ratings and the predicted ratings. The RMSE is calculated with equation (8). The accuracy is calculated on the entire test set where R_{test} is the test set. $f(u, i)$ represents the prediciton made by one of the algortihms for user u on item i. The metric calculates for every rating in the test set the squared difference between the predictions and the true ratings. These differences are summed and divided by the size of the test set, And finally we take the square root.

$$\text{RMSE}(f) = \sqrt{\frac{1}{|R_{\text{test}}|} \sum_{r_{ui} \in R_{\text{test}}} (f(u, i) - r_{ui})^2} \tag{8}$$

For Recall and Precision we first need to define the notion of relevant and recommended items. A relevant item is an item of which the known rating is above a threshold. A recommended item is an item where the predicted rating is above a threshold. The threshold we use for both relevant and recommended items is the user-mean. With these terms we can define Precision and Recall.

Recall@k is the fraction of recommended items in the top k relevant items divided by the total number of relevant items. See equation (9) with N_{rs} the set of items that are relevant and recommended and N_r the set of relevant items.

$$\text{Recall@k} = \frac{N_{rs}}{N_r} \tag{9}$$

Precision@k is the fraction of recommended items in the top k relevant items divided by the total number of recommended items. See equation (10) with N_{rs} is the set of items that are relevant and recommended. The set N_s is the set of recommended items.

$$\text{Precision@k} = \frac{N_{rs}}{N_s} \tag{10}$$

The NDCG@k metric is the normalized version of the Discounted Cumulative Gain (DCG). The normalization is done by dividing the DCG score of the predictions by the DCG score of the true ratings. The DCG score is calculated by summing the relevance of a rating divided by the log of the corresponding position see equation (11). The relevance is the rating for an item.

$$\text{DCG} = \sum_{i=1}^{k} \frac{\text{relevance}_i}{\log_2(i+1)} \tag{11}$$

The Recall@k, Precision@k and NDCG@k are calculated for every user in the test set. For each of these metrics we sum the individual scores and divide them by the number of users. Since the RMSE is calculated on the entire test we do not need to average the result. For RMSE the score of zero is the perfect score, since there is no difference between the predictions and the true labels. The other three metrics return a value between zero and one, where one is the perfect score.

4 The experiments and results

The experiments are conducted on the MovieLens100k data set, therefore the results and conclusion are not generalisable. For the parameters we tested all the algorithms with 8 different number of neighbours n. They range from 5 to 40 with steps of 5. For the graph regularisers we tested the range of μ from 0 to 1.5 with a step size of 0.01. For the Sobolev regulariser we tested 4 values for β: 0.5, 1, 1.5 and 2 and for ϵ we tested 0.1, 1 and 10.

4.1 Comparison of the algorithms

	RMSE	Recall@5	Recall@10	Recall@20	Precision@5	Precision@10	Precision@20	NDCG@5	NDCG@10	NDCG@20
CF	**1.092**	**0.521**	**0.715**	**0.864**	**0.640**	**0.692**	**0.593**	**0.914**	**0.892**	**0.932**
Tikhonov	**1.020**	**0.516**	**0.711**	**0.861**	**0.682**	**0.636**	**0.593**	**0.885**	**0.908**	**0.928**
Sobolev	**1.021**	**0.517**	**0.711**	**0.864**	**0.684**	**0.635**	**0.593**	**0.884**	**0.907**	**0.928**

Table 1: Performance of the user-based KNN collaborative filtering (CF), Tikhonov and Sobolev algorithms. Per metric we indicate with the colour gradient what the best performing algorithm is.

The performance of user-based KNN collaborative filtering, Tikhonov and Sobolev are very similar see table 1. There is not an algorithm that clearly outperforms the other two. If we would add a score of 2 points for best performing, 1 point for second best and 0 points for last. The scores would be 12 for CF, 10 for Tikhonov and 8 for Sobolev. We could say that collaborative filtering outperforms the other two, with the notion that the scores are almost identical. This means that the added complexity introduced by the Tikhonov and Sobolev regularisers have little to no influence. Thus the regulariser do not add any significant predictive value. Please not that the added complexity of the Sobolev regulariser compared to the Tikhonov regulariser is only visible in the third decimal.

The RMSE score of all the algorithms was around one, which means that on a rating scale of one to five, the average distance between the predictions and true ratings was one. A prediction algorithm with such a high margin of error will probably not convert into a very useful recommender system.

Since all the algorithms perform in the same way with the same results, It is safe to assume that an other more dominant factor is in play which leads to this behaviour. So if the algorithms have little predictive power it means that the underlying concept of similar users rate similarly might not hold for this data set. If we take a closer look at the data set, we notice that popular movies (blockbusters) are generally rated more often then the non-blockbuster movies. Because our similarity measure compares users on movies they both have rated, we put more emphasis on the ratings of these popular movies. We suspect that the predictive value of these blockbuster movies is relatively low for a users preference of all movies. For instance we all watched the high rated movie *Inception* but this does not mean that we all rate the same arthouse movies high.

4.2 Collaborative filtering

	RMSE	Recall@5	Recall@10	Recall@20	Precision@5	Precision@10	Precision@20	NDCG@5	NDCG@10	NDCG@20
n	10	25	35	25	40	35	25	40	35	30
score	1.092	0.521	0.715	0.864	0.640	0.692	0.593	0.914	0.892	0.932

Table 2: The scores of the best performing number of neighbours n for user-based KNN collaborative filtering on every tested metric.

The scores of the best performing n is collected in table 2. We can see that the metrics with a value of n above 20 i.e., more then 20 neighbours, gives a better performance. With the exception of the RMSE, where the best performing number of neighbours was 10. Intuitively we would expect that, the more neighbours that are included for the prediction, the better the results will get. This looks the case for Recall@k, Precision@k and NDCG@k. If we look at the RMSE we see that a small number of neighbours perform a little better then higher number of neighbours. An explanation could be that if more neighbours are used, the more noise is generated in the results.

4.3 Tikhonov

	RMSE	Recall@5	Recall@10	Recall@20	Precision@5	Precision@10	Precision@20	NDCG@5	NDCG@10	NDCG@20
n	40	40	40	40	35	40	40	40	40	40
μ	1.50	0.15	0.15	0.22	0.09	0.15	0.22	0.10	0.10	0.10
score	1.020	0.516	0.711	0.861	0.682	0.636	0.593	0.885	0.908	0.928

Table 3: The score of the best performing number of neighbours n and μ for the Tikhonov regulariser on every tested metric.

When we evaluate the best performing parameter sets for the Tikhonov regulariser, table 3. We notice that the regulariser performs best with a large number of n as would be expected. For the μ the algorithm prefers a high μ for RMSE, but a low μ for the Recall@k, Precision@k and NDCG@k metrics.

The relationship between μ and n for RMSE as shown in figure 3 we can see that the performance increases when selecting a higher n and μ. The lines seem to converge to a limit around 1.020. It is important to realise that the performance difference for all the different parameters is marginal. In this case the worst performance, excluding the outliers, is only 0.02 worse that the best performing parameter set. There are several spikes visible where the performance of a single μ was a lot worse than those around it. Upon further inspection of these spikes we noticed that the spikes originated from different train test splits and that if there was a spike in the data it was generally only one. Most of the individual runs of the cross validation did not have these spikes. The fact that all of the different metrics showed these spikes on the same line with the same μ led us to believe that it is an unlucky split in the train test data instead of an error in the calculation of the metrics.

For the Recall@k (figure 4), Precision@k (figure 5) and NDCG@k (figure 6) we combined the results of the different k and n values. We found some interesting results. As we can see the results where all grouped together by k. The order of performance for k was from high (k = 20) to low (k = 5) for Recall@k and NDCG@k, for Precision@k the order was the other way around. For every group of k we see the same relation between the different n and μ. The performance was best with a high n and the maximum was reached at a low μ. It is important to note that, although a maximum is reached at lower μ, the performance is almost identical for the other μ after the maximum. If the μ is zero the regulariser returns the input of the user we want to predict plus the mean of that user. In other words we return the user mean for every rating we want to predict. As soon as we add the regulariser term even in a small portion the performance has a steep ascend, but adding more regulariser does not translates itself into a significant better performance.

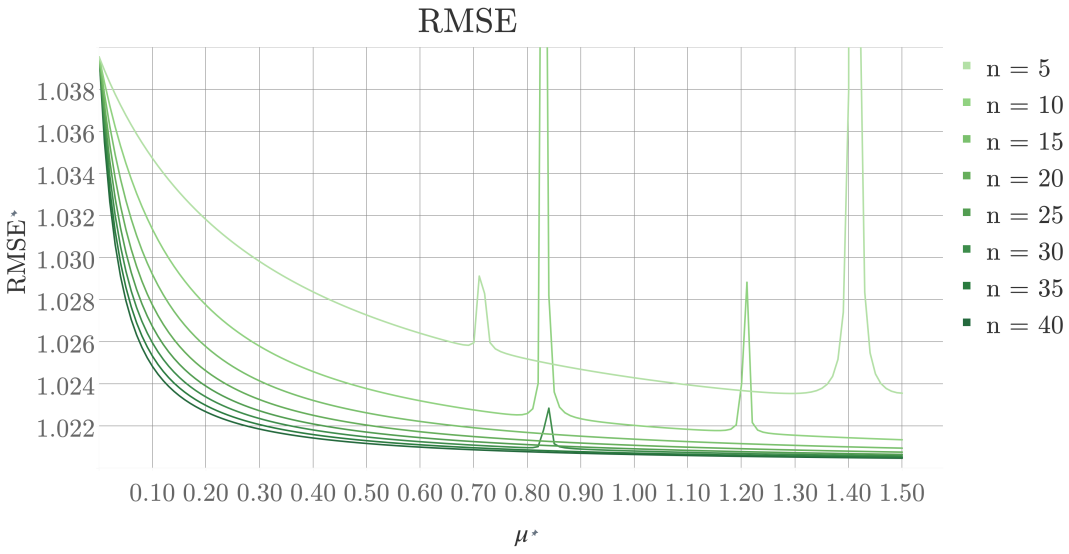


Figure 3: Relation between the performance of the parameters μ and n for the Tikhonov regulariser on the RMSE metric.

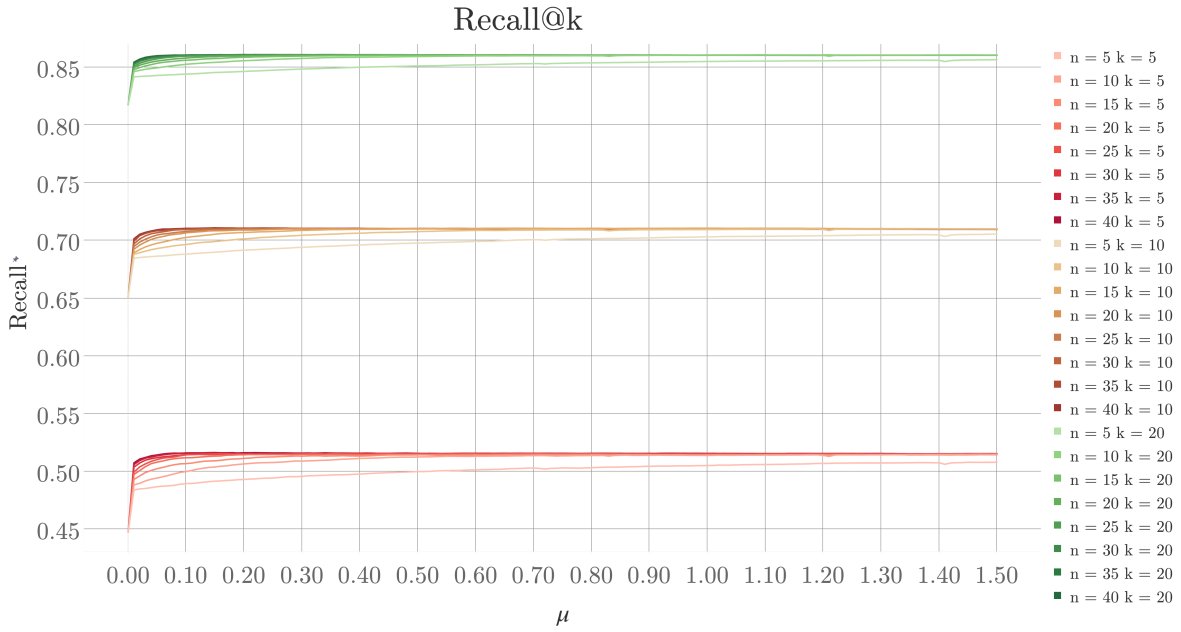


Figure 4: Relation between the performance of the parameters μ and n for the Tikhonov regulariser on the Recall@k metric.

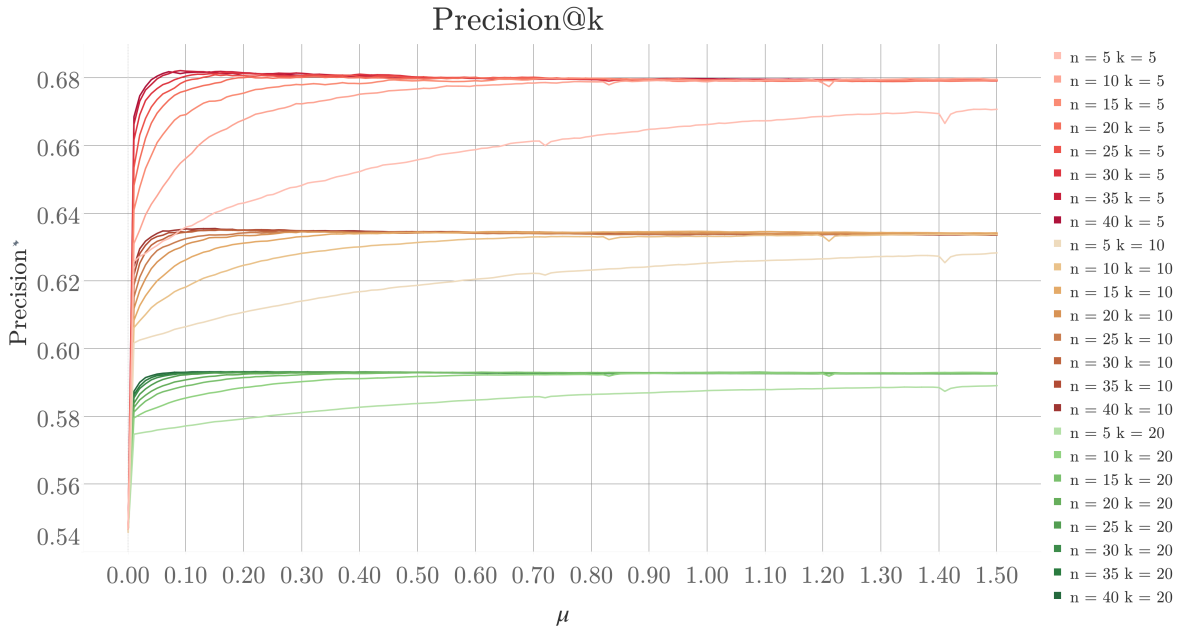


Figure 5: Relation between the performance of the parameters μ and n for the Tikhonov regulariser on the Precision@k metric.

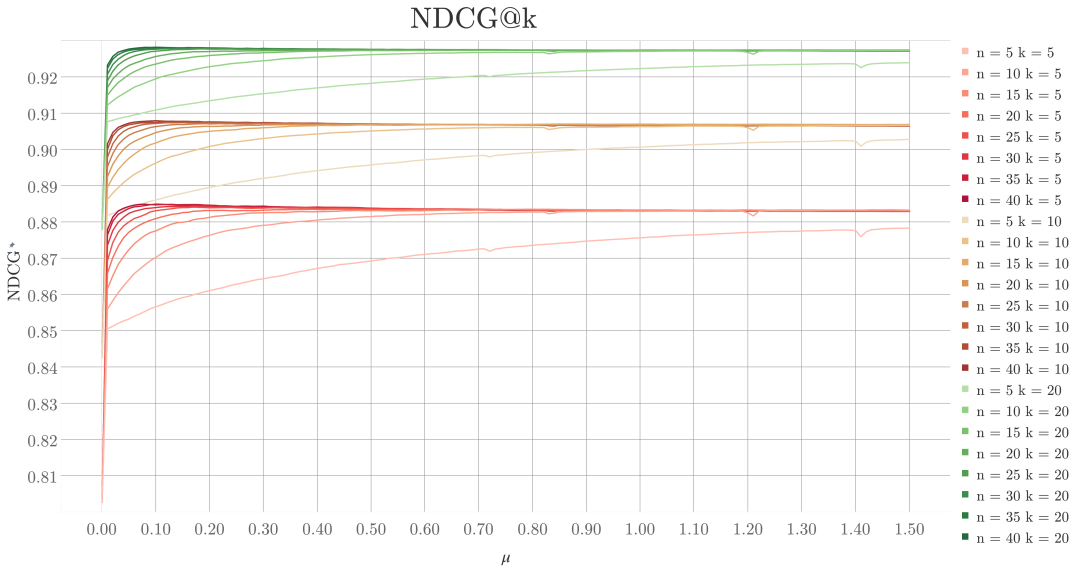


Figure 6: Relation between the performance of the parameters μ and n for the Tikhonov regulariser on the NDCG@k metric.

	RMSE	Recall@5	Recall@10	Recall@20	Precision@5	Precision@10	Precision@20	NDCG@5	NDCG@10	NDCG@20
n	40	40	40	40	40	25	35	40	35	35
μ	0.38	0.11	0.60	0.91	0.03	0.07	0.05	0.03	0.01	0.12
β	1.5	1.0	0.5	1.0	1.5	1.5	1.5	1.5	2.0	1.0
ϵ	0.1	1.0	0.1	10.0	0.1	0.1	0.1	0.1	1.0	1.0
score	1.021	0.517	0.711	0.864	0.684	0.635	0.593	0.884	0.907	0.928

Table 4: The score of the best performing number of neighbours n, μ, β and ϵ for the Sobolev regulariser on every tested metric.

4.4 Sobolev

Table 4 shows the values of the best performing parameter set for every metric. Just like Tikhonov we also find a preference for a higher n. It looks like there is a slight preference for lower μ. For the β and ϵ all the tested values are represented at least once in table 4. There is a slight preference for β 1.5 and ϵ 0.1.

The relation between μ, β, ϵ and n for the RMSE metric is shown in figure 8. There does not seem to be a clear relation between the n and μ. We highlighted the best performing parameter set. We can see that the μ has little to no effect on the results after the first steep descent. If we change the colouring from grouping by n to a grouping by ϵ we see that the ϵ dictates how well the algorithm will perform, see figure 9. A lower value of ϵ performs better. As we have described in section 3.2 the ϵ adds self loops to the graph, this means that self loops have a negative effect on the RMSE score. Within the ϵ clustering we also see some clustering of β however there is no clear order.

For Sobolev the analysis of the other metrics more challenging. With 8 values of n, 150 for μ, 4 different β, 3 values of ϵ and 3 different k, we generated 43,200 test results for every

metric. Grouping the results together per metric, resulted in unreadable figures. Therefore we first analysed if there was a different trend for the different k, this was not the case. We chose to show the k = 10 metrics in this section and the k = 5 and 20 are attached in appendix A. The order of performance for the different k was identical to that of Tikhonov.

If we look more closely to the individual plots of Recall@10 (figure 10), Precision@10 (figure 11) and NDCG@10 (figure 12) we find that the relation between n and μ is not as prominent as with Tikhonov. In general it still holds that a higher n performs better than a lower n and the influence of μ is after a steep ascend almost negligible, this is identical to Tikhonov. Unlike RMSE we did not find a strong relation or clustering for the different β and ϵ values.

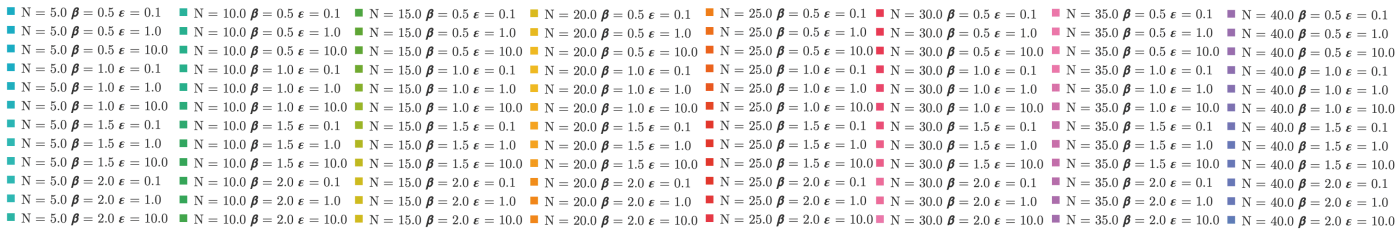


Figure 7: The legend for the Sobolev performance figures. Due to the size it is added once and is applicable for figure 8, 10, 11 and 12

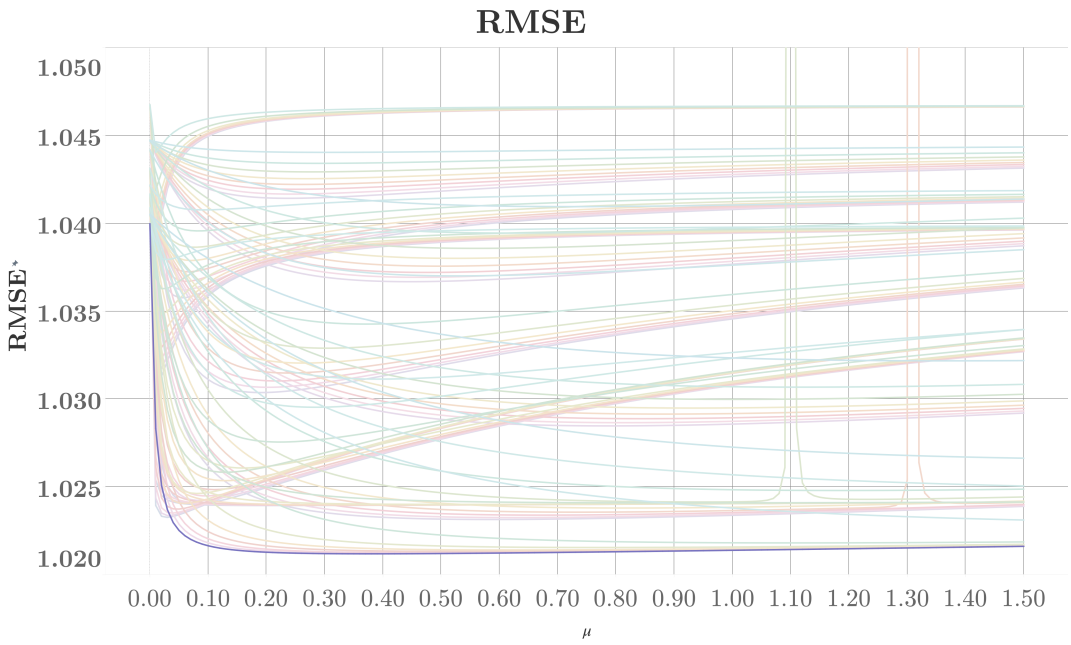


Figure 8: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the RMSE metric. The highlighted line is the parameter set with the best performance.

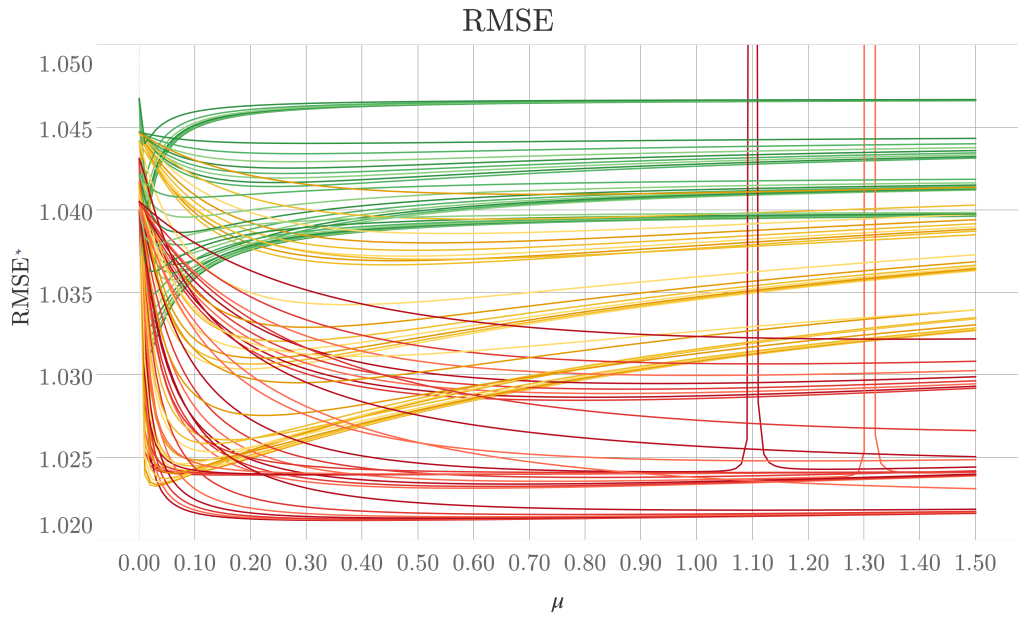


Figure 9: The RMSE performance for Sobolev with the colours coded to the ϵ value. Red is $\epsilon = 0.1$, Yellow is $\epsilon = 1.0$ and Green is $\epsilon = 10.0$

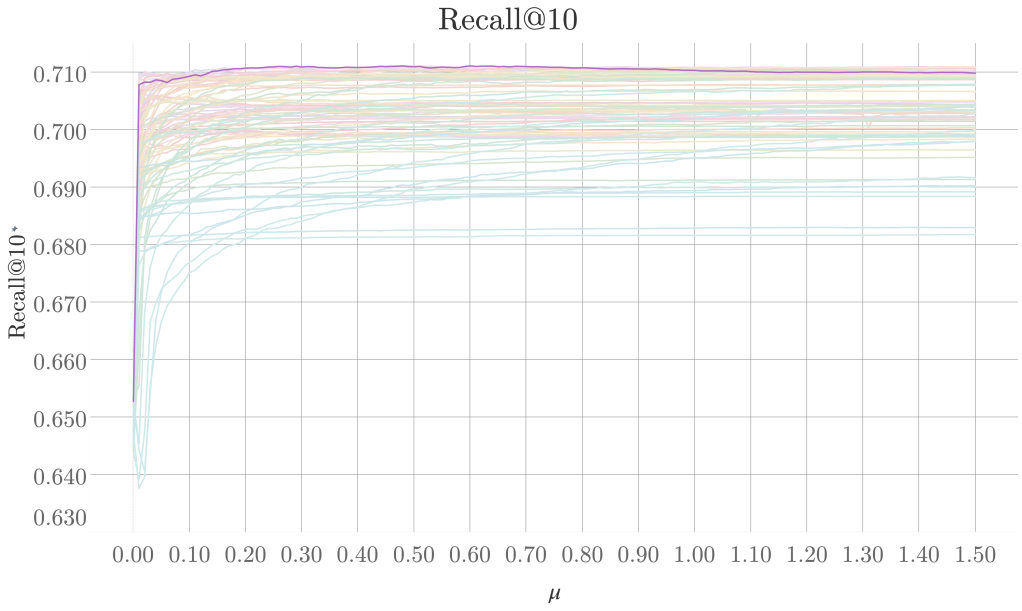


Figure 10: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Recall@10 metric. The highlighted line is the parameter set with the best performance.

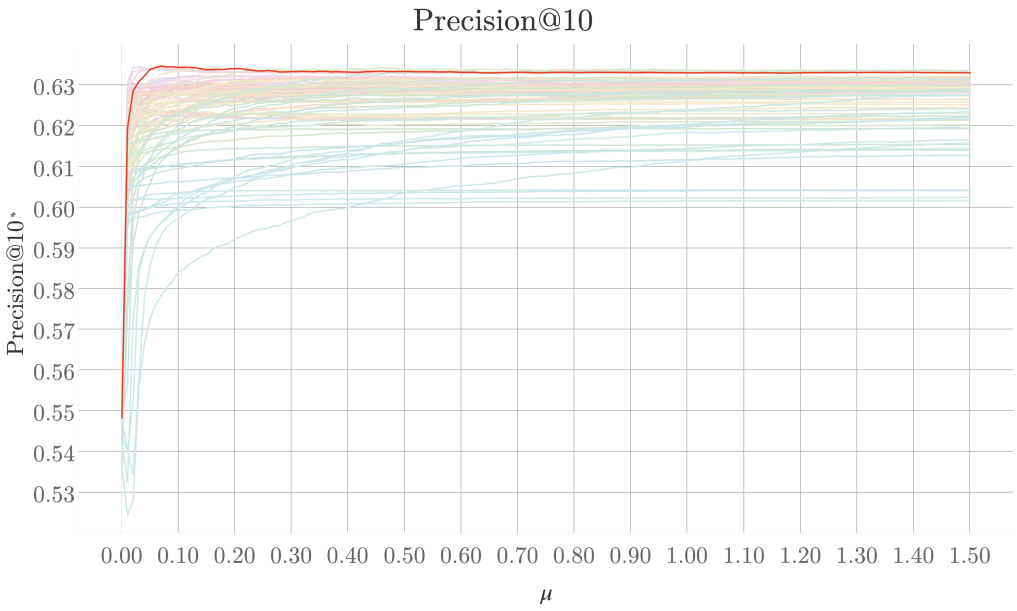


Figure 11: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Precision@10 metric. The highlighted line is the parameter set with the best performance.

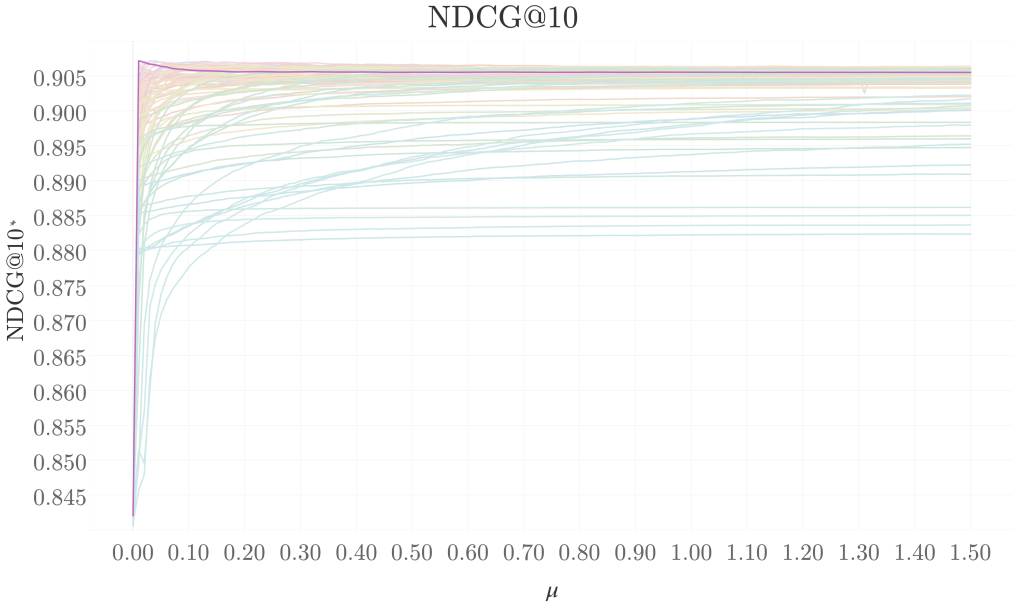


Figure 12: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the NDCG@10 metric. The highlighted line is the parameter set with the best performance.

5 Responsible research

Ethics are a topic every researcher has to think about when conducting experiments. In this research we did not do experiments on people, but used precollected user data. The data we used is the MovieLens100K data set collected by the Grouplens group of the university of Minnesota. They published this dataset and it is freely available for anyone to use as long as it is not for commercial purposes, acknowledge the use of the data set and not state or imply any endorsement form the University of Minnesota or the GroupLens Research Group [4]. This data set is often used for research in the field of recommendation systems.

Reproducibility of our methods is another topic that is important. As stated in [7] reproducibility of recommendation algorithms should be fairly easy, however the method of hyperparameter training, preprocessing and evalutating is most often not shared with the prediction algorithm. This makes it difficult to reproduce the experiment performed. To make sure our experiment is reproducable all the source code is made available to anyone who is interested[1].

The hyperparameter tuning uses random test an training splits. This makes the experiment not fully deterministic, since there is no way of guaranteeing that the same data splits are used. We circumvented this problem by taking the average of ten runs with different random splits.

6 Discussion

This paper researches how the Tikhonov and Sobolev graph regulariser perform for user-based KNN collaborative filtering. We tried to answer this question by implementing the algorithms and test them both on the MovieLens100k data set with different metrics. The metrics used to test the performance are RMSE, Recall@k, Precision@k and NDCG@k. The RMSE is used to measure the performance on the complete test set, whereas the other metrics only look at the k highest ratings per user in the test set.

The conducted experiments showed that the performance of the algorithms are very similar. The Tikhonov and Sobolev graph regularisers perform almost identical to user-based KNN collaborative filtering on the MovieLens100k data set. This means that the added complexity of the algorithms do not improve the performance of the predictions. The performance is so similar that we suspect that there is a more dominant factor at play which hampers the algorithms in their predictions. Therefore we also looked at the MovieLens100k data set. We found that the popular movies where rated more often then the non-popular movies. The similarity score is based on the movies users have in common. Because more popular films are rated more often, the similarity score puts heavy emphasis on these popular movies. This means that the weight of the popular movies is dominant in the predictions. As we see hardly any difference in predictive power between the different algorithms we have reason to suspect that the notion that similar user rate similarly is not applicable for rating all movies.

	RMSE	Recall@5	Recall@10	Precision@5	Precision@10	NDCG@5	NDCG@10
CF	**1.066**	**0.480**	**0.683**	**0.600**	**0.580**	**0.834**	**0.869**
Tikhonov	**1.006**	**0.521**	**0.713**	**0.688**	**0.635**	**0.889**	**0.910**
Sobolev	**1.006**	**0.520**	**0.713**	**0.688**	**0.635**	**0.889**	**0.910**

Table 5: The results of Melle for item-based similarity

[1] https://github.com/BachelorThesisRecSys/BachelorThesisRecSysUserBasedTikhonov

We also looked at the results of Melle [6]. His results are shown in table 5. This research focuses on item similarity whereas we focused on user similarity when using the same algorithms. If we look at the results we see a similar outcome. This means the different algorithms perform comparable and the RMSE is relatively high as with our results.

Our results give rise to further research.

- We limited the scope of the experiments to a single data set. More experiments on different data sets are needed to investigate if the results of this paper are generalisable or not. We leave these experiments for future work.
- The experiments in this paper are limited to global graph regularisers. The use of local graph regularisers can be used to check if the predictive power increases if the local variability of the graph is taken into account. This limitation can be addressed in future research.
- Lastly we suspect that there is a dominant factor that limits the usability of the investigated graph regularisers. It will be interesting to research if an other form of similarity between users that will take into account the dominant influence of popular movies will enhance the usability of the graph regularisers as recommender systems. We leave this as a direction for future work.

A Sobolev extra graphs

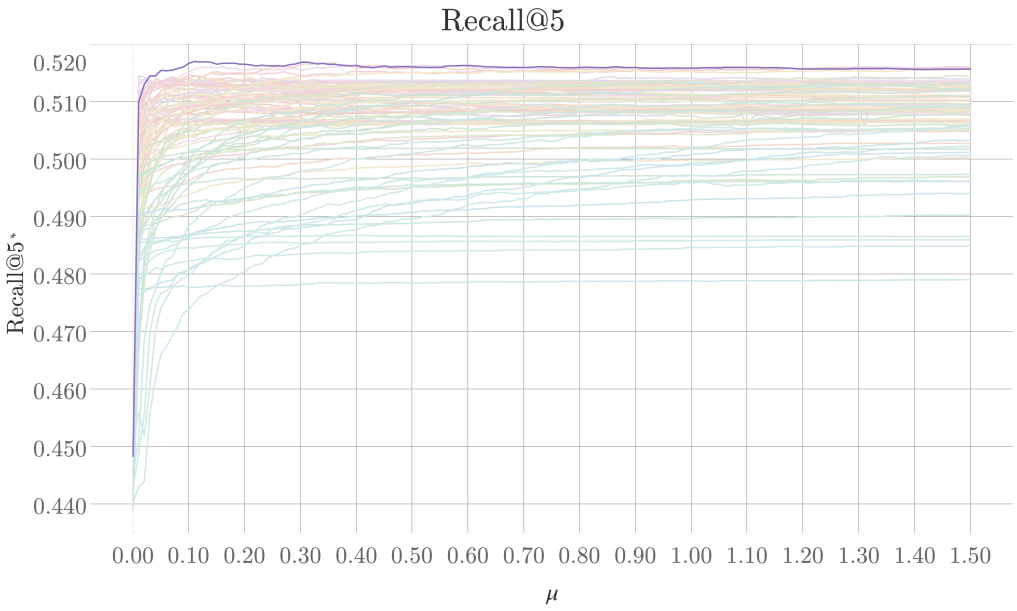


Figure 13: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Recall@5 metric. The highlighted line is the parameter set with the best performance.

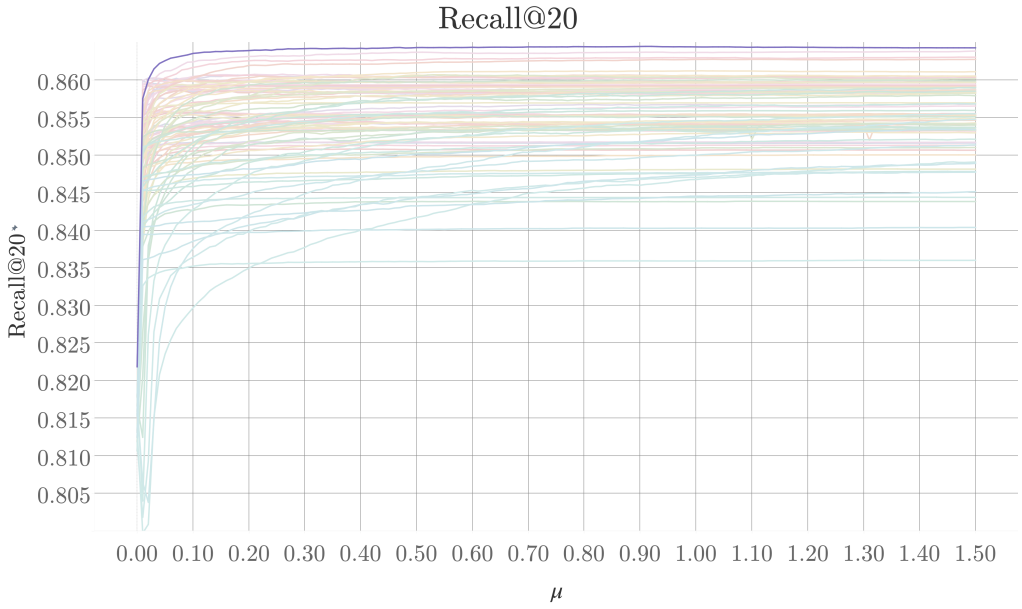


Figure 14: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Recall@20 metric. The highlighted line is the parameter set with the best performance.

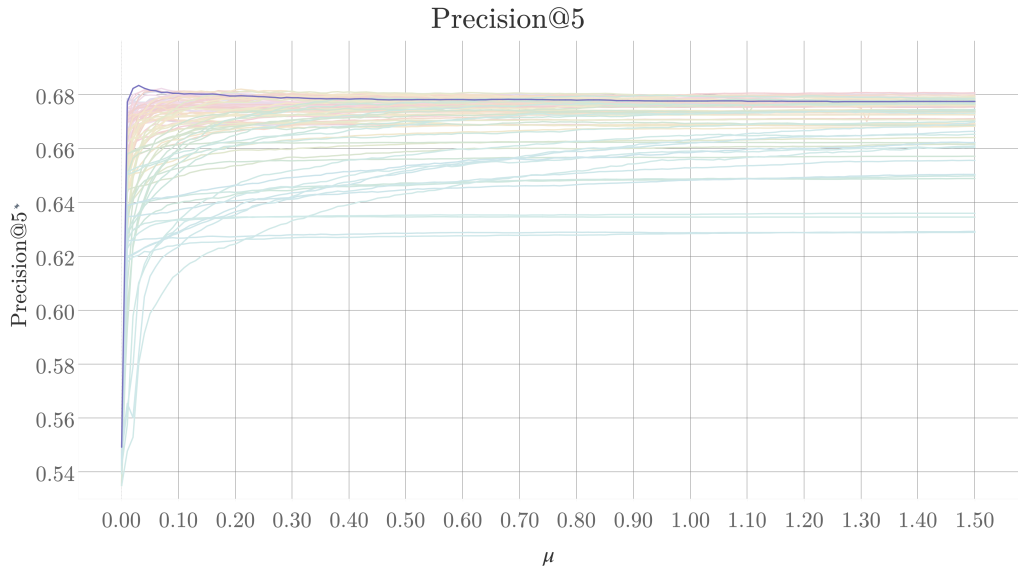


Figure 15: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Precision@5 metric. The highlighted line is the parameter set with the best performance.

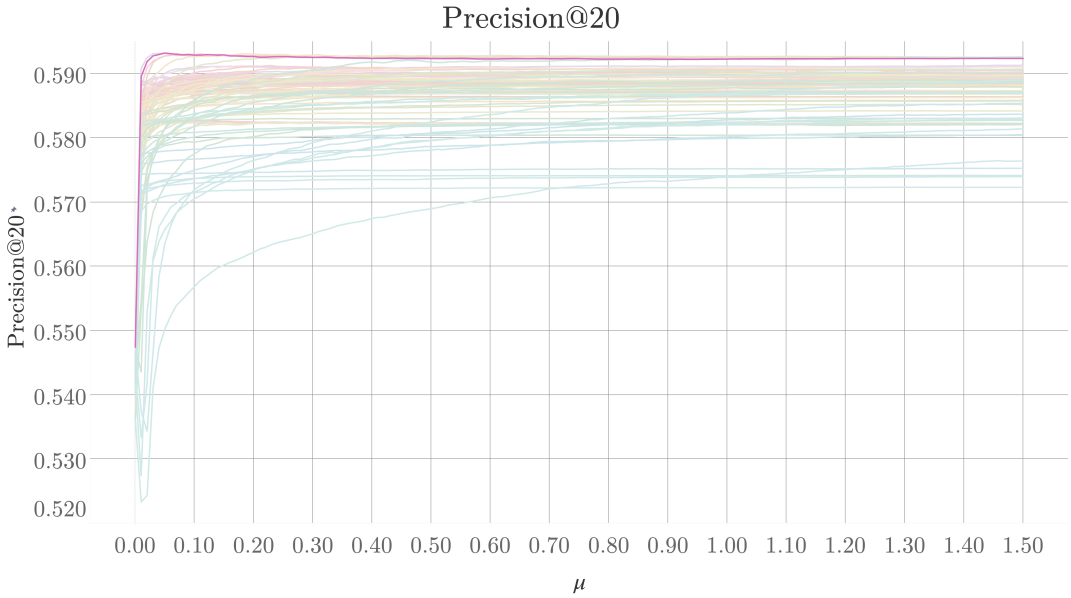


Figure 16: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the Precision@20 metric. The highlighted line is the parameter set with the best performance.

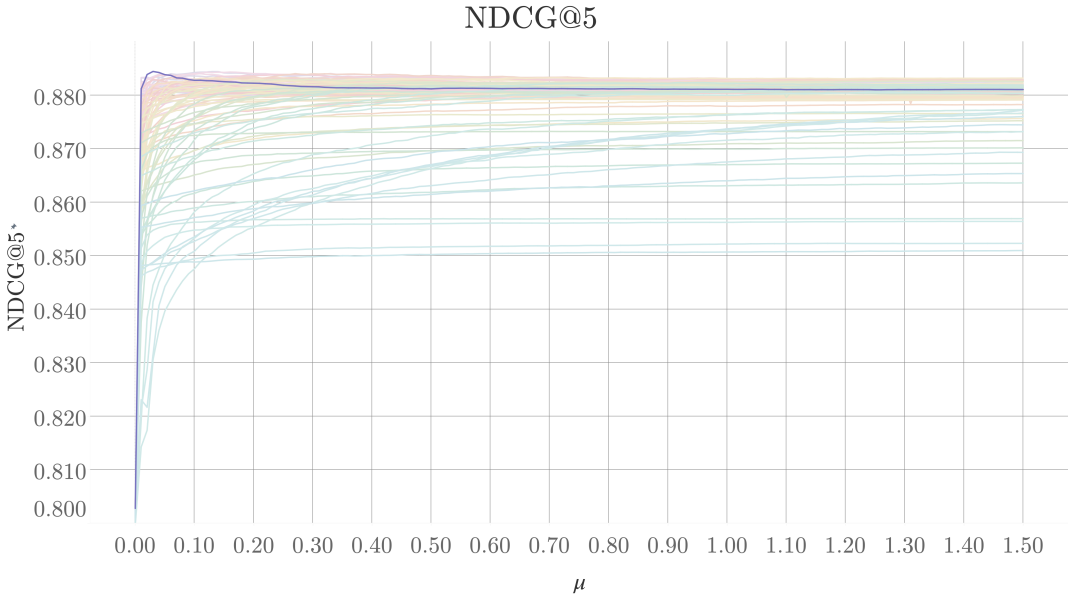


Figure 17: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the NDCG@5 metric. The highlighted line is the parameter set with the best performance.

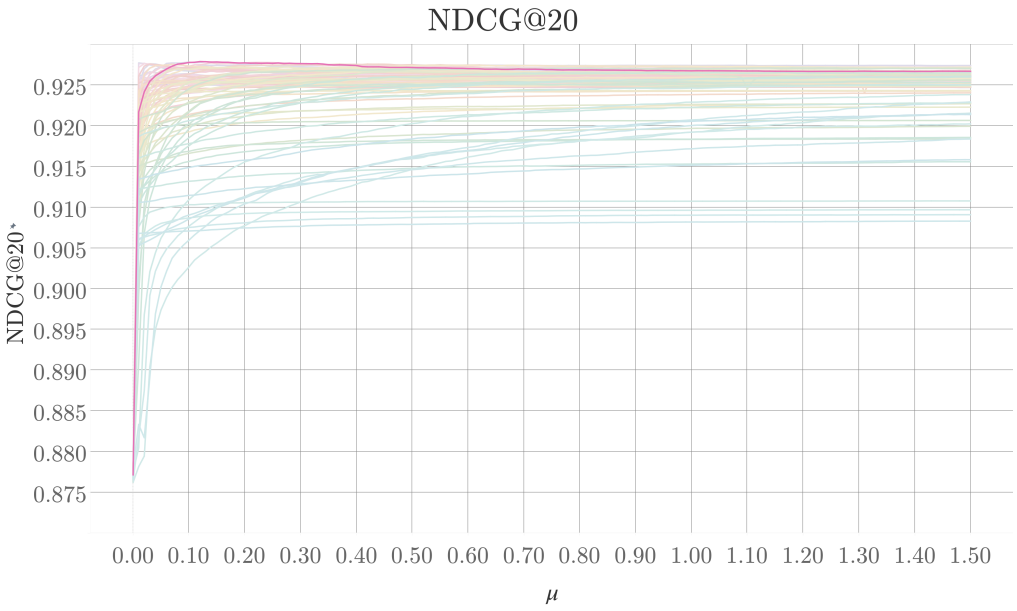


Figure 18: Relation between the performance of the parameters μ, n, β and ϵ for the Sobolev regulariser on the NDCG@20 metric. The highlighted line is the parameter set with the best performance.

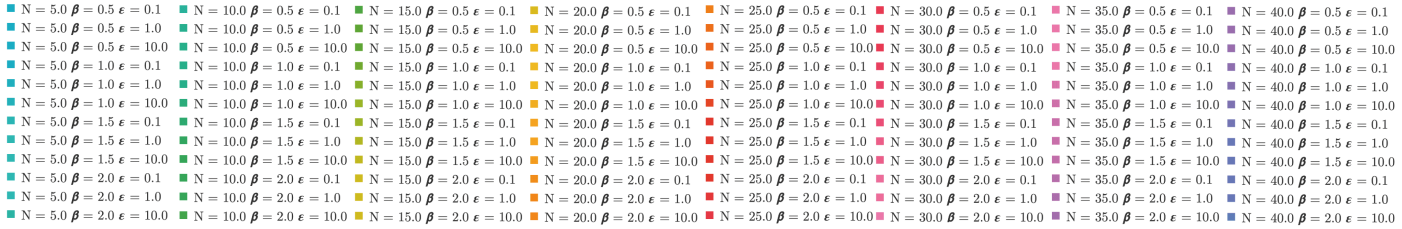


Figure 19: The legend for the Sobolev performance figures. Due to the size it is added once and is applicable for all the figures in the appendix

References

[1] C. Aggarwal. *Recommender Systems - The Textbook*. Springer, 2016.

[2] J. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou and T. Bouwmans. Reconstruction of time-varying graph signals via sobolev smoothness. *IEEE Transactions on Signal and Information Processing over Networks*, 8:201–214, 2022.

[3] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, jan 2004.

[4] F. Harper and J. Konstan. The movielens datasets: History and context. 5(4), dec 2015.

[5] A. Nikolakopoulos, X. Ning, C. Desrosiers and G. Karypis. Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems. survey, 2021.

[6] M. Koper. Item-item collaborative filtering via graph regularization. unpublished.

[7] P. Cremonesi M. Dacrema and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. RecSys '19, pages 101–109, New York, NY, USA, 2019. Association for Computing Machinery.

[8] A. Sandryhaila and J. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[9] H. Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, WWW '19, pages 3251–3257, New York, NY, USA, 2019. Association for Computing Machinery.

[10] D. Shuman, S. 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*, 30(3):83–98, 2013.

[11] E. Isufi, B. Das, A. Natali, M. Sabbaqi and M. Yang. Graph filters for processing and learning from network data. Technical report, Faculty of Electrical Engineering,Mathematics and Computer Science, Delft University of Technology, 2021.