Time-Aware Neighbourhood-Based Collaborative Filtering

T.W. de Zwart
Supervisor: R. van de Geer

Vrije Universiteit Amsterdam
Research Paper Business Analytics
May 2018

Summary

Recommender systems aim to provide relevant item recommendations to the users of a service. Evidently, well-performing recommender systems can aid corporations in increasing their users' consumption and satisfaction. This research set out to increase the performance of neighbourhood-based recommender systems by including temporal information in addition to ratings. To judge the effectiveness, we executed an evaluation procedure on data drawn from MovieLens and Yelp. Regarding prediction performance, we found ratings to already convey essentially all of the explanatory value. Moreover, for the MovieLens dataset we showed that binary interactions might even suffice in the similarity aspect of the neighbourhood-based systems. Thus, businesses looking to improve the prediction performance of their recommender systems are most likely to benefit more from other approaches than we investigated. However, the temporal adaptations might yield increased performance in different aspects of recommender systems, such as novelty, diversity and personality. Further research is needed to draw decisive conclusions based on these measures.

Contents

1 Introduction **4**

2 Neighbourhood-Based Collaborative Filtering **6**
2.1 Time-Unaware 6
2.1.1 User-to-User 6
2.1.2 Item-to-Item 8
2.2 Time-Aware 9
2.2.1 Temporal Similarity 10
2.2.2 Temporal Recency 11
2.2.3 Hybrid 12

3 Data **13**
3.1 MovieLens 13
3.2 Yelp 16

4 Evaluation **19**
4.1 Parameter Validation 21
4.2 Testing 22

5 Results **23**
5.1 MovieLens 23
5.1.1 Parameter Validation 23
5.1.2 Testing 26
5.2 Yelp 28
5.2.1 Parameter Validation 28
5.2.2 Testing 31

6 Conclusion and Discussion **34**

Appendices **37**

A Parameter Validation Results **37**
A.1 MovieLens 37
A.2 Yelp 42

1 Introduction

Large corporations such as Amazon deploy recommender systems to provide their users with personalized item recommendations [Linden et al., 2003]. Examples of items include E-commerce products (in the case of Amazon), films and restaurants. Evidently, well-performing recommender systems can aid corporations in increasing their users' consumption and satisfaction [Ricci et al., 2015, p. 5].

The systems come in many forms. Bobadilla et al. [2013] distinguish four classes: collaborative filtering, demographic filtering, content-based filtering and hybrid filtering. Collaborative filtering includes a neighbourhood-based approach that relies on explicit rating information. This is the most common method in the collaborative filtering landscape [Bobadilla et al., 2013] and possibly in the overall recommender systems landscape as well.

Neighbourhood-based collaborative filtering comprises two distinct variants: one that focuses on users' neighbourhoods and one that focuses on items' neighbourhoods. The former builds on the assumption that users with similar ratings tend to prefer similar items. The latter assumes that users have a preference for items that are similar to items they have rated highly in the past. Both systems operate in two phases. The first phase determines the neighbourhoods of the users or items and the second phase aggregates the ratings in these neighbourhoods.

Traditionally, the algorithms in these phases solely rely on explicit rating information. Often, timestamps accompany this rating data. If not, this temporal information is easy to acquire since it does not require additional user input. Intuitively, temporal information might convey explanatory value. This is predominantly due to user interests that change over time, but also due to evolution of certain types of items (for example restaurants).

Research has been conducted into incorporating temporal information in the neighbourhood determination phase [Liu et al., 2010, Organero et al., 2010, Wei et al., 2012, Hu et al., 2014] and rating aggregation phase [Ding and Li, 2005, Campos et al., 2010, Liu et al., 2010, Wei et al., 2012]. The presented methods include temporal data by considering the temporal similarity among historical ratings and the time distance from past ratings to the time of prediction.

This research investigates how temporal information can be included in neighbourhood-based collaborative filtering in order to improve the performance. We mainly focus on prediction performance, but also briefly touch upon the novelty, diversity and personality of recommendations. Regarding the temporal adaptations, the focus is on involving temporal similarity in the neighbourhood determination phase and the time distance from past ratings to the time of prediction in the aggregation phase. We refer to the former by the term *temporal similarity* and to the latter by *temporal recency*. To evaluate the performance, we apply the time-aware algorithms to datasets drawn from MovieLens and Yelp.

This paper begins by discussing neighbourhood-based collaborative filtering in Section 2. This includes the static variant as well as the adaptations for the time-aware variant. It will then go on to describe, process and visualize the data in Section 3. Section 4 outlines the evaluation and parameter tuning procedures. This also comprises a discussion of the performance aspects of recommender systems from a business perspective. Section 5 descriptively reports

the findings of the evaluation procedure. Additionally, it further investigates some remarkable results. Afterwards, Section 6 concludes and discusses this study.

2 Neighbourhood-Based Collaborative Filtering

Recommender systems in general and collaborative filtering algorithms in particular can be used to tackle two problems: rating prediction and top-n list recommendation [Charu, 2016, p. 30]. The rating prediction problem formulation is as follows: given a user and an item, predict the user's rating for the item. Throughout this paper, the terms *active user* and *active item* refer to the user-item pair for which the rating should be predicted. The top-n list recommendation problem definition is: given a user, predict the n most relevant recommendations. This problem formulation can be reduced to the rating prediction problem by recommending the n items with the highest predicted rating from the active user. In this report, we will therefore mostly focus on the rating prediction problem.

The temporal methods that we will consider are adaptations of the traditional, time-unaware neighbourhood-based methods. Therefore, we discuss these first in Section 2.1. We will then go on to present the time-aware variants in Section 2.2.

2.1 Time-Unaware

Time-unaware neighbourhood-based collaborative filtering comes in two forms: user-to-user and item-to-item. To predict a rating, the user variant combines the active item's ratings from users that are similar to the active user. The item-to-item collaborative filtering method generates a prediction by aggregating the active user's ratings of items similar to the active item.

Both variants infer predictions from explicit rating data. It is convenient to think of this data as a so called rating matrix. Let U be the set of users, let I be the set of items and let r_{ui} be the rating of user $u \in U$ for item $i \in I$. Then, the rating matrix R is a $|U|$ by $|I|$ matrix with elements r_{ui}. Typically, R is very sparse, since only a small subset of user-item pairs have a corresponding rating.

2.1.1 User-to-User

User-to-user collaborative filtering builds on the assumption that users with similar ratings tend to prefer similar items. This algorithm operates in two phases. The first phase determines the k users that are most similar to the active user, the so called neighbourhood. This requires a similarity measure which captures the similarity between the users' ratings. The second phase aggregates the active item's ratings from the users in the neighbourhood according to a prediction function.

Neighbourhood Determination To determine the neighbourhood of the active user u for active item i, we compute the similarity between u and all other users $v \in U \setminus \{a\}$. Many choices exist for the similarity function, but the two most popular are *cosine similarity* and *Pearson correlation* [Wei et al.,

2012]. The cosine similarity between user u and user v is:

$$\text{sim}(u,v) = \frac{\sum\limits_{i \in I_u \cap I_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_u \cap I_v} r_{ui}^2 \cdot \sum\limits_{i \in I_u \cap I_v} r_{vi}^2}}.$$

Here, I_u is the set of items rated by user u. Assuming that ratings are positive, the cosine similarity is a measure in the interval $[0, 1]$ where high values indicate a high similarity. When computing the similarity between users, it can occur that the denominator is zero. For example, this happens if two users have not rated any common items, i.e. if $I_u \cap I_v$ is empty. In such a situation, we are unable to compute a finite similarity and we simply set the similarity between the two users to zero. This applies to all user-to-user similarities we consider in this report.

Ricci et al. [2015] argue that the cosine similarity measure fails to capture differences in mean and variance between users' ratings and that the Pearson correlation does not have this drawback [p. 53]. The Pearson correlation between user u and user v is as follows:

$$\text{sim}(u,v) = \frac{\sum\limits_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_{u\cdot}) \cdot (r_{vi} - \bar{r}_{v\cdot})}{\sqrt{\sum\limits_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_{u\cdot})^2 \cdot \sum\limits_{i \in I_u \cap I_v} (r_{vi} - \bar{r}_{v\cdot})^2}}. \tag{1}$$

Here, $\bar{r}_{u\cdot}$ is the mean rating of user u, i.e. $\bar{r}_{u\cdot} = \frac{1}{|I_u|} \sum\limits_{i \in I_u} r_{ui}$. The Pearson correlation is a measure in the interval $[-1, 1]$. The sign indicates whether users u and v are similar (positive sign) or dissimilar (negative sign). The distance from 0 indicates the strength of the (dis)similarity. Note that there also exists a variant where the means $\bar{r}_{u\cdot}$ and $\bar{r}_{v\cdot}$ only consider items that are rated by both u and v, which is computationally more expensive [Charu, 2016, p. 35]. The variant displayed in Equation 1 is sometimes also referred to as the *adjusted cosine similarity*.

Let N_u^i be the set of k users that are most similar to user u and have rated item i. This set is commonly referred to as the neighbourhood corresponding to the user-item pair. Note that it is only sensible to include users with a positive similarity to the active user. Here, k is a parameter that indicates the size of the neighbourhood. Since the rating matrix is typically sparse, it can occur that the actual size of the neighbourhood is smaller than k.

Rating Aggregation Charu [2016] presents two prediction functions to aggregate the ratings once the neighbourhood has been determined. The *mean-centred* prediction function is:

$$\hat{r}_{ui} = \bar{r}_{u\cdot} + \frac{\sum\limits_{v \in N_u^i} \text{sim}(u,v) \cdot (r_{vi} - \bar{r}_{v\cdot})}{\sum\limits_{v \in N_u^i} \text{sim}(u,v)}.$$

Here, $\hat{r}_{ui}$ is the predicted rating for active item i from active user u. The *Z-score* prediction function is an adaptation of the mean-centred one. It includes the

users' rating standard deviation in addition to the mean:

$$\hat{r}_{ui} = \bar{r}_{u\cdot} + \sigma_u \cdot \frac{\sum\limits_{v \in N_u^i} \text{sim}(u,v) \cdot \frac{r_{vi} - \bar{r}_{v\cdot}}{\sigma_v}}{\sum\limits_{v \in N_u^i} \text{sim}(u,v)}.$$

Here, σ_u is the standard deviation of user u's ratings, i.e. $\sigma_u = \sqrt{\frac{\sum_{i \in I_u} (r_{ui} - \bar{r}_{u\cdot})^2}{|I_u| - 1}}$.

These prediction functions are not able to generate a rating prediction when the neighbourhood N_u^i is empty. If this is the case, we will simply predict the mean rating of the active item i. It can occur that the active item i has not received any ratings. Then, predicting item i's mean rating is impossible. In this situation, we predict the grand mean of all ratings.

2.1.2 Item-to-Item

Item-based collaborative filtering assumes that users have a preference for items that are similar to items they have rated highly in the past. It is closely related to the user-to-user variant: it comprises the same two stages and builds on comparable similarity and prediction functions. The main difference is that the neighbourhood is constructed in the item dimension instead of the user dimension.

Neighbourhood Determination Since the item-to-item variant determines the active item's neighbourhood, this phase requires a measure that captures the similarity between items. Again, the cosine similarity and Pearson correlation are popular metrics. The cosine similarity between item i and item j is as follows:

$$\text{sim}(i,j) = \frac{\sum\limits_{u \in U_i \cap U_j} r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_i \cap U_j} r_{ui}^2 \cdot \sum\limits_{u \in U_i \cap U_j} r_{uj}^2}}.$$

Here, U_i is the set of users that rated item i. Comparable to the user-to-user similarities, we set the similarity between two items to zero if we are unable to compute a finite similarity between the two. This holds for all item-to-item similarities we consider in this report.

The Pearson correlation between item i and item j is:

$$\text{sim}(i,j) = \frac{\sum\limits_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_{\cdot i}) \cdot (r_{uj} - \bar{r}_{\cdot j})}{\sqrt{\sum\limits_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_{\cdot i})^2 \cdot \sum\limits_{u \in U_i \cap U_j} (r_{uj} - \bar{r}_{\cdot j})^2}}.$$

Here, $\bar{r}_{\cdot i}$ is the mean rating of item i, i.e. $\bar{r}_{\cdot i} = \frac{1}{|U_i|} \sum\limits_{u \in U_i} r_{ui}$. The neighbourhood M_u^i consists of the k items most similar to the active item i which are rated by the active user u.

Rating Aggregation To aggregate the ratings in the neighbourhood M_u^i, we can again use the mean-centred or Z-score prediction formulas [Ricci et al., 2015,

p. 50-52]. For item-to-item collaborative filtering, the mean-centred prediction function is as follows:

$$\hat{r}_{ui} = \bar{r}_{\cdot i} + \frac{\sum\limits_{j \in M_u^i} \text{sim}(i,j) \cdot (r_{uj} - \bar{r}_{\cdot j})}{\sum\limits_{i \in M_u^i} \text{sim}(i,j)}.$$

The item-based formulation of the Z-score prediction is:

$$\hat{r}_{ui} = \bar{r}_{\cdot i} + \sigma_i \frac{\sum\limits_{j \in M_u^i} \text{sim}(i,j) \cdot \frac{r_{uj} - \bar{r}_{\cdot j}}{\sigma_j}}{\sum\limits_{i \in M_u^i} \text{sim}(i,j)}$$

Here, σ_i is the standard deviation of item i's ratings, i.e. $\sigma_i = \sqrt{\frac{\sum_{u \in U_i} (r_{ui} - \bar{r}_{\cdot i})^2}{|U_i| - 1}}$. In the case that the neighbourhood N_u^i is empty, we again will predict the mean rating of the active item i. If this is impossible, we predict the grand mean of all ratings.

2.2 Time-Aware

Table 1 shows an instance of a rating matrix. Suppose that we want to predict user A's rating for item W. According to time-unaware user-to-user collaborative filtering, the similarity between user A and B would be equal to the similarity between user A and C.

Table 1: *Hypothetical rating matrix*

	Item W	Item X	Item Y	Item Z
User A	?	4	5	?
User B	3	4	5	2
User C	2	4	5	3

Now assume that the corresponding rating dates are as in Table 2. Intuitively, user A is more similar to B than to C, because the common ratings are temporally closer. As a result, user A's predicted rating for item W should be more strongly influenced by user B's rating. Recall that in this report we refer to this notion of temporal closeness between historical ratings by the term temporal similarity.

Table 2: *Hypothetical time matrix*

	Item W	Item X	Item Y	Item Z
User A	?	17 Aug 2017	26 Oct 2017	?
User B	01 Aug 2016	17 Aug 2017	26 Oct 2017	16 Feb 2018
User C	22 Apr 2018	10 Feb 2016	22 Apr 2018	23 Mar 2017

Now suppose that B and C are equally similar to A (i.e. we use the time-unaware similarity). Time-unaware user-based collaborative filtering would ignore the time difference between B's rating of W and C's rating of W. However,

C's rating should instinctively have more influence on the predicted rating, since it is more recent. Remember that throughout this report the term temporal recency refers to this phenomenon.

For the temporal adaptations, it is convenient to also think of the time data as a matrix. Let t_{ui} be the timestamp of the rating of user $u \in U$ for item $i \in I$. Then, the timestamp matrix T is a $|U|$ by $|I|$ matrix with elements t_{ui}. The timestamp matrix T has the same sparsity structure as the rating matrix R. Typically, we record the timestamps as a date-time pair. However, the adaptations require the elements of T to be numeric. We can achieve this by, for example, defining t_{ui} as the number of seconds, hours or days since the timestamp of the first rating.

Although the focus of this section is on user-to-user collaborative filtering, the same notions and models analogously apply to the item-to-item variant. Section 2.2.1 deals with the incorporation of temporal similarity in the neighbourhood determination phase. In Section 2.2.2, we include temporal recency in the rating aggregation phase. Section 2.2.3 combines the two models into one hybrid, time-aware algorithm.

2.2.1 Temporal Similarity

Hu et al. [2014] include temporal similarity in the neighbourhood determination phase by incorporating weights in the similarity function. These weights ensure that temporally closer ratings have more influence on the similarity between users. Another approach is to simply compute the similarity of users based on the timestamp matrix T instead of the rating matrix R. This is proposed by Organero et al. [2010]. However, they do not present a thorough evaluation of its performance on publicly available datasets. In this report, we build further on the approach of Organero et al. [2010] by aggregating similarities that are obtained by applying the similarity function separately to the rating data and the timestamp data.

To compute the similarity between user u and v using the Pearson correlation function, we execute three steps.

1. Compute the *rating similarity* between u and v according to Equation 1. Let $\text{rsim}(u, v)$ denote this rating similarity.

2. Compute the *timestamp similarity* between u and v as:

$$\text{tsim}(u,v) = \frac{\sum\limits_{i \in I_u \cap I_v} (t_{ui} - \bar{t}_{u\cdot}) \cdot (t_{vi} - \bar{t}_{v\cdot})}{\sqrt{\sum\limits_{i \in I_u \cap I_v} (t_{ui} - \bar{t}_{u\cdot})^2 \cdot \sum\limits_{i \in I_u \cap I_v} (t_{vi} - \bar{t}_{v\cdot})^2}}.$$

   Here, $\bar{t}_{u\cdot}$ is the average timestamp of user u's ratings, i.e. $\bar{t}_{u\cdot} = \frac{1}{|I_u|} \sum\limits_{i \in I_u} t_{ui}$.

3. Combine the rating and timestamp similarity by means of a weighted mean to come to an aggregated similarity between user u and v:

$$\text{sim}(u,v) = \alpha \cdot \text{rsim}(u,v) + (1-\alpha) \cdot \text{tsim}(u,v).$$

   Here, $\alpha \in [0, 1]$ is a parameter that regulates the relative importance of the rating similarity. Higher values of α result in a greater emphasis on rating

similarity and consequently a smaller emphasis on timestamp similarity. Note that we obtain the traditional, time-unaware algorithm when $\alpha = 1$.

Alternatively, we can use the cosine similarity instead of the Pearson correlation to determine $\text{rsim}(u, v)$ and $\text{tsim}(u, v)$. Unlike the Pearson correlation, the outcome of the cosine similarity depends on the range of the values it is applied to. However, it is not necessarily a problem if the ranges of the ratings and timestamps are different: tuning the parameter α allows us to eliminate the effect of the range difference.

Apart from the different method of similarity computation, the algorithm is equal to the time-unaware variant described in Section 2.1.1. This temporal-similarity adaptation can be analogously included in item-to-item neighbourhood-based collaborative filtering.

2.2.2 Temporal Recency

Ding and Li [2005], Campos et al. [2010] and Liu et al. [2010] alter the prediction function of the time-unaware algorithms to incorporate temporal recency in the rating aggregation phase. They include weights $w_{vi}(t)$ such that recent ratings contribute more to the predicted rating than old ratings. Here, t denotes the time of prediction. For the user-based mean-centred prediction function, the result is as follows:

$$\hat{r}_{ui} = \bar{r}_{u\cdot} + \frac{\sum_{v \in N_u^i} \text{sim}(u, v) \cdot (r_{vi} - \bar{r}_{v\cdot}) \cdot w_{vi}(t)}{\sum_{v \in N_u^i} \text{sim}(u, v) \cdot w_{vi}(t)}.$$

The adapted Z-score prediction function is similar:

$$\hat{r}_{ui} = \bar{r}_{u\cdot} + \sigma_u \cdot \frac{\sum_{v \in N_u^i} \text{sim}(u, v) \cdot \frac{r_{vi} - \bar{r}_{v\cdot}}{\sigma_v} \cdot w_{vi}(t)}{\sum_{v \in N_u^i} \text{sim}(u, v) \cdot w_{vi}(t)}.$$

Strictly speaking, the predicted rating now depends on t and we should thus denote it by $\hat{r}_{uit}$. We deliberately do not adopt this notation, because this would unnecessarily clutter the evaluation section.

Several approaches exist for the computation of weights. Liu et al. [2010] propose an *exponential-decay* function:

$$w_{vi}(t) = e^{-\lambda \cdot (t - t_{vi})}.$$

Here, $\lambda \geq 0$ is a parameter that regulates the rate of decay. Higher values of λ result in a greater emphasis on recent ratings. Note that we obtain the traditional, time-unaware algorithm when $\lambda = 0$.

Another option is a window-based decay function [Campos et al., 2010]. This approach simply ignores ratings that are older than a predefined window-length. We will not consider this weight function in this research.

As was the case for the temporal-similarity adaptation of the previous Section, the rest of the algorithm is equal to the time-unaware variant described in Section 2.1.1. The outlined temporal-recency adaptations can be analogously included in item-to-item neighbourhood-based collaborative filtering.

2.2.3 Hybrid

The hybrid model combines the temporal adaptations from the two previous Sections. This model computes the similarity as described in Section 2.2.1. Like the time-unaware variant, it then determines the neighbourhood by selecting the k most similar users. Afterwards, the model applies the time-aware rating aggregation phase from Section 2.2.2.

3 Data

In this report, we apply the models from Section 2 to datasets drawn from MovieLens and Yelp. The former contains rating information about films obtained from the MovieLens recommender. This dataset comes in different sizes. We focus on the 1 million variant.[1] The Yelp dataset[2] originates from a challenge and contains information about the businesses, reviews and users from the Yelp service. In this report, we examine the data from the eleventh round of the challenge. We use the review portion of this data for the evaluation of the models. The expectation is that the performance gain of the time-aware models is larger for the Yelp dataset. This is because businesses such as restaurants can change over time, whereas films can not. We will explore the MovieLens and Yelp datasets in Sections 3.1 and 3.2 respectively.

3.1 MovieLens

The MovieLens 1 million dataset contains 1,000,209 reviews from 6,040 users about 3,706 films. The corresponding rating matrix thus has a density of 4.5%. Table 3 shows the first five rows of the dataset. The user attribute indicates the user id. These ids range from 1 to 6,040. Similarly, the item attribute gives the item id, ranging from 1 to 3,952. The fact that the highest item id is larger than the number of items suggests that there are films without any ratings. The third attribute, rating, specifies the value of the rating, as the name suggests. These ratings are integers on a scale from 1 to 5. The final attribute contains the timestamp in seconds since the Unix epoch (01 Jan 1970, 00:00:00 (UTC)). The smallest timestamp is 956,703,932 and the largest is 1,046,454,590 Converted to dates, the values of this attribute range from 26 Apr 2000 to 28 Feb 2003.

Table 3: *MovieLens dataset snapshot*

User	Item	Rating	Timestamp
1	1193	5	978300760
1	661	3	978302109
1	914	3	978301968
1	3408	4	978300275
1	2355	5	978824291

Figure 1 shows the cumulative number of ratings over time. In the year 2000, the cumulative number of ratings was sharply increasing. This growth decreased from 2001 and onwards. As a result, 90.5% of ratings have a corresponding timestamp that is on or before 31 Dec 2000. To obtain a more real-world insight of the performance of the models in an actively used recommender systems setting, we remove the remaining ratings from the data.

As a consequence, the dataset now contains 904,721 reviews from 6,034 users about 3,678 films. Furthermore, there are users that have no associated ratings. The same applies to films, but as mentioned before this was already the case with the complete dataset. For simplicity, we recode the user and item ids in

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://yelp.com/dataset/

such a way that they range from 1 to 6,034 and 1 to 3,678 respectively. Note that we lose the real-world interpretation of the ids during this process. In this report, this is not a problem since we do not link the ids to the corresponding users and films. After the removal and recoding processes, the density of the corresponding rating matrix is still 4.5%.

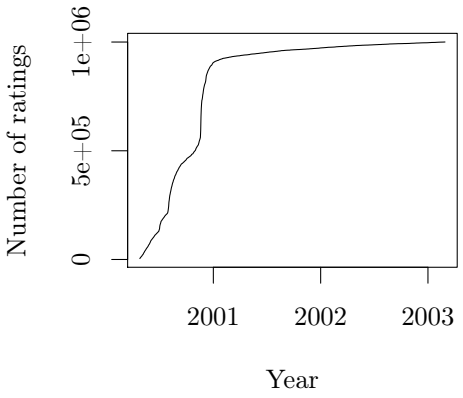

Figure 1: *Cumulative number of ratings over time (MovieLens)*

Figure 2 displays two histograms of the number of ratings per user, where the right graph is simply a zoomed in version of the left graph. The left graph shows a long tail: the majority of uses has given less than 500 ratings. Moreover, the right histogram shows a large spike at 20 ratings. We can explain this as follows. Initially, the complete dataset only included users with 20 ratings or more. As a result of removing the ratings after 31 Dec 2000, there is a small number of users with less than 20 ratings. The minimum number of ratings per user is 2 and the maximum is 2,029. On average, a user has given 149.9 ratings with a standard deviation of 174.1.

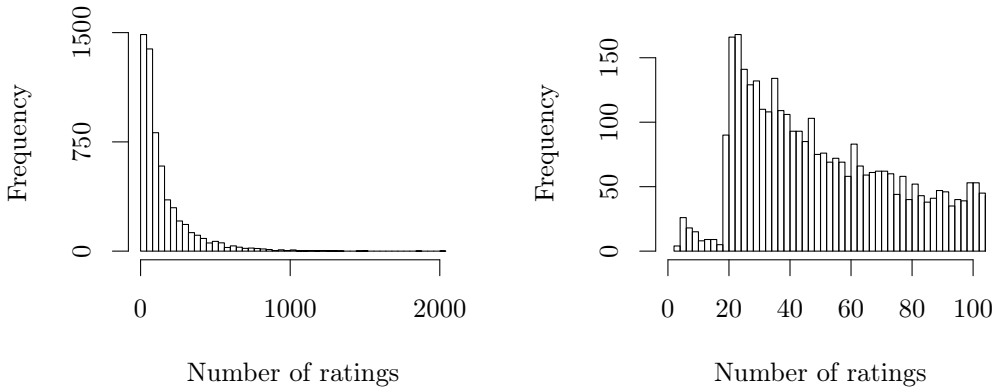

Figure 2: *Histograms of the number of ratings per user (MovieLens)*

Analogously, Figure 3 shows two histograms of the number of ratings per item. Compared to the user counterpart, the range is wider with a minimum of 1 and a maximum of 3,291. Furthermore, the distribution of the number of ratings per item also has a long tail. Unlike in the user-based figure, the right

graph does not show any apparent anomalies in the item-based figure. Because the dataset contains more users than items, it is not surprising that the number of ratings per item is generally larger than the number of ratings per user. On average, an item has 246.0 ratings with a standard deviation of 356.9.

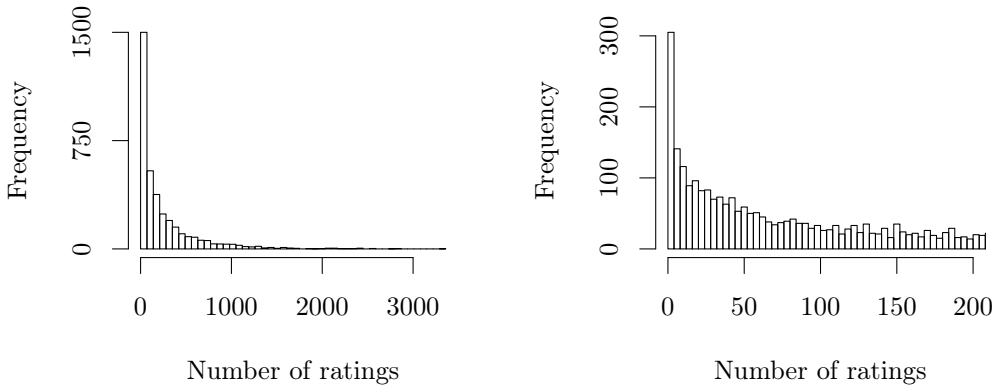

Figure 3: *Histograms of the number of ratings per item (MovieLens)*

Figure 4 depicts the mean daily rating over time, together with a smoothed trend. The mean daily rating seems to be slightly higher halfway November and in the beginning of July and October. In contrast, the opposite appears to be true for the start of the months August, November and December. However, these seasonal patterns are not terribly extreme. On all days combined, the overall mean rating is 3.59 with a standard deviation of 1.12.

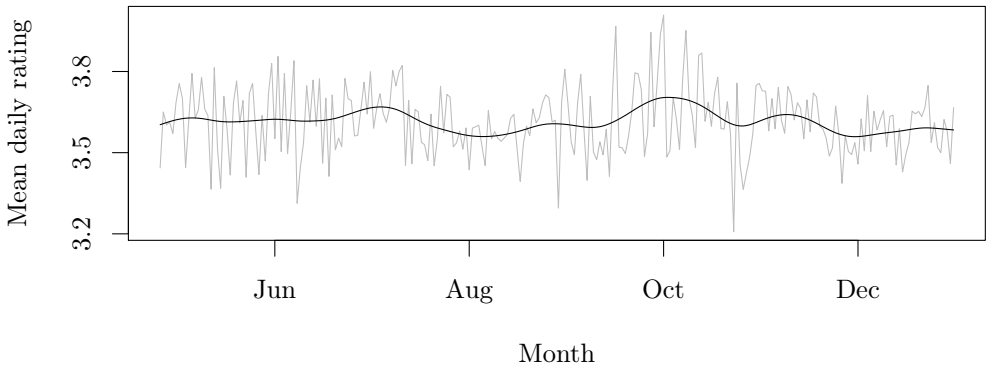

Figure 4: *Mean daily rating over time (MovieLens)*

In the implementation of the models, the large values of the timestamp attribute can cause integer overflows. Therefore, we convert the timestamps from seconds since the Unix epoch to days since 25 Apr 2000, 00:00:00 (GMT). As a result, the timestamp values now range from 0.96 to 250.96.

3.2 Yelp

The Yelp dataset contains 5,261,669 reviews from 1,326,101 users about 174,567 businesses. The corresponding rating matrix thus has a density of 0.0023%, much lower than the MovieLens equivalent. Table 4 shows a snapshot of the data that is most relevant for this research. The user and item attributes indicate the user and business ids. Both are strings of 22 characters. The third attribute, rating, specifies the value of the rating, as the name suggests. Similar to the MovieLens dataset, these ratings are integers on a scale from 1 to 5. The final attribute contains the date of the rating. The values of this attribute range from 22 Jul 2004 to 11 Dec 2017.

Table 4: *Yelp dataset snapshot*

User	Item	Rating	Date
le_brG6cwrzvWdKEGqA7YA	uz7UbvVUwsg68Rok6kbqRg	5	22 Jul 2004
w_6miJytUt6z8oRkGjVG-A	9X-43jnj6-6ZBuBdFm7BLA	2	15 Sep 2004
sE3ge33huDcNJGW3V4obww	PD2MAlYYi9HCqPH7IBKwTg	5	12 Oct 2004
c6HT44PKCaXqzN_BdgKPCw	u8C8pRvaHXg3PgDrsUHJHQ	5	19 Oct 2004
yYSBB5q7bY-qSVvmMgk4FA	GCRvrxMSC1nzShyM4Y-guQ	5	19 Oct 2004

In view of running time, we need to decrease the size of the dataset. Neighbourhood-based collaborative filtering relies on rating overlap between users and items. Therefore, we decrease the size in the following data selection procedure that aims at preserving overlap. We start by selecting the ratings of businesses of the category Restaurants that are located in the state Nevada. These are the largest category and state in terms of number of ratings, as can be seen in Figure 5. This selection contains 1,041,803 reviews from 358,722 users about 7,135 restaurants.

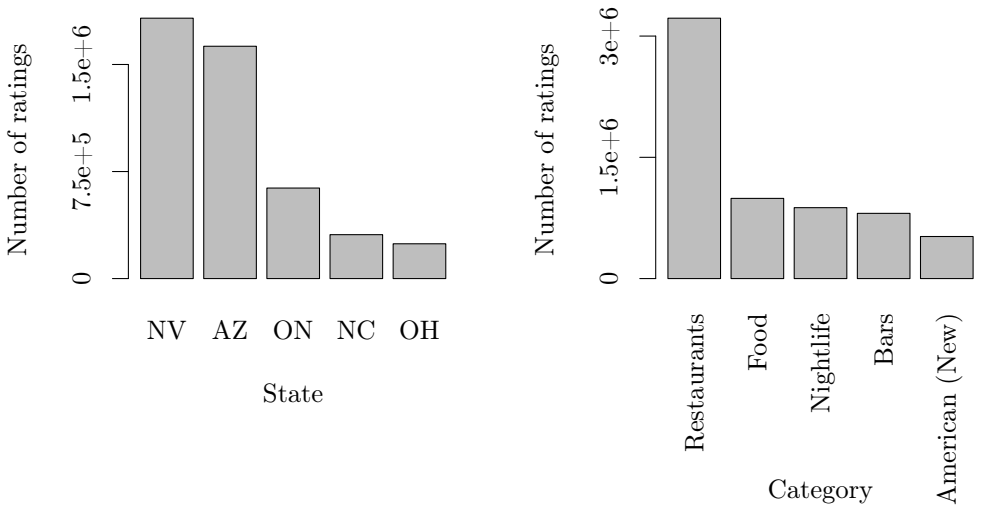

Figure 5: *Number of ratings per state and category (5 largest)*

To further decrease the number of users, we select the reviews of users with

at least 15 ratings. We filter out users with a low number of ratings, since they will most likely not benefit from neighbourhood-based collaborative filtering due to their limited amount of data. The resulting dataset contains 299,971 ratings from 8,605 users about 6,888 restaurants. Consequently, the corresponding rating matrix has a density of 0.51%. For convenience, we again recode the user and item ids to integers from 1 to 8,605 and 1 to 6,888 respectively.

Figure 6 shows the cumulative number of ratings over time. Unlike the MovieLens set, this figure does not show any counter-intuitive patterns for the Yelp data. In the beginning, the number of ratings grows slowly. As time progresses, it increases roughly exponentially. The oldest rating originates from 28 Apr 2005, whereas the most recent rating was given on 11 Dec 2017.

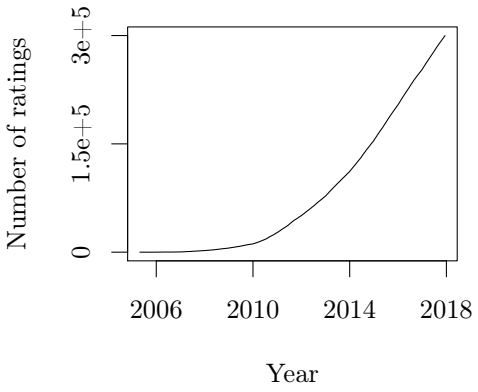

Figure 6: *Cumulative number of ratings over time (Yelp)*

Figure 7 depicts two histograms that contain information about the number of ratings per user. The right graph is a zoomed-in version of the left. Again, the graphs show a long tail: there is an extremely small amount of users with more than 80 ratings. Furthermore, the right graph indicates that there are no users with less than 15 ratings. This is a result from our above-described data selection procedure. Moreover, the maximum number of ratings per user is 1,291. On average, a user has 34.9 ratings with a standard deviation of 40.0.

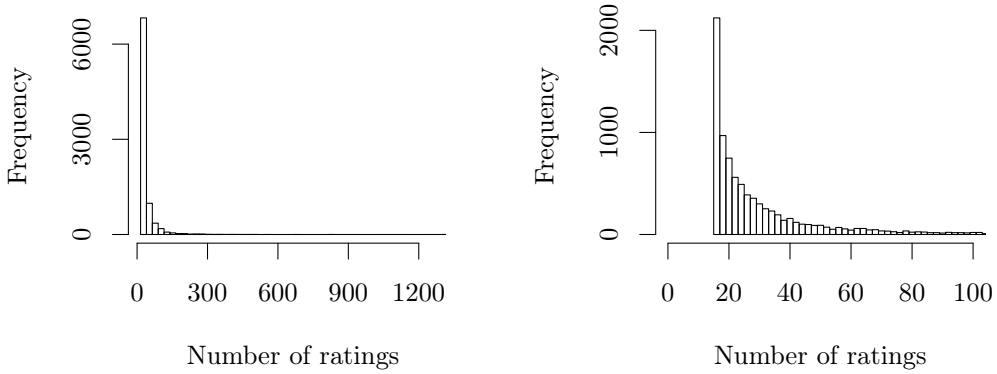

Figure 7: *Histograms of the number of ratings per user (Yelp)*

Likewise, Figure 8 displays two histograms of the number of ratings per item. The minimum number of ratings per item is 1 and the maximum is 1,005. Thus, the range is narrower compared to the user counterpart. Apart from the recurring long tail, these graphs show no noteworthy patterns. The average number of ratings per item is 43.5 with a standard deviation of 74.5.

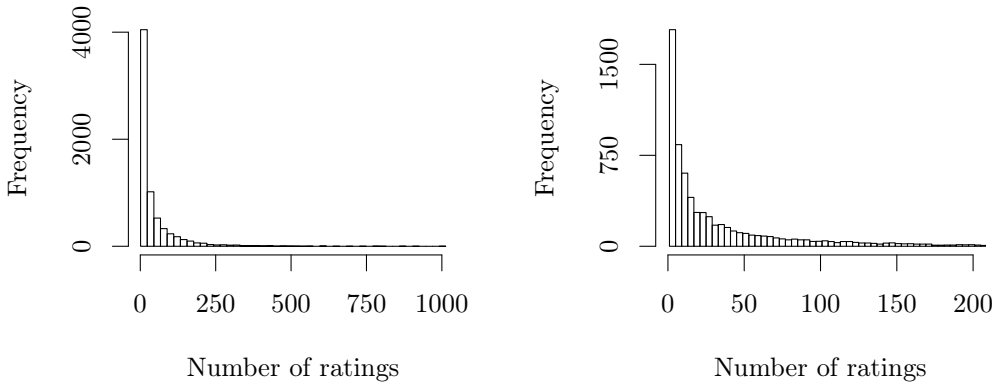

Figure 8: *Histograms of the number of ratings per item (Yelp)*

Figure 9 shows the mean daily rating over time, along with a smoothed trend. Up until 2009, the trend appears to be decreasing. Afterwards, it seems to be increasing, although very slowly. Furthermore, it stands out that the fluctuation in mean daily rating is much larger in the first years compared to the last. This is most likely a consequence from the low number of ratings per day in the beginning of the Yelp service: extremely high or low ratings have more influence on the mean daily rating when the number of ratings per day is limited. On all days combined, the overall mean rating is 3.72 with a standard deviation of 1.19.

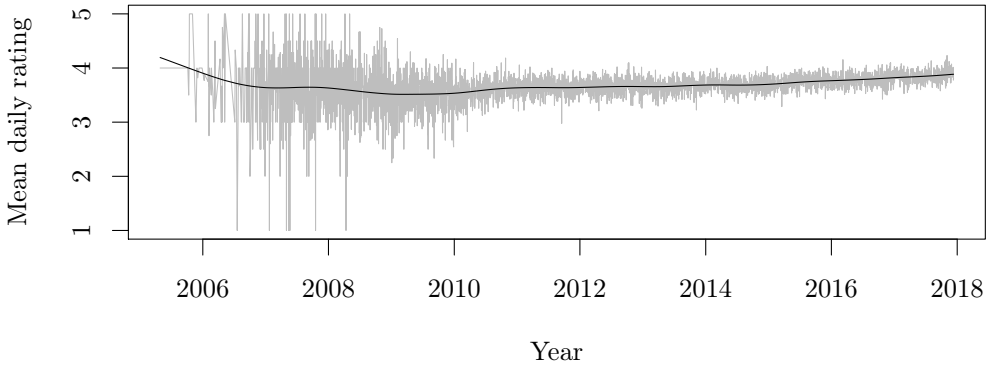

Figure 9: *Mean daily rating over time (Yelp)*

As mentioned in Section 2.2, the time-aware models require numerical timestamps. We obtain such timestamps by converting the dates to the number of days since 27 Apr 2005. The resulting timestamp values are integers and range from 1 to 4,611.

4 Evaluation

As is common in machine learning, we evaluate the performance of the algorithms based on a train-validate-test approach. Therefore, we split up the data into a training set, a validation set and a testing set, containing respectively 70%, 15% and 15% of the ratings and the corresponding timestamps. We train the models on the training set. In the context of neighbourhood-based collaborative filtering, this means that the neighbourhood determination infers from the data in this set. We use the validation set to determine the parameter values that yield the best performance. To come to an unbiased conclusion, we obtain the final performance values by fitting the models with optimal parameters on the training set and the validation set combined, before measuring the performance on the test set.

To avoid temporal dependency between the three sets, we split the data based on time. This is essential, since the models incorporate temporal information. Therefore, the test set contains the 15% most recent ratings. Likewise, the training set consists of the 70% oldest observations. The validation set contains the ratings that remain.

Evidently, accurate rating predictions indirectly result in relevant recommendations to users. Hence, prediction performance is an essential aspect of recommender systems from a business perspective. We therefore mainly focus on optimizing the prediction performance of the models. To measure this, we calculate the root mean square error (RMSE) and the mean absolute error (MAE) metrics. Suppose that we want to evaluate a model's prediction performance on the dataset $S \in \{\text{validation}, \text{test}\}$. Let $O_S \subset U \times I$ denote the set of observations that are included in S. The elements of O_S are user-item pairs (u, i). Then:

$$\text{RMSE}(S) = \sqrt{\frac{1}{|O_S|} \sum_{(u,i) \in O_S} (r_{ui} - \hat{r}_{ui})^2},$$

and

$$\text{MAE}(S) = \frac{1}{|O_S|} \sum_{(u,i) \in O_S} |r_{ui} - \hat{r}_{ui}| .$$

Both measures are error based, which means that lower values indicate better performance. The RMSE more strongly penalizes large errors compared to the MAE. Including both measures allows us to distinguish models that make a few large errors from models that consistently have a small prediction error.

The RMSE and MAE are metrics that measure the prediction performance on individual user-item pairs. Besides this, the quality of a recommender system is also characterized by various aspects of the top-n lists it is able to produce. Examples of these aspects include novelty and diversity [Bobadilla et al., 2013]. The novelty of recommendations refers to the degree of newness of the items in the recommendation list. Commonly, the newness of a recommended item is inversely related to the similarity with the items that the active user has already rated in the past. The diversity of recommendations is related to the extent to which the recommended items are dissimilar. Another characteristic is the personality of the top-n lists, which refers to the degree of diversity among the top-n recommendations to different users.

Although sufficient prediction performance is necessary for a good-quality recommender system, the top-n list characteristics are also important from a business perspective. These characteristics directly affect the user's experience, whereas the user does not explicitly see the predicted rating in most recommender system implementations. Evidently, it is desirable to have an algorithm with a sufficient level of novelty and diversity. This enables the users to discover unfamiliar and diverse types of items, which can assist in increasing their satisfaction. Furthermore, it is beneficial to have a recommender system with adequately personalized recommendations, instead of one that simply recommends the same top-n list to every user.

Thus, recommender systems should not solely focus on prediction performance, but also consider other characteristics such as novelty, diversity and personality. We will therefore report the models' performance in terms of these metrics. In view of running time, we will only measure these on the models we apply to the test set.

For the definitions of novelty and diversity we drew inspiration from Bobadilla et al. [2013]. The novelty of the top-n list recommended to user u is:

$$\text{novelty}(u) = \begin{cases} \frac{1}{|\tilde{I}_u| \cdot |Z_u|} \sum_{i \in \tilde{I}_u} \sum_{j \in Z_u} (1 - \text{isim}(i,j)) & \text{if } |\tilde{I}_u| > 0 \\ 1 & \text{else.} \end{cases}$$

Here, $\tilde{I}_u$ is the set of items for which a rating of user u is included in the training or validation set, Z_u is the set of items that are included in the top-n list recommended to user u and $\text{isim}(i,j)$ is the item-similarity between item i and j. In this report, we only consider the cosine similarity of the ratings in view of simplicity. Naturally, other options are possible in terms of the similarity measure (e.g. Pearson correlation) and the values to which we apply the measure (e.g. the timestamp values).

We compute the diversity of the top-n list recommended to user u as follows (assuming that $n \geq 2$):

$$\text{diversity}(u) = \frac{1}{|Z_u| \cdot (|Z_u| - 1)} \sum_{i \in Z_u} \sum_{j \in Z_u : j \neq i} (1 - \text{isim}(i,j)) \,.$$

The outcomes of the novelty and diversity metrics are in the range $[0, 1]$ where larger values indicate higher levels of novelty and diversity.

Ideally, we would combine the novelty and diversity over all users to come to an aggregated measure. In view of running time, this is not feasible. Therefore, we randomly sample a set of 50 users. Note that we use the same random sample of users when evaluating different models on the same dataset. Let $\tilde{U}$ denote the set of sampled users. Then, we aggregate the novelty and diversity of the users in $\tilde{U}$ by applying an unweighted mean.

Furthermore, we formulate the personality measure as follows:

$$\text{personality} = \frac{\left| \cup_{u \in \tilde{U}} Z_u \right|}{\sum_{u \in \tilde{U}} |Z_u|}.$$

Again, the outcome lies in the range $[0, 1]$ and higher values indicate a higher level of personality.

Throughout the rest of this report, we fix $n = 10$. Following the brief description at the beginning of Section 2, we obtain the top-10 list for user u

by selecting the 10 items that have the highest predicted rating and are not rated by the user (i.e. not an element of $\tilde{I}_u$). When using the temporal-recency models to obtain top-10 lists, we need to specify t, the time of prediction. For the MovieLens dataset we use $t = 250.96$ and for the Yelp we use $t = 4,611$. These are the largest timestamp values that are present in these datasets.

4.1 Parameter Validation

In our evaluation procedure, we determine the optimal parameters of the models based on prediction performance in terms of RMSE and MAE. To start, we determine the optimal parameters for the time-unaware models. We consider these models with all combinations of the following parameters: variant $v \in \{\text{user-to-user}, \text{item-to-item}\}$, similarity measure $s \in \{\text{cosine}, \text{Pearson}\}$, neighbourhood size $k \in \{2, 5, 10, 20, 30, 40, 50, 60, 75, 100, 150, 250\}$ and prediction function $p \in \{\text{mean-centred}, \text{Z-score}\}$.

In view of running time, we can not investigate all possible parameter combinations for the time-aware models. Therefore, we fix $k = k^*_{v,s}$ and $p = p^*_{v,s}$ based on the validation results of the time-unaware models. We choose the parameter values $k^*_{v,s}$ and $p^*_{v,s}$ that yield a reasonable balance between prediction performance and running time. These fixed values can differ based on the values of v and s.

With k and p fixed, we consider the temporal-similarity models with all combinations of the following parameters: $v \in \{\text{user-to-user}, \text{item-to-item}\}$, $s \in \{\text{cosine}, \text{Pearson}\}$ and $\alpha \in \{0, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.95, 0.975, 1\}$.

The parameter tuning procedure described above is equal for both datasets. This is not the case for the λ parameter we consider in the temporal-recency models, since the two sets span time periods of greatly different lengths. For the temporal-recency models we apply to both datasets, we vary the variant and similarity measure: $v \in \{\text{user-to-user}, \text{item-to-item}\}$ and $s \in \{\text{cosine}, \text{Pearson}\}$. Additionally, we vary the parameter λ. For the MovieLens data, we consider $\lambda \in \{0.001, 0.0025, 0.005, 0.01, 0.02\}$. For the Yelp dataset, we consider $\lambda \in \{0.0001, 0.00025, 0.0005, 0.001, 0.002\}$. We deem these parameter values to be reasonable based on Figures 10 and 11. Given a parameter combination, we again only consider the fixed values of k and p.

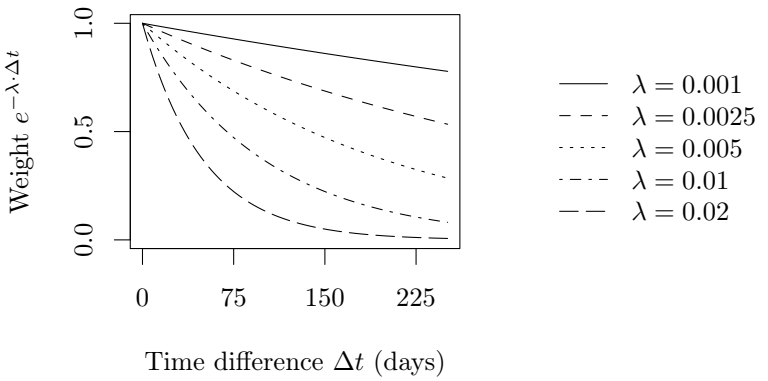

Figure 10: *Temporal recency weight functions MovieLens*

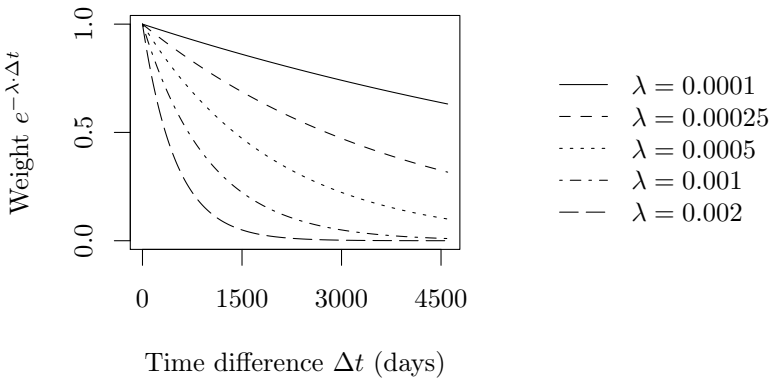

Figure 11: *Temporal recency weight functions Yelp*

For the user-to-user and item-to-item hybrid models, we simply combine the temporal-similarity and recency models' parameters that yield the best prediction performance on the validation set.

4.2 Testing

Based on the validation results, we evaluate the RMSE, MAE, novelty, diversity and personality of the following three types of models on the test set.

1. The best-performing (in terms of RMSE and MAE) temporal-similarity, temporal-recency and hybrid models for both variants $v \in \{$user-to-user, item-to-item$\}$. These are only included if they outperform their time-unaware equivalents in terms of prediction performance.

2. The time-unaware equivalents of the models in 1. We include these to be able to determine the performance gains of the time-aware adaptations.

3. The best-performing (in terms of RMSE and MAE) time-unaware models for both variants $v \in \{$user-to-user, item-to-item$\}$. Including these models allows us to argue whether it is sensible to use the time-aware models at all.

5 Results

In this section, we descriptively report the results of the evaluation procedure described in Section 4. In addition, we further investigate some of the remarkable results. This comprises analyses of the prediction performance convergence for large k and the explanatory value of binary interactions.

Sections 5.1 and 5.2 deal with the MovieLens and Yelp datasets respectively. For reference, we provide all validation results in Appendix A. We present all testing results in the respective sections below.

5.1 MovieLens

We start with a general remark: the MAE of the models applied to the MovieLens dataset shows similar patterns to the RMSE. In view of clarity, we omit the figures of the MAE results in this section and display them in Appendix A.1.

5.1.1 Parameter Validation

Time-unaware models Figure 12 depicts the prediction performance of the user-to-user variant of the time-unaware models in terms of the RMSE. In view of clarity, the x-axis is truncated at $k = 75$. The figure shows that lower values of k yield worse performance when k is relatively small. For $k \geq 50$, however, the RMSE remains approximately stable. Moreover, the mean-centred prediction function seems to consistently outperform the Z-score equivalent. Furthermore, the Pearson correlation yields better performance compared to the cosine similarity measure.

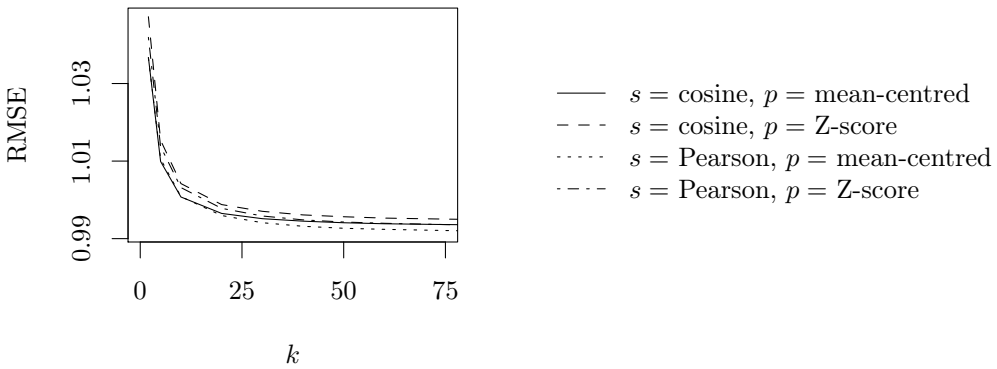

Figure 12: *RMSE of the time-unaware user-to-user models applied to the MovieLens validation set*

Likewise, we display the RMSE of the item-to-item time-unaware models in Figure 13. Again, the x-axis is truncated at $k = 75$. Apart from the fact that the item-to-item variant generally yields lower RMSE values, the relative patterns are similar to the user-to-user variant: lower values of k result in worse performance and the RMSE is approximately stable for $k \geq 50$. In addition, the mean-centred prediction function again yields lower RMSE values compared to

the Z-score function, although the difference is extremely small. In contrast to the user-to-user variant, the cosine similarity performs better than the Pearson correlation for the item-to-item variant.

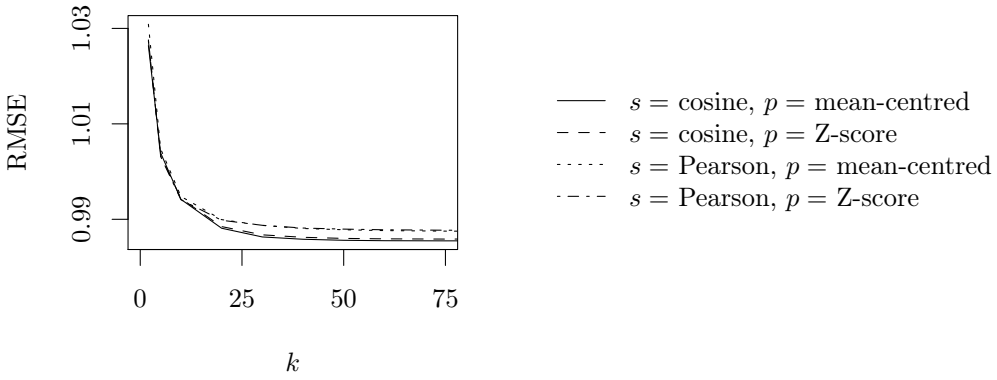

Figure 13: *RMSE of the time-unaware item-to-item models applied to the MovieLens validation set*

Since the models with lower values of k have a shorter running time and the prediction performance does not seem to improve for values larger than 50, we fix $k^*_{v,s} = 50$, for $v \in \{\text{user-to-user}, \text{item-to-item}\}$ and $s \in \{\text{cosine}, \text{Pearson}\}$. Additionally, we fix $p^*_{v,s} = \text{mean-centred}$ for $v \in \{\text{user-to-user}, \text{item-to-item}\}$ and $s \in \{\text{cosine}, \text{Pearson}\}$ because of its smaller RMSE (and MAE) compared to the Z-score prediction function.

Convergence In Figures 12 and 13, we found remarkable results for high values of k. First of all, the user-to-user variant of the time-unaware models seems to converge to a stable RMSE when the value of k is sufficiently large. As the neighbourhood size k grows, the models might eventually predict values close to the item means over all users. The algorithm that simply predicts the mean rating of the active item is sometimes referred to as the *item-mean popular* algorithm. On the MovieLens validation set, this algorithm yields a RMSE of 0.9950. This is approximately equal to the RMSE value that the time-unaware models converge to. Thus, we conclude that the prediction performance of user-to-user neighbourhood-based collaborative filtering indeed converges to the performance of the item-mean popular algorithm as k increases. However, we expect that the personality of the item-mean popular algorithm is considerably lower. We will further investigate this in Section 5.1.2.

Similar to the user-to-user variant, the RMSE of the item-to-item models also converge when k is sufficiently large. We therefore investigate whether it is plausible that the item-to-item models converge to the *user-mean popular* algorithm, which simply predicts the mean rating of the active user. When applied to the MovieLens validation set, the user-mean popular algorithm results in a RMSE of 1.1458, which is considerably higher than the value of the RMSE to which the item-to-item models converge (approximately 0.985). Thus, we conclude that the item-to-item time-unaware models do not converge to the user-mean popular algorithm in terms of prediction performance.

Temporal-similarity models Figure 14 shows the prediction performance of the temporal-similarity models. Recall that the temporal-similarity models with $\alpha = 1$ are simply equal to their time-unaware equivalent. What stands out is that including temporal similarity does not seem to strongly affect the performance of the models that use the cosine similarity measure. In contrast, the RMSE of the models that use the Pearson correlation does to some extent depend on the value of α. For the user-to-user variant, the RMSE decreases monotonically as α increases and thus the time-unaware model performs best. For the item-to-item variant, however, we obtain the lowest RMSE with $\alpha = 0.5$.

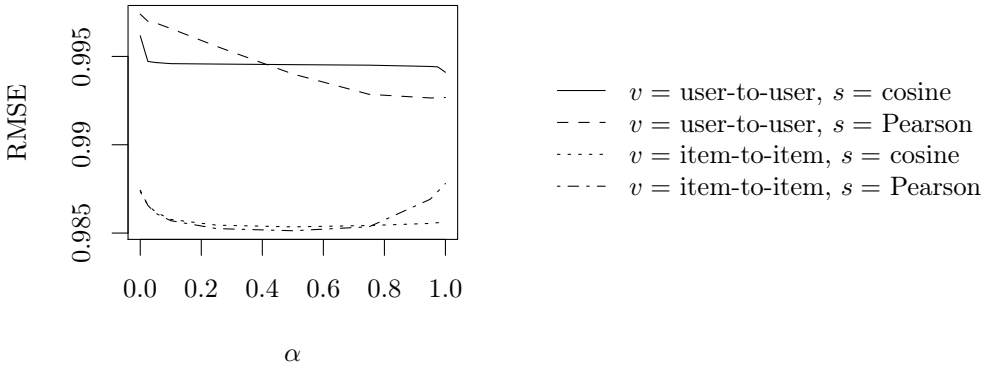

Figure 14: *RMSE of the temporal-similarity models applied to the MovieLens validation set*

Temporal-recency models Figure 15 shows the RMSE of the temporal-recency models. Remember that we obtain the time-unaware models with $\lambda = 0$. Interestingly, the value of λ does not seem to affect the errors of the item-to-item models. For the user-to-user variant, the RMSE increases monotonically as λ grows. All in all, we conclude that the including temporal recency is not sensible from a prediction performance viewpoint.

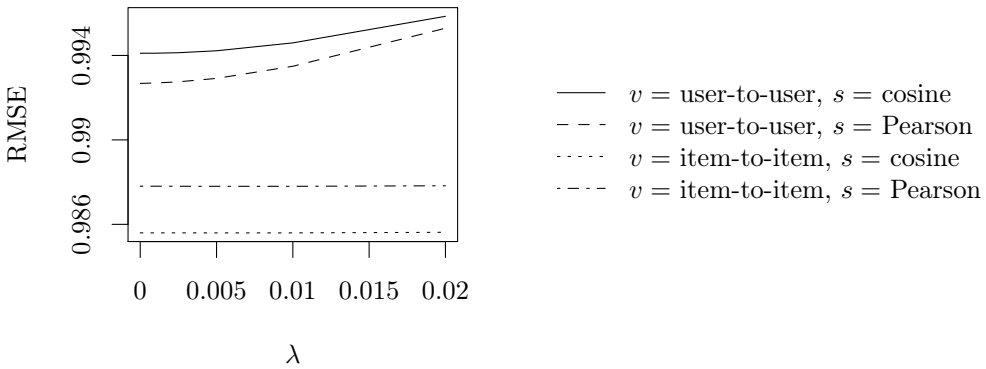

Figure 15: *RMSE of the temporal-recency models applied to the MovieLens validation set*

5.1.2 Testing

Table 5 lists the collaborative filtering models we apply to the MovieLens test set. For convenience, we name these models with letters from A to D. Model A is the best performing item-to-item temporal-similarity model. The parameter evaluation procedure showed that the temporal-similarity adaptation is not sensible for the user-to-user variant, in terms of prediction performance. The same holds for the temporal-recency and hybrid models for both variants $v \in \{\text{user-to-user, item-to-item}\}$. Therefore, we do not evaluate the performance of these models on the test set. Model B is the time-unaware equivalent of model A. Furthermore, we include models C and D, which are the best-performing time-unaware models. In addition to the models listed in the table, we briefly test the item-mean popular algorithm.

Table 5: *MovieLens test set collaborative filtering models*

Name	v	s	k $(= k^*_{v,s})$	p $(= p^*_{v,s})$	α	λ
A	item-to-item	Pearson	50	mean-centred	0.5	0
B	item-to-item	Pearson	50	mean-centred	1	0
C	user-to-user	Pearson	50	mean-centred	1	0
D	item-to-item	cosine	50	mean-centred	1	0

Prediction performance Figure 16 depicts histograms of the test-set evaluation errors of the collaborative filtering models. The errors seem to be approximately identically distributed across the four models. The only (minor) difference is that the most negative error is slightly lower for models B and C when compared to A and B, but this difference is not noticeable in the histograms.

The fact that is hardly a prediction performance difference between the models is also reflected in the RMSE and MAE that we show in Table 6. Including temporal similarity in the item-to-item model with s = Pearson marginally decreases the RMSE with 0.072% and the MAE with 0.065%. Compared to the best-performing time-unaware model (model D), however, the temporal-similarity model performs worse. Furthermore, the user-to-user collaborative filtering model outperforms the item-mean popular algorithm, whereas they performed approximately equal on the validation set. Apparently, the former benefits more from the larger amount of training data than the latter.

Top-n characteristics In terms of novelty, diversity and personality, the collaborative filtering models do not differ extremely. There are, however, large differences between these models and the item-mean popular algorithm. To start, the novelty and diversity of the popular algorithm are considerably higher. This can be explained as follows. It is intuitively logical that the items with a mean rating of 5 stars have received a low number of ratings. These are exactly the items that the item-mean popular algorithm recommends in its top-10 lists. In general, items with a low number of ratings are extremely dissimilar to all the other items. This is because of the small amount of overlapping ratings (in many cases there is even no overlap). Hence, the items in the top-10 lists are

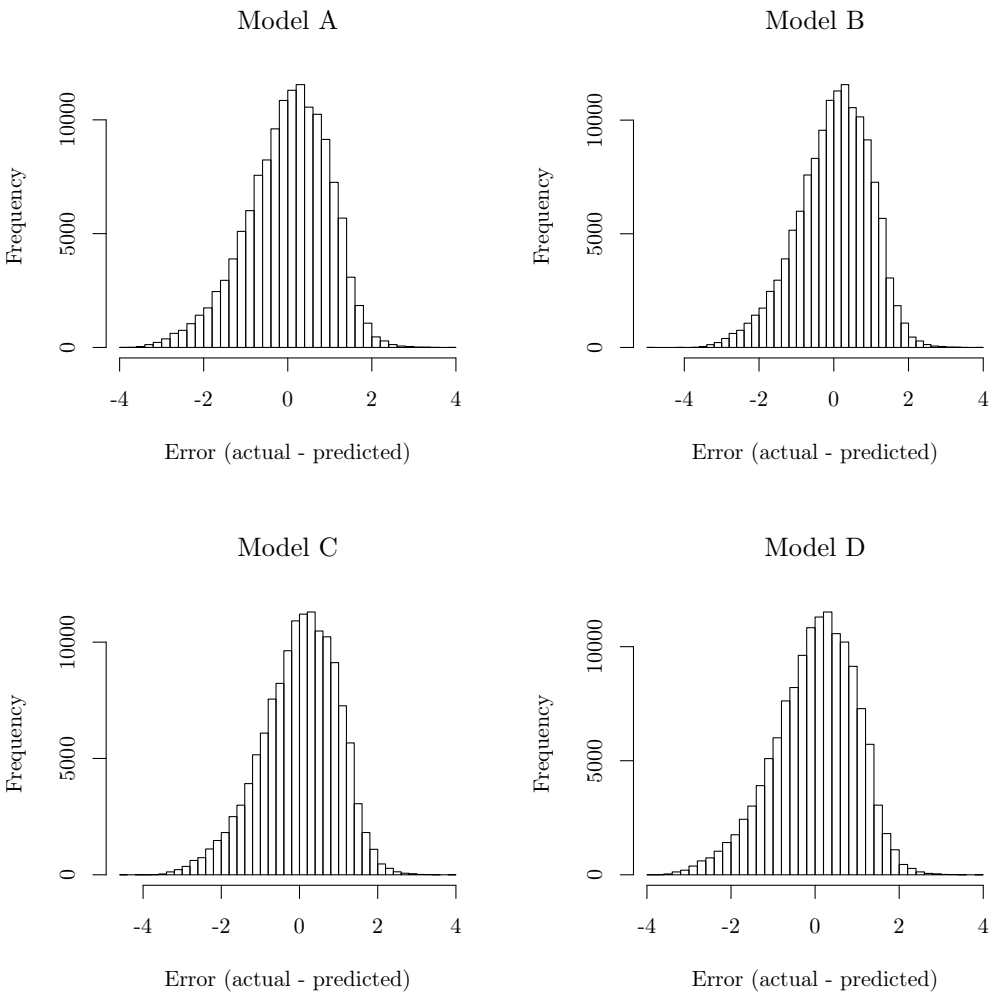

Figure 16: *Histograms of the errors of the collaborative filtering models applied to the MovieLens test set*

predominantly dissimilar to each other and to all other items. Therefore, the novelty and diversity of the item-mean popular algorithm are relatively high.

Compared to the collaborative filtering models, the item-mean popular algorithms is far less personalized. This was expected, since the mean ratings of items are equal for all users. Note that it is still possible that the popular algorithm's top-10 lists differ between users, because these exclude items that are already rated by the active user. However, such an exclusion does not occur often since the items with a high mean rating generally have a low number of ratings. In fact, all users receive exactly the same top-10 lists in this case.

Binary interactions Throughout the parameter validation and testing procedures, the fact that the temporal adaptations do not strongly affect the prediction performance of the models has been a remarkable, recurring outcome.

Table 6: *RMSE, MAE, novelty, diversity and personality of the models applied to the MovieLens test set*

Name	RMSE	MAE	Novelty	Diversity	Personality
A	0.9718	0.7730	0.1041	0.1703	0.830
B	0.9725	0.7735	0.1074	0.1846	0.806
C	0.9759	0.7768	0.1107	0.1879	0.826
D	0.9712	0.7730	0.1053	0.1857	0.790
Item-mean popular	0.9862	0.7869	0.4085	0.5000	0.020

Especially the lacking effect of the temporal-similarity adaptation lead us to believe that a binary rating matrix (where $r_{ui} = 1$ if user u rated item i and 0 otherwise) would suffice for this dataset. This type of data is sometimes also referred to as the *binary interactions* between users and items. This belief would imply that the number of common users or items would be the most valuable information for the similarity computations, as opposed to the actual values of the ratings or timestamps.

Preliminary analysis shows that this belief is plausible, since model D results in a RMSE of 0.9790 when applied to the binary rating matrix. This is just 0.803% higher than when we apply the model to the non-binary matrix. Further parameter tuning based on the binary variant, possibly with different similarity functions that are especially attuned to binary data, could likely decrease this RMSE even further. Note that we solely used the binary rating matrix for the similarity computations and not for the rating aggregations.

5.2 Yelp

5.2.1 Parameter Validation

Time-unaware models Figures 17 and 18 display the performance of the user-to-user time-unaware models in terms of the RMSE and MAE respectively. For clarity, the x-axes are truncated at $k = 75$. It stands out that the models perform worse when applied to the Yelp dataset instead of MovieLens. This is most likely a consequence of the lower density of the Yelp rating matrix. Furthermore, the figures show that the relation between the RMSE (and MAE) and k is similar for the Yelp and MovieLens datasets. As we will later see, this is also the case for the item-to-item variant. Therefore, we again fix $k^*_{v,s} = 50$, for $v \in \{\text{user-to-user}, \text{item-to-item}\}$ and $s \in \{\text{cosine}, \text{Pearson}\}$.

Overall, the user-to-user model with $s = \text{cosine}$ and $p = \text{Z-score}$ yields the lowest RMSE and MAE of all user-based models. According to the RMSE, the cosine similarity measure performs best when combined with the Z-score prediction function, whereas it is optimal to combine the Pearson correlation measure with the mean-centred prediction function. The figure containing the MAE of the models shows a slightly different pattern. According to the MAE, the mean-centred and Z-score prediction functions perform nearly identical when combined with the Pearson correlation measure. Therefore, we base our fixed values of p solely on the RMSE: $p^*_{\text{user-to-user,cosine}} = \text{Z-score}$ and $p^*_{\text{user-to-user,Pearson}} = \text{mean-centred}$.

We depict the RMSE and MAE of the item-to-item time-unaware models in

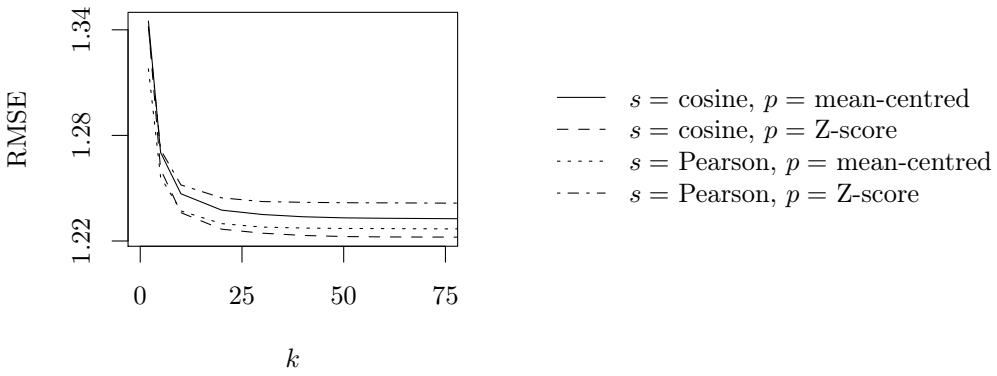

Figure 17: *RMSE of the time-unaware user-to-user models applied to the Yelp validation set*

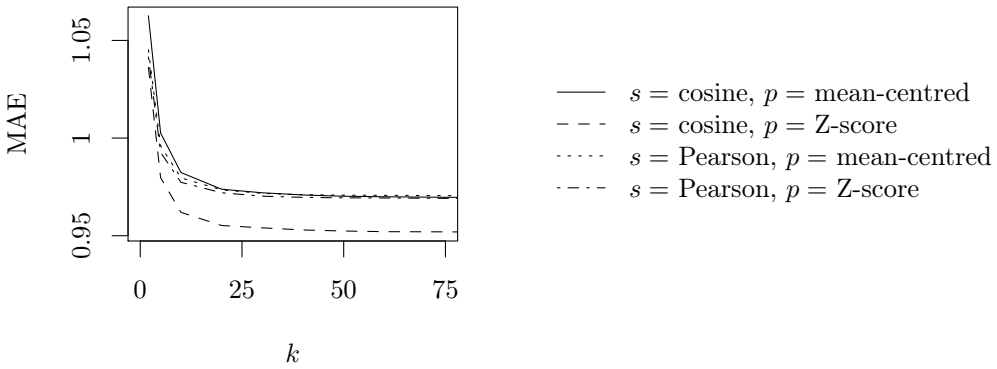

Figure 18: *MAE of the time-unaware user-to-user models applied to the Yelp validation set*

the Figures 19 and 20. Again, we truncate the x-axes at $k = 75$. Similar to the user-to-user variant, the item-to-item models perform worse when we apply them to the Yelp data instead of the MovieLens set. Both figures show that the cosine similarity measure is superior to the Pearson correlation for the item-to-item models. Furthermore, the RMSE figure indicates that there is a negligible performance difference between the two prediction functions. In contrast, the Z-score function yields lower MAE values for both similarity measures. Hence, we fix $p^*_{\text{item-to-item,cosine}} = p^*_{\text{item-to-item,Pearson}} = \text{Z-score}$ for further evaluation and testing.

Convergence Similar to the MovieLens results, the time-unaware models applied to the Yelp data showed convergence in terms of RMSE and MAE for large values of k. Therefore, we again investigate the item-mean and user-mean algorithms. When we apply the item-mean algorithm to the Yelp validation set, we obtain a RMSE of 1.2046. We can therefore conclude that this algorithm slightly outperforms the best-performing user-to-user model, which converges to a RMSE of approximately 1.22. Most likely, the performance of the collab-

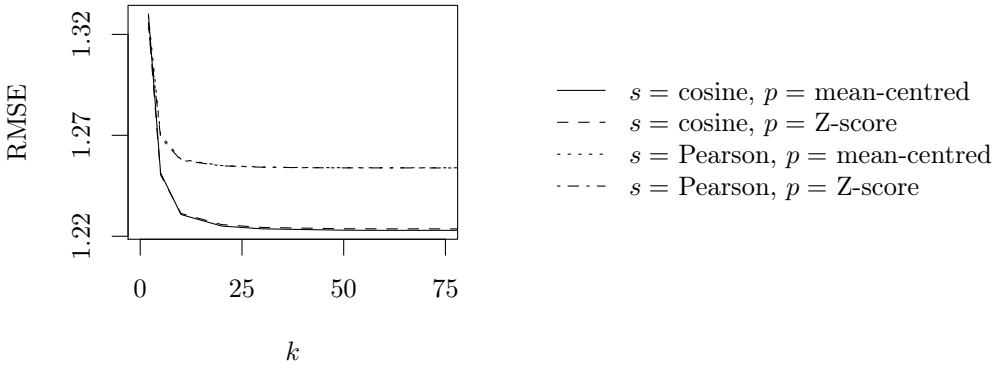

Figure 19: *RMSE of the time-unaware item-to-item models applied to the Yelp validation set*

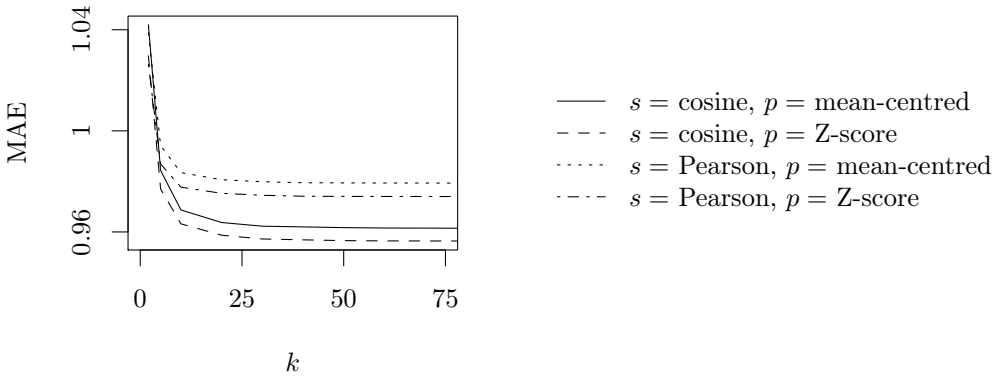

Figure 20: *MAE of the time-unaware item-to-item models applied to the Yelp validation set*

orative filtering models is limited due to the low density of the rating matrix. However, we again expect the personality of the item-mean popular algorithm to be considerably lower. We will further investigate this for the Yelp dataset in Section 5.2.2.

Comparable to what we found for the MovieLens models, the item-to-item time-unaware models do not converge to the user-mean popular algorithm in terms of prediction performance. The popular algorithm yields a RMSE of 1.2597. This is considerably higher than the RMSE value to which the best-performing item-to-item collaborative filtering model converges (approximately 1.22).

Temporal-similarity models As was the case for the time-unaware models, the RMSE and MAE display different patterns for the time-aware models. However, these differences seem to be propagations of the differences that are already present in the validation results of the time-unaware models we presented above. Therefore, we omit the figures of the time-aware MAE validation results in this section and display them in Appendix A.2.

Figure 21 depicts the performance of the temporal-similarity models in terms of the RMSE. Similar to what we have seen with the MovieLens dataset, the RMSE of the models that use the cosine similarity does not seem to depend on the value of α. This is different for the models with s = Pearson. Although the RMSE seems to fluctuate somewhat randomly, the figure suggests that the models that purely use the timestamp or rating similarity yield better results than a mix of the two. There is no value of α that clearly shows better performance than the time-unaware models. All in all, we thus conclude that including temporal similarity is not sensible from a prediction performance point-of-view.

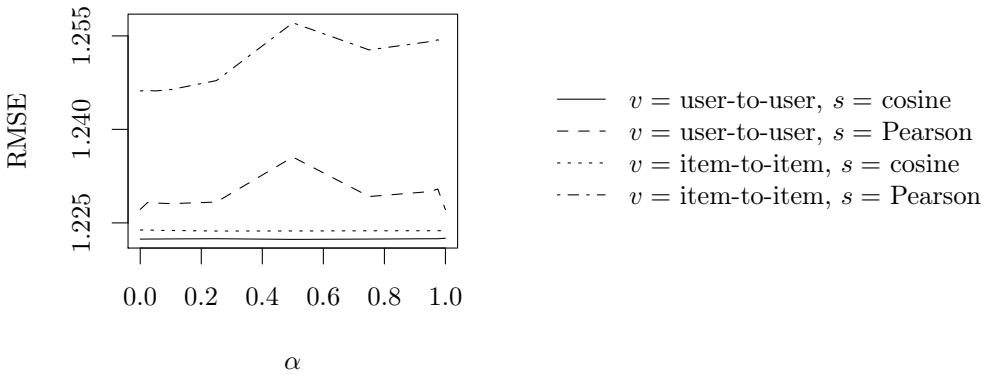

Figure 21: *RMSE of the temporal-similarity models applied to the Yelp validation set*

Temporal-recency models Figure 22 depicts the RMSE of the temporal-recency models. All parameter combinations approximately have the same general pattern: the performance degrades monotonically as λ increases. The RMSE of the user-to-user models seems to increase faster than the RMSE of the item-to-item variants. As was the case for the MovieLens set, including temporal recency is thus not rational for the models we apply to the Yelp data, in terms of prediction performance.

5.2.2 Testing

Table 7 lists the collaborative filtering models we apply to the Yelp test set, for convenience named E and F. Since the parameter validation procedure showed that it is not rational to apply the time-aware models based on the RMSE and MAE, we solely test the best-performing time-unaware models. In addition to the models listed in the table, we briefly test the item-mean popular algorithm.

Table 7: *Yelp test set collaborative filtering models*

Name	v	s	k $(= k^*_{v,s})$	p $(= p^*_{v,s})$	α	λ
E	user-to-user	cosine	50	Z-score	1	0
F	item-to-item	cosine	50	Z-score	1	0

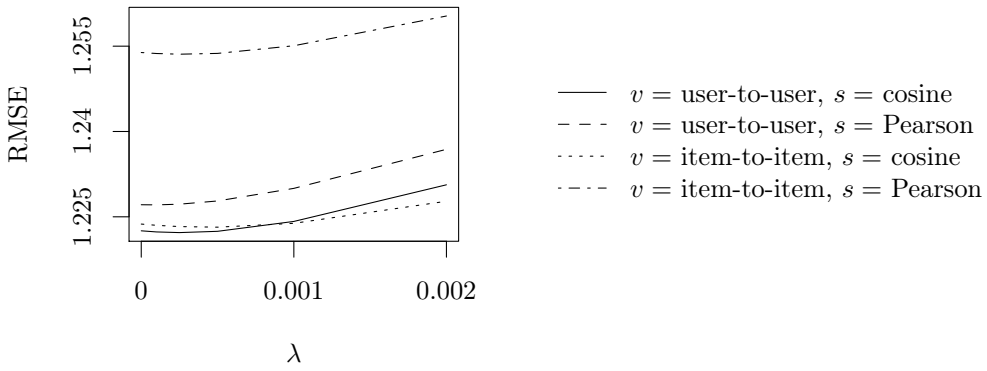

Figure 22: *RMSE of the temporal-recency models applied to the Yelp validation set*

Prediction performance We depict histograms of the collaborative filtering models' errors in Figure 23. Surprisingly, the largest error made by model E is bigger than 8, even tough this is hard to notice in the figure. This implies that this model predicts a rating that is relatively far outside the range $[1, 5]$, even though all the actual ratings are within this range. Apart from this anomaly, the distributions of the errors of both models do not seem to differ strongly.

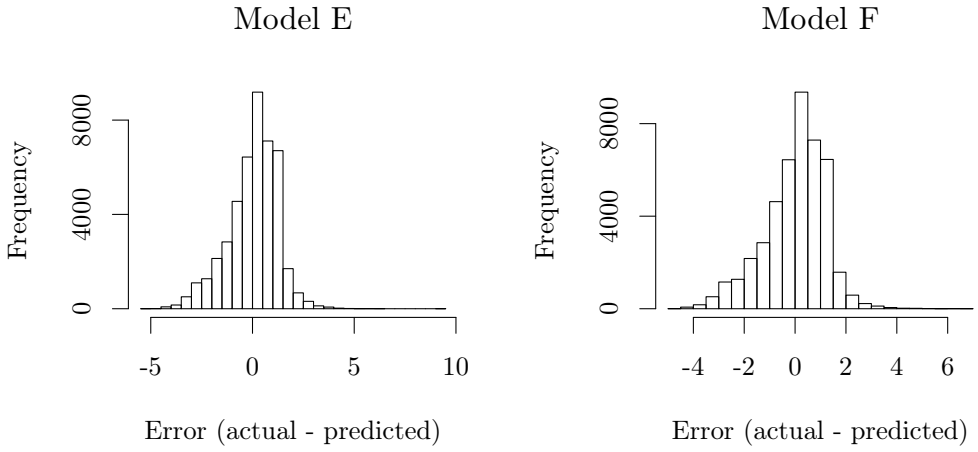

Figure 23: *Histograms of the errors of the collaborative filtering models applied to the Yelp test set*

In terms of the RMSE and MAE (presented in Table 8), the item-to-item variant performs slightly better than the user-to-user equivalent. Similar to what we found during the parameter validation, the item-mean popular algorithm also performs slightly better than the user-to-user model in terms of prediction performance.

Top-n characteristics The novelty, diversity and personality results are, for a large part, comparable to the MovieLens dataset: the collaborative filtering models do not differ strongly with each other and do differ with the item-mean popular algorithm. Again, the popular algorithm yields higher novelty and diversity, but considerably lower personality. However, the novelty and diversity values of the collaborative filtering models are relatively high compared to the MovieLens results. This is most likely a consequence of the lower density of Yelp's rating matrix. Due to this, the number of common ratings between items is generally lower and thus so are the similarities.

Table 8: *RMSE, MAE, novelty, diversity and personality of the models applied to the Yelp test set*

Name	RMSE	MAE	Novelty	Diversity	Personality
E	1.2125	0.9395	0.3337	0.4377	0.780
F	1.2029	0.9310	0.3259	0.4289	0.864
Item-mean popular	1.1958	0.9612	0.4702	0.5000	0.020

Binary interactions Similar to the MovieLens data, we apply the best performing model (model F) to the binary equivalent of the rating matrix. This results in a RMSE of 1.2873, which is 7.016% higher than when we apply the model to the non-binary matrix. We can therefore conclude that the ratings (and timestamps) seem to provide additional explanatory value over the number of common ratings between users or items.

6 Conclusion and Discussion

This research set out to improve the performance of neighbourhood-based collaborative filtering by including temporal information. We mainly focussed on prediction performance, but also briefly touched upon the novelty, diversity and personality of the recommendations. With regard to the temporal adaptations, the focus was on incorporating temporal similarity in the neighbourhood determination phase and the time distance from past ratings to the time of prediction in the rating aggregation phase. To evaluate the performance, we applied the time-aware algorithms as well as their time-unaware equivalents to data drawn from MovieLens and Yelp.

We found that generally the temporal-similarity and temporal-recency adaptations do not greatly increase the prediction performance of neighbourhood-based collaborative filtering. In many cases, the performance even slightly degraded. These findings suggest that the ratings already convey essentially all of the explanatory value for prediction performance. Hence, businesses looking to decrease the prediction errors of their recommender systems are most likely to benefit more from other approaches.

A possible explanation for the lacking effect of the temporal-similarity adaptation on the MovieLens dataset might be that neither the ratings nor the timestamps, but the binary interactions between users and items contain the explanatory value for the similarity computations. We did not find this explanation to be plausible for the Yelp data. Furthermore, it is difficult to explain the minor prediction performance degradation of the temporal-recency adaptation, but it might be related to the evaluation approach. The outcome of Ding and Li [2005] suggests that a leave-one-out procedure combined with distinct parameter values per user or item might lead to different results.

Based on the viewpoints of novelty, diversity and personality, we can not draw any decisive conclusions. In this research, the focus on these measures was simply too limited to be able to substantiate any conclusive claims. Further research could deeper investigate the temporal models by more strongly focusing on the novelty, diversity and personality measures. It would, for example, be informative to incorporate these measures in the parameter validation procedure. Furthermore, other performance characteristics might be usefully examined to possibly capture added value of the temporal adaptations. Examples not only include trust and serendipity [Ricci et al., 2015, p. 295, 297], but also the evolution of the recommendations over time.

Generally, the scope of this research was limited in terms of the analysis and interpretation of what exactly caused the found outcomes. Instead, the main focus was on descriptively reporting the results of the evaluation procedure. Further research is needed to thoroughly explain these results.

Future studies could also dive deeper into the subject of temporal information in various other ways. Firstly, different evaluation procedures and user or item dependent parameter values might produce interesting findings. In addition, further studies could investigate similarity functions that are specifically attuned to temporal data. Related to this would be the exploration of: applying other methods of combining the rating and timestamp similarity, blending different similarity measures and employing various recency-weight functions. More broadly, further experiments could usefully explore if the incorporation

of temporal information leads to better performance of other recommendation algorithms, such as matrix factorization and deep learning.

References

J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

P. G. Campos, A. Bellogín, F. Díez, and J. E. Chavarriaga. Simple time-biased knn-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 20–23, 2010.

C. A. Charu. *Recommender Systems Textbook*. Springer International Publishing, first edition, 2016. ISBN 978-3-319-29659-3.

Y. Ding and X. Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 485–492, 2005.

Y. Hu, Q. Peng, X. Hu, and R. Yang. Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. *IEEE Transactions on Services Computing*, 8(5):782–794, 2014.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

N. N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 95–102, 2010.

M. Organero, G. Ramirez-Gonzalez, P. Merino, and C. Delgado-Kloos. A collaborative recommender system based on space-time similarities. *IEEE Pervasive Computing*, 9:81–87, 2010.

F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*. Springer US, second edition, 2015. ISBN 978-1-4899-7637-6.

S. Wei, N. Ye, and Q. Zhang. Time-aware collaborative filtering for recommender systems. In *Pattern Recognition*, pages 663–670, 2012.

Appendices

A Parameter Validation Results

A.1 MovieLens

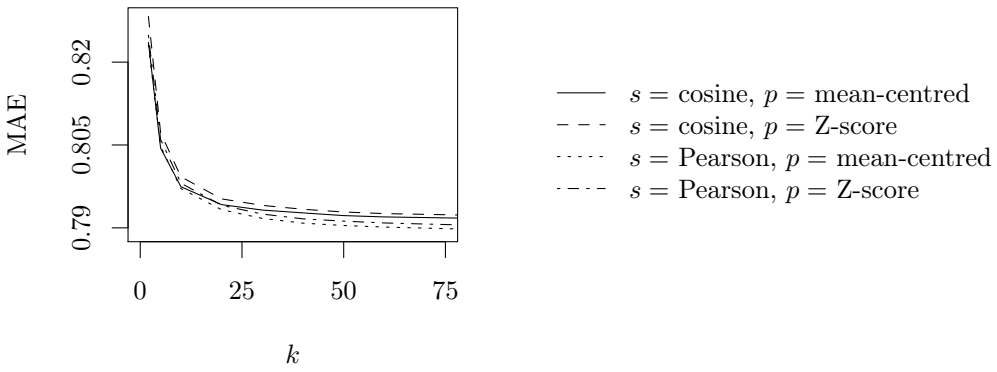

Figure 24: *MAE of the time-unaware user-to-user models applied to the MovieLens validation set*

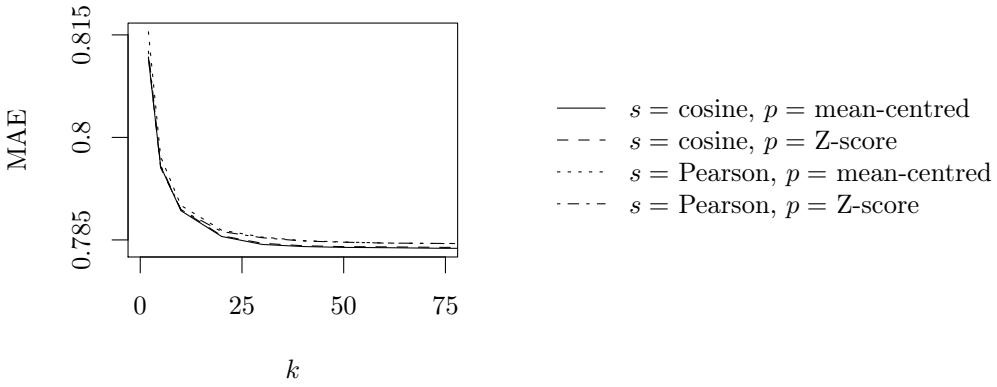

Figure 25: *MAE of the time-unaware item-to-item models applied to the MovieLens validation set*

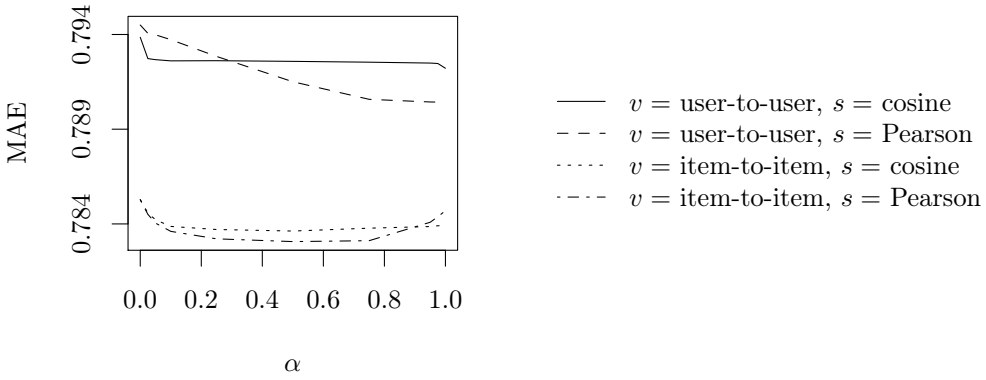

Figure 26: *MAE of the temporal-similarity models applied to the MovieLens validation set*

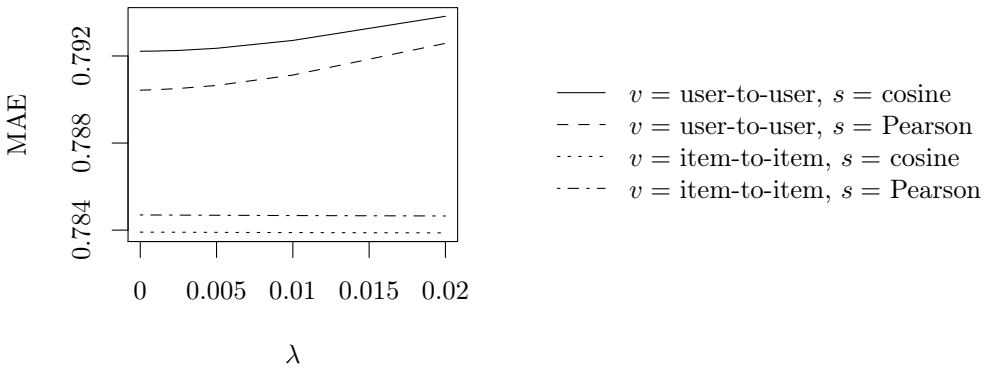

Figure 27: *MAE of the temporal-recency models applied to the MovieLens validation set*

Table 9: *RMSE and MAE of the time-unaware models applied to the MovieLens validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	2	mean-centred	1	0	1.0367	0.8236
user-to-user	cosine	2	Z-score	1	0	1.0472	0.8283
user-to-user	cosine	5	mean-centred	1	0	1.0099	0.8044
user-to-user	cosine	5	Z-score	1	0	1.0154	0.8072
user-to-user	cosine	10	mean-centred	1	0	1.0007	0.7974
user-to-user	cosine	10	Z-score	1	0	1.0042	0.7992
user-to-user	cosine	20	mean-centred	1	0	0.9965	0.7942
user-to-user	cosine	20	Z-score	1	0	0.9988	0.7952
user-to-user	cosine	30	mean-centred	1	0	0.9952	0.7932
user-to-user	cosine	30	Z-score	1	0	0.9970	0.7941
user-to-user	cosine	40	mean-centred	1	0	0.9945	0.7927
user-to-user	cosine	40	Z-score	1	0	0.9961	0.7934

user-to-user	cosine	50	mean-centred	1	0	0.9941	0.7922
user-to-user	cosine	50	Z-score	1	0	0.9957	0.7929
user-to-user	cosine	60	mean-centred	1	0	0.9938	0.7920
user-to-user	cosine	60	Z-score	1	0	0.9953	0.7926
user-to-user	cosine	75	mean-centred	1	0	0.9936	0.7918
user-to-user	cosine	75	Z-score	1	0	0.9950	0.7924
user-to-user	cosine	100	mean-centred	1	0	0.9934	0.7915
user-to-user	cosine	100	Z-score	1	0	0.9947	0.7920
user-to-user	cosine	150	mean-centred	1	0	0.9932	0.7913
user-to-user	cosine	150	Z-score	1	0	0.9944	0.7916
user-to-user	cosine	250	mean-centred	1	0	0.9930	0.7910
user-to-user	cosine	250	Z-score	1	0	0.9942	0.7913
user-to-user	Pearson	2	mean-centred	1	0	1.0366	0.8232
user-to-user	Pearson	2	Z-score	1	0	1.0419	0.8249
user-to-user	Pearson	5	mean-centred	1	0	1.0106	0.8047
user-to-user	Pearson	5	Z-score	1	0	1.0137	0.8060
user-to-user	Pearson	10	mean-centred	1	0	1.0008	0.7971
user-to-user	Pearson	10	Z-score	1	0	1.0032	0.7981
user-to-user	Pearson	20	mean-centred	1	0	0.9960	0.7933
user-to-user	Pearson	20	Z-score	1	0	0.9978	0.7941
user-to-user	Pearson	30	mean-centred	1	0	0.9941	0.7917
user-to-user	Pearson	30	Z-score	1	0	0.9958	0.7925
user-to-user	Pearson	40	mean-centred	1	0	0.9932	0.7909
user-to-user	Pearson	40	Z-score	1	0	0.9948	0.7916
user-to-user	Pearson	50	mean-centred	1	0	0.9927	0.7904
user-to-user	Pearson	50	Z-score	1	0	0.9943	0.7912
user-to-user	Pearson	60	mean-centred	1	0	0.9924	0.7901
user-to-user	Pearson	60	Z-score	1	0	0.9940	0.7909
user-to-user	Pearson	75	mean-centred	1	0	0.9921	0.7899
user-to-user	Pearson	75	Z-score	1	0	0.9936	0.7906
user-to-user	Pearson	100	mean-centred	1	0	0.9917	0.7895
user-to-user	Pearson	100	Z-score	1	0	0.9932	0.7902
user-to-user	Pearson	150	mean-centred	1	0	0.9915	0.7893
user-to-user	Pearson	150	Z-score	1	0	0.9930	0.7899
user-to-user	Pearson	250	mean-centred	1	0	0.9914	0.7891
user-to-user	Pearson	250	Z-score	1	0	0.9928	0.7897
item-to-item	cosine	2	mean-centred	1	0	1.0264	0.8116
item-to-item	cosine	2	Z-score	1	0	1.0274	0.8118
item-to-item	cosine	5	mean-centred	1	0	1.0039	0.7958
item-to-item	cosine	5	Z-score	1	0	1.0033	0.7956
item-to-item	cosine	10	mean-centred	1	0	0.9942	0.7893
item-to-item	cosine	10	Z-score	1	0	0.9944	0.7894
item-to-item	cosine	20	mean-centred	1	0	0.9881	0.7855
item-to-item	cosine	20	Z-score	1	0	0.9885	0.7856
item-to-item	cosine	30	mean-centred	1	0	0.9863	0.7843
item-to-item	cosine	30	Z-score	1	0	0.9867	0.7845
item-to-item	cosine	40	mean-centred	1	0	0.9858	0.7840
item-to-item	cosine	40	Z-score	1	0	0.9862	0.7842
item-to-item	cosine	50	mean-centred	1	0	0.9856	0.7839
item-to-item	cosine	50	Z-score	1	0	0.9860	0.7840

item-to-item	cosine	60	mean-centred	1	0	0.9855	0.7839
item-to-item	cosine	60	Z-score	1	0	0.9859	0.7840
item-to-item	cosine	75	mean-centred	1	0	0.9855	0.7838
item-to-item	cosine	75	Z-score	1	0	0.9859	0.7839
item-to-item	cosine	100	mean-centred	1	0	0.9855	0.7838
item-to-item	cosine	100	Z-score	1	0	0.9859	0.7840
item-to-item	cosine	150	mean-centred	1	0	0.9858	0.7841
item-to-item	cosine	150	Z-score	1	0	0.9863	0.7843
item-to-item	cosine	250	mean-centred	1	0	0.9864	0.7845
item-to-item	cosine	250	Z-score	1	0	0.9868	0.7848
item-to-item	Pearson	2	mean-centred	1	0	1.0309	0.8155
item-to-item	Pearson	2	Z-score	1	0	1.0276	0.8126
item-to-item	Pearson	5	mean-centred	1	0	1.0048	0.7972
item-to-item	Pearson	5	Z-score	1	0	1.0032	0.7959
item-to-item	Pearson	10	mean-centred	1	0	0.9948	0.7900
item-to-item	Pearson	10	Z-score	1	0	0.9941	0.7894
item-to-item	Pearson	20	mean-centred	1	0	0.9899	0.7864
item-to-item	Pearson	20	Z-score	1	0	0.9898	0.7862
item-to-item	Pearson	30	mean-centred	1	0	0.9887	0.7854
item-to-item	Pearson	30	Z-score	1	0	0.9887	0.7853
item-to-item	Pearson	40	mean-centred	1	0	0.9881	0.7849
item-to-item	Pearson	40	Z-score	1	0	0.9882	0.7848
item-to-item	Pearson	50	mean-centred	1	0	0.9878	0.7847
item-to-item	Pearson	50	Z-score	1	0	0.9879	0.7847
item-to-item	Pearson	60	mean-centred	1	0	0.9876	0.7846
item-to-item	Pearson	60	Z-score	1	0	0.9878	0.7845
item-to-item	Pearson	75	mean-centred	1	0	0.9875	0.7845
item-to-item	Pearson	75	Z-score	1	0	0.9877	0.7845
item-to-item	Pearson	100	mean-centred	1	0	0.9874	0.7845
item-to-item	Pearson	100	Z-score	1	0	0.9877	0.7845
item-to-item	Pearson	150	mean-centred	1	0	0.9875	0.7846
item-to-item	Pearson	150	Z-score	1	0	0.9877	0.7846
item-to-item	Pearson	250	mean-centred	1	0	0.9876	0.7847
item-to-item	Pearson	250	Z-score	1	0	0.9878	0.7847

Table 10: *RMSE and MAE of the temporal-similarity models applied to the MovieLens validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	50	mean-centred	0.000	0	0.9962	0.7939
user-to-user	cosine	50	mean-centred	0.025	0	0.9947	0.7927
user-to-user	cosine	50	mean-centred	0.050	0	0.9947	0.7927
user-to-user	cosine	50	mean-centred	0.100	0	0.9946	0.7926
user-to-user	cosine	50	mean-centred	0.250	0	0.9946	0.7926
user-to-user	cosine	50	mean-centred	0.500	0	0.9945	0.7926
user-to-user	cosine	50	mean-centred	0.750	0	0.9945	0.7925
user-to-user	cosine	50	mean-centred	0.950	0	0.9944	0.7925
user-to-user	cosine	50	mean-centred	0.975	0	0.9944	0.7925

user-to-user	cosine	50	mean-centred	1.000	0	0.9941	0.7922
user-to-user	Pearson	50	mean-centred	0.000	0	0.9974	0.7945
user-to-user	Pearson	50	mean-centred	0.025	0	0.9970	0.7941
user-to-user	Pearson	50	mean-centred	0.050	0	0.9969	0.7940
user-to-user	Pearson	50	mean-centred	0.100	0	0.9966	0.7937
user-to-user	Pearson	50	mean-centred	0.250	0	0.9956	0.7928
user-to-user	Pearson	50	mean-centred	0.500	0	0.9940	0.7915
user-to-user	Pearson	50	mean-centred	0.750	0	0.9929	0.7906
user-to-user	Pearson	50	mean-centred	0.950	0	0.9927	0.7904
user-to-user	Pearson	50	mean-centred	0.975	0	0.9927	0.7904
user-to-user	Pearson	50	mean-centred	1.000	0	0.9927	0.7904
item-to-item	cosine	50	mean-centred	0.000	0	0.9873	0.7853
item-to-item	cosine	50	mean-centred	0.025	0	0.9866	0.7846
item-to-item	cosine	50	mean-centred	0.050	0	0.9862	0.7842
item-to-item	cosine	50	mean-centred	0.100	0	0.9858	0.7839
item-to-item	cosine	50	mean-centred	0.250	0	0.9854	0.7837
item-to-item	cosine	50	mean-centred	0.500	0	0.9854	0.7836
item-to-item	cosine	50	mean-centred	0.750	0	0.9854	0.7838
item-to-item	cosine	50	mean-centred	0.950	0	0.9855	0.7839
item-to-item	cosine	50	mean-centred	0.975	0	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1.000	0	0.9856	0.7839
item-to-item	Pearson	50	mean-centred	0.000	0	0.9874	0.7853
item-to-item	Pearson	50	mean-centred	0.025	0	0.9866	0.7845
item-to-item	Pearson	50	mean-centred	0.050	0	0.9861	0.7840
item-to-item	Pearson	50	mean-centred	0.100	0	0.9857	0.7836
item-to-item	Pearson	50	mean-centred	0.250	0	0.9853	0.7832
item-to-item	Pearson	50	mean-centred	0.500	0	0.9851	0.7831
item-to-item	Pearson	50	mean-centred	0.750	0	0.9854	0.7831
item-to-item	Pearson	50	mean-centred	0.950	0	0.9869	0.7841
item-to-item	Pearson	50	mean-centred	0.975	0	0.9874	0.7844
item-to-item	Pearson	50	mean-centred	1.000	0	0.9878	0.7847

Table 11: *RMSE and MAE of the temporal-recency models applied to the MovieLens validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	50	mean-centred	1	0.0000	0.9941	0.7922
user-to-user	cosine	50	mean-centred	1	0.0010	0.9941	0.7922
user-to-user	cosine	50	mean-centred	1	0.0025	0.9941	0.7923
user-to-user	cosine	50	mean-centred	1	0.0050	0.9942	0.7924
user-to-user	cosine	50	mean-centred	1	0.0100	0.9946	0.7927
user-to-user	cosine	50	mean-centred	1	0.0200	0.9959	0.7938
user-to-user	Pearson	50	mean-centred	1	0.0000	0.9927	0.7904
user-to-user	Pearson	50	mean-centred	1	0.0010	0.9927	0.7905
user-to-user	Pearson	50	mean-centred	1	0.0025	0.9927	0.7905
user-to-user	Pearson	50	mean-centred	1	0.0050	0.9929	0.7906
user-to-user	Pearson	50	mean-centred	1	0.0100	0.9935	0.7911
user-to-user	Pearson	50	mean-centred	1	0.0200	0.9953	0.7926

item-to-item	cosine	50	mean-centred	1	0.0000	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1	0.0010	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1	0.0025	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1	0.0050	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1	0.0100	0.9856	0.7839
item-to-item	cosine	50	mean-centred	1	0.0200	0.9856	0.7839
item-to-item	Pearson	50	mean-centred	1	0.0000	0.9878	0.7847
item-to-item	Pearson	50	mean-centred	1	0.0010	0.9878	0.7847
item-to-item	Pearson	50	mean-centred	1	0.0025	0.9878	0.7847
item-to-item	Pearson	50	mean-centred	1	0.0050	0.9878	0.7847
item-to-item	Pearson	50	mean-centred	1	0.0100	0.9878	0.7847
item-to-item	Pearson	50	mean-centred	1	0.0200	0.9878	0.7847

A.2 Yelp

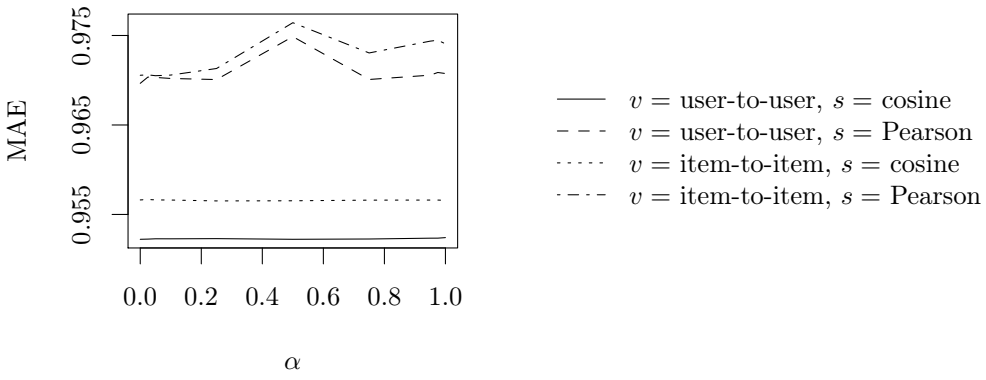

Figure 28: *MAE of the temporal-similarity applied to the Yelp validation set*

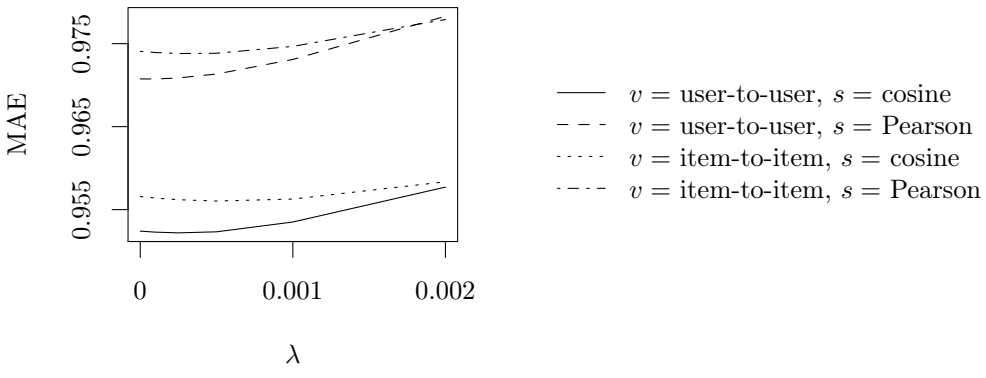

Figure 29: *MAE of the temporal-recency models applied to the Yelp validation set*

Table 12: *RMSE and MAE of the time-unaware models applied to the Yelp validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	2	mean-centred	1	0	1.3450	1.0626
user-to-user	cosine	2	Z-score	1	0	1.3418	1.0362
user-to-user	cosine	5	mean-centred	1	0	1.2703	1.0026
user-to-user	cosine	5	Z-score	1	0	1.2607	0.9797
user-to-user	cosine	10	mean-centred	1	0	1.2468	0.9824
user-to-user	cosine	10	Z-score	1	0	1.2361	0.9620
user-to-user	cosine	20	mean-centred	1	0	1.2375	0.9738
user-to-user	cosine	20	Z-score	1	0	1.2267	0.9552
user-to-user	cosine	30	mean-centred	1	0	1.2351	0.9720
user-to-user	cosine	30	Z-score	1	0	1.2244	0.9540
user-to-user	cosine	40	mean-centred	1	0	1.2338	0.9709
user-to-user	cosine	40	Z-score	1	0	1.2232	0.9529
user-to-user	cosine	50	mean-centred	1	0	1.2332	0.9702
user-to-user	cosine	50	Z-score	1	0	1.2225	0.9524
user-to-user	cosine	60	mean-centred	1	0	1.2329	0.9699
user-to-user	cosine	60	Z-score	1	0	1.2223	0.9521
user-to-user	cosine	75	mean-centred	1	0	1.2327	0.9697
user-to-user	cosine	75	Z-score	1	0	1.2222	0.9520
user-to-user	cosine	100	mean-centred	1	0	1.2324	0.9694
user-to-user	cosine	100	Z-score	1	0	1.2219	0.9517
user-to-user	cosine	150	mean-centred	1	0	1.2323	0.9693
user-to-user	cosine	150	Z-score	1	0	1.2220	0.9518
user-to-user	cosine	250	mean-centred	1	0	1.2323	0.9693
user-to-user	cosine	250	Z-score	1	0	1.2221	0.9518
user-to-user	Pearson	2	mean-centred	1	0	1.3178	1.0452
user-to-user	Pearson	2	Z-score	1	0	1.3414	1.0415
user-to-user	Pearson	5	mean-centred	1	0	1.2563	0.9961
user-to-user	Pearson	5	Z-score	1	0	1.2719	0.9927
user-to-user	Pearson	10	mean-centred	1	0	1.2371	0.9796
user-to-user	Pearson	10	Z-score	1	0	1.2519	0.9773
user-to-user	Pearson	20	mean-centred	1	0	1.2299	0.9735
user-to-user	Pearson	20	Z-score	1	0	1.2445	0.9720
user-to-user	Pearson	30	mean-centred	1	0	1.2279	0.9717
user-to-user	Pearson	30	Z-score	1	0	1.2424	0.9702
user-to-user	Pearson	40	mean-centred	1	0	1.2273	0.9710
user-to-user	Pearson	40	Z-score	1	0	1.2420	0.9696
user-to-user	Pearson	50	mean-centred	1	0	1.2271	0.9707
user-to-user	Pearson	50	Z-score	1	0	1.2417	0.9694
user-to-user	Pearson	60	mean-centred	1	0	1.2271	0.9707
user-to-user	Pearson	60	Z-score	1	0	1.2417	0.9693
user-to-user	Pearson	75	mean-centred	1	0	1.2269	0.9705
user-to-user	Pearson	75	Z-score	1	0	1.2415	0.9691
user-to-user	Pearson	100	mean-centred	1	0	1.2268	0.9705
user-to-user	Pearson	100	Z-score	1	0	1.2415	0.9691
user-to-user	Pearson	150	mean-centred	1	0	1.2268	0.9704

user-to-user	Pearson	150	Z-score	1	0	1.2415	0.9690
user-to-user	Pearson	250	mean-centred	1	0	1.2268	0.9704
user-to-user	Pearson	250	Z-score	1	0	1.2415	0.9690
item-to-item	cosine	2	mean-centred	1	0	1.3301	1.0420
item-to-item	cosine	2	Z-score	1	0	1.3259	1.0297
item-to-item	cosine	5	mean-centred	1	0	1.2514	0.9844
item-to-item	cosine	5	Z-score	1	0	1.2504	0.9770
item-to-item	cosine	10	mean-centred	1	0	1.2308	0.9687
item-to-item	cosine	10	Z-score	1	0	1.2313	0.9632
item-to-item	cosine	20	mean-centred	1	0	1.2251	0.9637
item-to-item	cosine	20	Z-score	1	0	1.2258	0.9587
item-to-item	cosine	30	mean-centred	1	0	1.2237	0.9623
item-to-item	cosine	30	Z-score	1	0	1.2244	0.9572
item-to-item	cosine	40	mean-centred	1	0	1.2233	0.9620
item-to-item	cosine	40	Z-score	1	0	1.2240	0.9569
item-to-item	cosine	50	mean-centred	1	0	1.2230	0.9617
item-to-item	cosine	50	Z-score	1	0	1.2237	0.9566
item-to-item	cosine	60	mean-centred	1	0	1.2229	0.9616
item-to-item	cosine	60	Z-score	1	0	1.2237	0.9565
item-to-item	cosine	75	mean-centred	1	0	1.2229	0.9615
item-to-item	cosine	75	Z-score	1	0	1.2237	0.9564
item-to-item	cosine	100	mean-centred	1	0	1.2228	0.9614
item-to-item	cosine	100	Z-score	1	0	1.2236	0.9563
item-to-item	cosine	150	mean-centred	1	0	1.2228	0.9613
item-to-item	cosine	150	Z-score	1	0	1.2236	0.9563
item-to-item	cosine	250	mean-centred	1	0	1.2228	0.9613
item-to-item	cosine	250	Z-score	1	0	1.2236	0.9563
item-to-item	Pearson	2	mean-centred	1	0	1.3293	1.0389
item-to-item	Pearson	2	Z-score	1	0	1.3235	1.0263
item-to-item	Pearson	5	mean-centred	1	0	1.2702	0.9942
item-to-item	Pearson	5	Z-score	1	0	1.2689	0.9869
item-to-item	Pearson	10	mean-centred	1	0	1.2581	0.9835
item-to-item	Pearson	10	Z-score	1	0	1.2577	0.9777
item-to-item	Pearson	20	mean-centred	1	0	1.2550	0.9807
item-to-item	Pearson	20	Z-score	1	0	1.2549	0.9753
item-to-item	Pearson	30	mean-centred	1	0	1.2543	0.9799
item-to-item	Pearson	30	Z-score	1	0	1.2542	0.9745
item-to-item	Pearson	40	mean-centred	1	0	1.2540	0.9795
item-to-item	Pearson	40	Z-score	1	0	1.2540	0.9741
item-to-item	Pearson	50	mean-centred	1	0	1.2540	0.9794
item-to-item	Pearson	50	Z-score	1	0	1.2539	0.9741
item-to-item	Pearson	60	mean-centred	1	0	1.2539	0.9794
item-to-item	Pearson	60	Z-score	1	0	1.2539	0.9740
item-to-item	Pearson	75	mean-centred	1	0	1.2539	0.9793
item-to-item	Pearson	75	Z-score	1	0	1.2538	0.9740
item-to-item	Pearson	100	mean-centred	1	0	1.2539	0.9793
item-to-item	Pearson	100	Z-score	1	0	1.2538	0.9740
item-to-item	Pearson	150	mean-centred	1	0	1.2539	0.9793
item-to-item	Pearson	150	Z-score	1	0	1.2538	0.9740
item-to-item	Pearson	250	mean-centred	1	0	1.2539	0.9793

item-to-item	Pearson	250	Z-score	1	0	1.2538	0.9739

Table 13: *RMSE and MAE of the temporal-similarity models applied to the Yelp validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	50	Z-score	0.000	0	1.2224	0.9522
user-to-user	cosine	50	Z-score	0.025	0	1.2224	0.9522
user-to-user	cosine	50	Z-score	0.050	0	1.2224	0.9523
user-to-user	cosine	50	Z-score	0.100	0	1.2224	0.9523
user-to-user	cosine	50	Z-score	0.250	0	1.2225	0.9523
user-to-user	cosine	50	Z-score	0.500	0	1.2223	0.9522
user-to-user	cosine	50	Z-score	0.750	0	1.2224	0.9523
user-to-user	cosine	50	Z-score	0.950	0	1.2225	0.9523
user-to-user	cosine	50	Z-score	0.975	0	1.2225	0.9524
user-to-user	cosine	50	Z-score	1.000	0	1.2225	0.9524
user-to-user	Pearson	50	mean-centred	0.000	0	1.2272	0.9697
user-to-user	Pearson	50	mean-centred	0.025	0	1.2283	0.9704
user-to-user	Pearson	50	mean-centred	0.050	0	1.2282	0.9703
user-to-user	Pearson	50	mean-centred	0.100	0	1.2281	0.9702
user-to-user	Pearson	50	mean-centred	0.250	0	1.2283	0.9701
user-to-user	Pearson	50	mean-centred	0.500	0	1.2356	0.9749
user-to-user	Pearson	50	mean-centred	0.750	0	1.2292	0.9701
user-to-user	Pearson	50	mean-centred	0.950	0	1.2300	0.9706
user-to-user	Pearson	50	mean-centred	0.975	0	1.2304	0.9709
user-to-user	Pearson	50	mean-centred	1.000	0	1.2271	0.9707
item-to-item	cosine	50	Z-score	0.000	0	1.2239	0.9566
item-to-item	cosine	50	Z-score	0.025	0	1.2239	0.9567
item-to-item	cosine	50	Z-score	0.050	0	1.2238	0.9566
item-to-item	cosine	50	Z-score	0.100	0	1.2238	0.9566
item-to-item	cosine	50	Z-score	0.250	0	1.2237	0.9565
item-to-item	cosine	50	Z-score	0.500	0	1.2237	0.9565
item-to-item	cosine	50	Z-score	0.750	0	1.2237	0.9566
item-to-item	cosine	50	Z-score	0.950	0	1.2237	0.9566
item-to-item	cosine	50	Z-score	0.975	0	1.2237	0.9566
item-to-item	cosine	50	Z-score	1.000	0	1.2237	0.9566
item-to-item	Pearson	50	Z-score	0.000	0	1.2462	0.9706
item-to-item	Pearson	50	Z-score	0.025	0	1.2463	0.9706
item-to-item	Pearson	50	Z-score	0.050	0	1.2462	0.9705
item-to-item	Pearson	50	Z-score	0.100	0	1.2464	0.9706
item-to-item	Pearson	50	Z-score	0.250	0	1.2478	0.9713
item-to-item	Pearson	50	Z-score	0.500	0	1.2571	0.9764
item-to-item	Pearson	50	Z-score	0.750	0	1.2528	0.9731
item-to-item	Pearson	50	Z-score	0.950	0	1.2541	0.9744
item-to-item	Pearson	50	Z-score	0.975	0	1.2544	0.9745
item-to-item	Pearson	50	Z-score	1.000	0	1.2539	0.9741

Table 14: *RMSE and MAE of the temporal-recency models applied to the Yelp validation set*

v	s	k	p	α	λ	RMSE	MAE
user-to-user	cosine	50	Z-score	1	0.00000	1.2225	0.9524
user-to-user	cosine	50	Z-score	1	0.00010	1.2223	0.9523
user-to-user	cosine	50	Z-score	1	0.00025	1.2222	0.9522
user-to-user	cosine	50	Z-score	1	0.00050	1.2224	0.9523
user-to-user	cosine	50	Z-score	1	0.00100	1.2242	0.9535
user-to-user	cosine	50	Z-score	1	0.00200	1.2306	0.9577
user-to-user	Pearson	50	mean-centred	1	0.00000	1.2271	0.9707
user-to-user	Pearson	50	mean-centred	1	0.00010	1.2271	0.9707
user-to-user	Pearson	50	mean-centred	1	0.00025	1.2272	0.9709
user-to-user	Pearson	50	mean-centred	1	0.00050	1.2278	0.9713
user-to-user	Pearson	50	mean-centred	1	0.00100	1.2300	0.9731
user-to-user	Pearson	50	mean-centred	1	0.00200	1.2369	0.9783
item-to-item	cosine	50	Z-score	1	0.00000	1.2237	0.9566
item-to-item	cosine	50	Z-score	1	0.00010	1.2235	0.9564
item-to-item	cosine	50	Z-score	1	0.00025	1.2233	0.9562
item-to-item	cosine	50	Z-score	1	0.00050	1.2232	0.9560
item-to-item	cosine	50	Z-score	1	0.00100	1.2239	0.9563
item-to-item	cosine	50	Z-score	1	0.00200	1.2277	0.9584
item-to-item	Pearson	50	Z-score	1	0.00000	1.2539	0.9741
item-to-item	Pearson	50	Z-score	1	0.00010	1.2537	0.9739
item-to-item	Pearson	50	Z-score	1	0.00025	1.2536	0.9738
item-to-item	Pearson	50	Z-score	1	0.00050	1.2537	0.9738
item-to-item	Pearson	50	Z-score	1	0.00100	1.2551	0.9747
item-to-item	Pearson	50	Z-score	1	0.00200	1.2603	0.9779