TCNSVD: A Temporal and Community-Aware Recommender Approach

Mohsen Shahriari
RWTH Aachen University
shahriari@dbis.rwth-aachen.de

Martin Barth
RWTH Aachen University
barth@dbis.rwth-aachen.de

Ralf Klamma
RWTH Aachen University
klamma@dbis.rwth-aachen.de

Christoph Trattner
MODUL University Vienna
christoph.trattner@modul.ac.at

ABSTRACT

Recommender systems support users in finding relevant items in overloaded information spaces. Researchers and practitioners have proposed many different collaborative filtering algorithms for different information scenarios, domains and contexts. One of the latter, are time-aware recommender methods that consider temporal dynamics in the users' interests in certain items, topics, etc. While there is extensive research on time-aware recommender systems, surprisingly, researchers have paid little attention to model temporal community structure dynamics (community drift). In consequence, recommender systems seldom exploit explicit and implicit community structures that are present in online systems, where one can see what others have been watching, sharing and or tagging. In this paper, we propose a recommender method that not only considers temporal interest dynamics in online communities, but also exploits the social structure by the means of community detection algorithms. We conducted offline experiments on the Netflix dataset and the latest MovieLens dataset with tag information. Our method outperformed the current state-of-the-art in rating and item-ranking prediction. This work contributes to the connection of two separate recommender research directions, in which exploits community structure and temporal effects together in recommender systems.

KEYWORDS

Collaborative Filtering, Community Detection, Community Drift, Time-Aware Recommender Models

ACM Reference format:
Mohsen Shahriari, Martin Barth, Ralf Klamma, and Christoph Trattner. 2017. TCNSVD: A Temporal and Community-Aware Recommender Approach. In *Proceedings of Workshop on Temporal Reasoning in Recommender Systems, Como, Italy, August 2017 (RecTemp'17),* 7 pages.
DOI: 10.475/123_4

RecTemp'17, Como, Italy

DOI: 10.475/123_4

1 INTRODUCTION

Since many years, recommender systems based on collaborative filtering techniques provide recommendations for us by applying specific approaches on a huge rating matrix. However, it is expensive to create and maintain such huge rating matrices for online shops and rating websites. From our own experience, we know that we do not update ratings when our judgment has changed due to changes in our taste, nor do we reflect that our ratings are based on the influence of somebody we know. But, research has shown that we can increase the recommendation accuracy by taking into account temporal effects with computational methods e.g. changes in the user preferences or the item popularity over time. This has lead to the development of time-aware recommender systems [21]. In parallel, the social network research community has investigated the detection and analysis of community structures as implicit influence among members of a social network [14]. Members of a community are supposed to possess similar properties so that they form dense connections inside communities and sparse connections among them. Correspondingly, more and more recommender systems consider community structures to e.g. improve accuracy [9]. However, one important property of social network research is still missing in recommender research, namely the temporal dynamics of online community structures [1, 22]. Temporal community structures - detected from explicit and implicit users' interactions and item-item networks - provide dynamics of collective information carried by groups of people. These communities are dynamic similar to the network, in which this needs to be reflected in recommender systems. To the best of our knowledge, there is still no other work that explains to what extent temporal dynamics of online communities can be effective in the proposal of a recommender system.

Objective. In this line of research, we propose two recommender models named CNSVD and TCNSVD. CNSVD considers the collective user preferences and item receptions at the same time. TCNSVD, on the other hand, includes temporal dynamics of (overlapping) community structures, which is not yet addressed by the research community. These models are extensions to the NSVD and TNSVD models proposed by Koren [20, 21] - leaning upon neighborhood and factor models of recommendation. Using user-user and item-item networks contributes to the evaluation of (overlapping) community detection algorithms as well as the graph construction methods. Furthermore, we perform extensive studies of the

proposed models on two large-scale and popular datasets - MovieLens and Netflix - and compare them with the state-of-the-art approaches.

2 RELATED WORK

In this paper, the proposed algorithms are related to both time-aware and community-aware recommender models. As such the related work in the area is shortly reviewed.

Time-Aware Recommendation. Some research has been done in the proposal of recommender systems dealing with temporal effects in recent years. Campos et al. [5] presented a survey and analysis of time-aware recommender systems. They claimed that time is one of the most useful contextual dimensions in recommender systems. Koren [21] applied a model-based collaborative filtering approach using a combination of neighborhood and factor models. He used models that track temporal shifts over several relevant characteristics e.g. user and item biases, covering both long-term and short-term temporal effects. Daneshmand et al. [10] assumed a hidden item network structure that can be inferred from users' sequences of selecting items. The basic idea is that if two items in the hidden network are related, then a user selecting one of the two items is likely to also select the other item. Baltrunas and Amatriain [2] proposed a slightly different approach to time dependency in recommendation, and assumed time-changing but repetitive user preferences. They recommended music, which depends on the time of day, week or year using a collaborative filtering approach. Charlin et al. proposed a dynamic matrix factorization model based on Poisson factorization for recommendation, which considers temporal users' interests and item popularity [8].

Community-Aware Recommendation. There has also been work on using community detection for recommendation tasks. Dolgikh and Jelinek [11] proposed an approach to recommend music using community detection. They constructed an artist-tag network for each user from the user's favorite artists and the tags assigned to these artists. Community detection was applied on these networks to determine each user's interest subfield, which were then used for recommending artists to the user. Cao et al. [7] applied a neighborhood-based collaborative filtering approach for recommending movies to users. They reduced computation time and improved recommendation precision by using community structures. They applied a community detection algorithm on the network constructed from similarity among users which was derived from the user-item ratings. Choo et al. [9] proposed a neighborhood-based collaborative filtering approach that uses user communities. The user network was derived from review-reply patterns, i.e. if one user reviews an item and another user replies to that review then there may be a relation between the two users. User communities were derived from this network and used as a basis for the recommendation process. Bellogin and Parapar [3] constructed a user graph using Pearson correlation similarity and applied normalized graph cuts to find clusters of users. These clusters were then used for neighbor selection in user-based collaborative filtering. Other approaches using community structures alleviated the cold-start problem in collaborative filtering. [25] and [30] both addressed the cold-start problem for new users by taking into account additional user information.

Summary. To the best of our knowledge, there are no approaches that take into account temporal dynamics of community structures to support recommendation. Thus, the literature manifests that 1) our knowledge regarding performance of (overlapping) community structures on recommendation systems is imperceptible. 2) we are not aware about the goodness of graph construction similarity metrics, e.g. Jaccard, Cosine, etc, in time-evolving recommender models. 3) we also know very little regarding the effect of metadata information on graph construction in temporal community-aware recommender systems. 4) we need models to connect temporal dynamics with (overlapping) community structures to improve item ranking and precision accuracy metrics in recommender systems.

3 PROPOSED RECOMMENDER MODELS

In this section, we introduce the proposed recommender models. Koren [20] employed the neighborhood and factor models to compute the rating of a specific user on a particular item. In this model, weights of user or item similarities are interpreted as offsets that need to be added to a baseline estimation. In other words, this approach combines three components including a baseline estimation, a neighborhood and a factor model, in which can be written as follows:

$$\boxed{\mu + b_u + b_i} + \boxed{|I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj}) w_{ij} + c_{ij})} + \boxed{q_i^T (p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j)}, \tag{1}$$

where

- the first block of terms refers to the baseline estimation, in which μ is the average rating over all users and items and b_u is the user bias, i.e. the deviation of the average rating given by user u from μ. Besides, b_i is the deviation of the average rating given to item i from μ (item bias).
- the second block of terms refers to the neighborhood model contribution, in which I_u is the set of items rated by user u, w_{ij} relates to explicit rating feedback, which is multiplied by $r_{uj} - b_{uj}$, and c_{ij} is related to implicit feedback and is added whenever user u has given a rating to item j. In addition, to avoid overestimation of the rating of users who provide much feedback, i.e. $|I_u|$ is high, the estimation is scaled down by multiplying with $|I_u|^{-\frac{1}{2}}$.
- finally the last block of terms indicates the factor model contribution, in which q_i and p_u are vectors characterizing item i and user u, respectively. Moreover, the user preference vector u is complemented by a sum of vectors y_j, that represent implicit feedback from each item $j \in I_u$.

This model is named Neighborhood-Integrated SVD (NSVD), in which parameters, i.e. b_u, b_i, w_{ij}, c_{ij}, y_{ij}, q_i and p_u are learned by minimizing a squared error function. Koren extended the NSVD model by using temporal information to improve recommendation accuracy. The temporal information allows the modeling of user preferences and item characteristics that change over time [21].

Temporal information was included into each of the three components i.e., baseline estimation, neighborhood model and factor models as follows:

$$\boxed{\mu + b_u(t) + b_i(t)} \tag{2}$$

$$+\boxed{|I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_u \cdot |t-t_{uj}|}((r_{uj} - b_{uj})w_{ij} + c_{ij})} \tag{3}$$

$$+\boxed{q_i^T (p_u(t) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j)}, \tag{4}$$

where

- the first block of terms refers to the time-aware baseline estimation. $b_u(t)$ and $b_i(t)$ indicate the time-dependent user and item bias at time t, in which $b_u(t)$ can be computed as with $b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t}$. Here, b_u, $\alpha_u \cdot dev_u(t)$ and $b_{u,t}$ describe time-independent user bias, the linear drift of the user bias and a time-specific parameter capturing short-lived effects. Moreover, item characteristics are less complex to be described and thus $b_i(t)$ can be simply computed by $b_i + b_{i,\text{Bin}(t)}$, in which short-lived effects are captured through time by $b_{i,\text{Bin}(t)}$.
- the second block of terms describes the time-aware neighborhood model contribution where the function $e^{-\phi_u \cdot |t-t_j|}$ decays the item contributions w_{ij} and c_{ij} such that ratings that are more distant to time t have less impact on the estimation.
- the last block of terms indicates the contribution of the time-aware factor model. $p_u(t)$ represents the time-dependent user preferences that can be captured through parameters including time-independent preference, gradual and short-lived effects.

This model is known as Time-aware Neighborhood-Integrated SVD (TNSVD), in which parameters, i.e. b_u, $b_{u,t}$, b_i, $b_{i,\text{Bin}(t)}$, w_{ij}, c_{ij}, y_{ij}, α_u, α_{uk}, ϕ_u, q_i, p_{uk} and $p_{uk,t}$ are again learned by minimizing a squared error function. In the following subsections, we introduce two models based on NSVD and TNSVD models.

3.1 Community-Aware NSVD Model

The community-aware neighborhood-integrated SVD (CNSVD) model is an extension of the NSVD model that use community information to improve rating prediction accuracy. Similar to the NSVD model, it consists of baseline estimation, neighborhood and factor model contributions.

Baseline Estimation. For the baseline estimation, we presume that user and item communities have their own bias. The average rating of a user community tends to deviate from the overall average rating and each user's average rating tends to deviate from the community's rating, and likewise with item communities. This is based on the assumption that users or items in a community have similar preferences or characteristics, which may imply a common bias and thus the baseline estimation b_{ui}, is extended as follows:

$$\mu + b_u + \boxed{b_{C_u}} + b_i + \boxed{b_{C_i}}, \tag{5}$$

, in which

- b_{C_u} shows the community bias for user u, in which C_u represents the set of user communities that user u belongs to. We model the user community bias as follows:

$$b_{C_u} = \sum_{C \in C_u} b_C \cdot m_{uC}, \tag{6}$$

where we sum over all communities that user u belongs to, in other words, we add the corresponding biases b_C weighted by the user's membership level m_{uC} regarding each community.
- b_{C_i} refers to the community bias for item i with C_i as the set of item communities that i belongs to. Similarly, we define the item community bias as follows:

$$b_{C_i} = \sum_{C \in C_i} b_C \cdot m_{iC}, \tag{7}$$

where b_C shows the community bias of community C, in which m_{iC} represents the membership level of item i belonging to community C.

Neighborhood Model. To use user community information for the neighborhood model, we extend the original neighborhood model to capture additional implicit feedback from each item that has been rated by a member of one of the user's communities. The extension is as follows:

$$|I_u|^{-\frac{1}{2}} \sum_{j \in I_u} ((r_{uj} - b_{uj})w_{ij} + c_{ij}) + \boxed{|I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} d_{ij}}, \tag{8}$$

where the block of terms shows the community contribution in the neighborhood model. Here, I_{C_u} represents the set of items that any user belonging to one of user u's communities has rated. Then, d_{ij} shows the offset that such an item j contributes to the rating for item i.

Latent Factor Model. For the latent factor model, we again consider the community information as implicit feedback, in which each item rated by a user's community contributes to the user's preference vector p_u. Using the original factor model, we can extend it as follows:

$$\begin{aligned}(q_i + \boxed{\sum_{C \in C_i} o_C \cdot m_{iC}}+)^T (p_u + \boxed{\sum_{C \in C_u} o_C \cdot m_{uC}}+ + \\ |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + \boxed{|I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j}),\end{aligned} \tag{9}$$

where the vector z_j represents the contribution from item j. Also, to model the user's communities, we introduce an additional vector o_C that represents the preferences or characteristics of community C. Since a user can belong to multiple communities, we define o_{C_u} as the combined community preferences of all communities that user u belongs to. This is achieved by summing the community factors o_C weighted by the user's membership level m_{uC} to each community. Likewise, to model the item community characteristics, we define o_{C_i} as the combined characteristics of the communities that item i belongs to. Combining baseline estimation, neighborhood model and factor model from equations 5, 8 and 9, we compute the predicted rating $\hat{r}_{ui}$ from a user u to an item i.

3.2 Time and Community-Aware NSVD (TCNSVD) Model

The TCNSVD model is an extension of the TNSVD model that uses temporal dynamics of community structures. To capture user and item community drift, i.e. time-changing community structures, we compute user and item graphs and their community structures for several time ranges. Here, available time range is divided into community time bins. The set of communities of a user u at time t is represented by $C_{u,\text{CBin}(t)}$ and the corresponding set of item communities is shown by $C_{i,\text{CBin}(t)}$, where $\text{CBin}(t)$ indicates a function that returns the community time bin for a given time t. TCNSVD model has three components that are explained in the following.

Baseline Estimation. For the baseline estimation, we extend the Equation 5 to capture time-dependent community biases and a time-dependent linear drift in community biases and thus we write the baseline estimation for TCNSVD model as follows:

$$\begin{aligned}&\mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} \\ &+ b_{u,\text{Period(t)}} + b_i + b_{i,\text{Bin}(t)} + b_{i,\text{Period(t)}} \\ &+ \boxed{\sum_{C \in C_{u,\text{CBin}(t)}} (b_C + b_{C,t}) \cdot m_{uC}} + \boxed{\sum_{C \in C_u} \alpha_C \cdot \text{dev}_C(t) \cdot m_{uC}} + \\ &\boxed{\sum_{C \in C_{i,\text{CBin}(t)}} (b_C + b_{C,\text{Bin}(t)}) \cdot m_{iC}},\end{aligned} \tag{10}$$

in which the block of terms shows the community contributions. In this formula, the time-independent bias of community C is denoted as b_C, and community bias based on short-lived effects at time t is denoted as $b_{C,t}$ for user communities and $b_{C,\text{CBin}(t)}$ for item communities. Linear drift in community biases is captured by $\alpha_C \cdot \text{dev}_C(t)$. All community biases are summed over the respective set of communities and weighted by the respective user's or item's community membership level. For the time-independent biases and the short-lived temporal effects, we use the dynamic community structures. However, for the linear drift we use static community structures so that longer-term temporal effects on each community can be captured. In addition, the periodic user and item biases are also reflected by $b_{u,\text{Period(t)}}$ and $b_{i,\text{Period(t)}}$, where Period(t) represents a function that returns an index showing the week day of time t. To keep our model from getting overly complex, we capture only one of these potential recurring temporal effects, namely weekly recurring effects.

Neighborhood Model. For the neighborhood model, we use a decay function $e^{-\psi_u \cdot |t-t_j|}$ on the additional implicit feedback d_{ij}, which was added to Equation 8. This leads to the following formula:

$$\begin{aligned}&|I_u|^{-\frac{1}{2}} \sum_{j \in I_u} e^{-\phi_u \cdot |t-t_j|}((r_{uj} - b_{uj})w_{ij} + c_{ij}) + \\ &\boxed{|I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} e^{-\psi_u \cdot |t-t_j|} d_{ij}}.\end{aligned} \tag{11}$$

Latent Factor Model. For the latent factor model, we change the vector modeling the user community preferences o_{Cu}, which was introduced in Equation 8 to being a function $o_{C_u}(t)$:

$$o_{C_u}(t) = \sum_{C \in C_u} o_C(t) \cdot, m_{uC}, \tag{12}$$

with the community preference vector o_C being replaced by the time-dependent vector function $o_C(t)$. As with the user preferences vector function, $o_C(t)$ is made up of multiple functions, each representing a component of the vector, i.e. $o_C(t)^T = (o_{C1}(t), o_{C2}(t), \ldots, o_{Cn}(t))$. Each component is defined as:

$$o_{Ck}(t) = o_{Ck} + \alpha_{Ck} \cdot \text{dev}_C(t) + o_{Ck,t} \quad k = 1, \ldots, n, \tag{13}$$

where o_{Ck} is the time-independent part of the community preferences, $o_{Ck,t}$ captures short-term effects and $\alpha_{Ck} \cdot \text{dev}_C(t)$ models a linear shift of community preferences. Using the user community preferences, we extend the formula for the factor model as follows:

$$\hat{r}_{ui}(t) = (q_i + o_{C_i})^T (p_u(t) + o_{C_u}(t) + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j + |I_{C_u}|^{-\frac{1}{2}} \sum_{j \in I_{C_u}} z_j). \tag{14}$$

Koren [21] does not make the item vector q_i time-dependent. He claims that item characteristics are inherent and do not change with time. We expect that this also applies to the item community vector, so we also leave it to be time-independent. Finally, we combine the baseline estimation, the neighborhood model and the factor model by summing their predictions.

For all the models, a squared error function is minimized to learn the parameters of the model e.g. regularization parameters, using stochastic gradient descent. A regularization factor λ is used to penalize high parameter values to avoid overfitting the training data. An implementation is included in LibRec [1] and we adapt it to our NSVD-based models [15]. We performed some validations on hold-out data and found the optimum learning rate and regularization parameters for the NSVD, TNSVD, CNSVD and TCNSVD models using RMSE error. We selected the learning rate decay strategy from LibRec and used the same combination of learning rate and regularization for parameters of each model. The best learning rates and regularization parameters for each model are given in Table 1.

Model	Learning Rate	Regularization Factor
NSVD	0.1	0.1
TNSVD	0.001	0.001
CNSVD	0.01	0.1
TCNSVD	0.0001	1

Table 1: Best learning parameters for each model (tested on hold-out data).

4 EVALUATION

Regarding experiments, we use the popular Netflix (NF) and the latest MovieLens (ML) dataset to evaluate the proposed recommendation algorithms. The basic statistics of the datasets are shown Table 2. In MovieLens both ratings and tags information are available. As for tags-based graph construction, an edge is created between any two users that have used the same tag on any item. Similarly, an edge is created between any two items that have received the same tag from any user [16]. Regarding MovieLens and Netflix graph construction based on rating information, we apply

[1] http://www.librec.net/

Table 2: Basic statistics of Netflix and MovieLens datasets.

Dataset	Users	Items	Ratings
Netflix	480,189	17,770	100,480,507
MovieLens	138,000	27,000	20,000,000

k-NN proposed by Park et al. as an approach mainly suitable for information retrieval and recommender algorithms [23]. As such, to compute the similarities between users and items from the rating information, we use similarity measures such as Pearson Correlation, Cosine Similarity and Jaccard Mean Squared Distance (JMSD) [4, 6, 17, 19, 28].

For the evaluation of our proposed recommender model, we compute both rating prediction and the item ranking quality as well as accuracy. For measuring the accuracy of rating predictions, we used the Root Mean Squared Error (RMSE). For measuring the accuracy of item rankings, we used the measures Precision at k (Prec@k), Recall at k (Rec@k) and Normalized Discounted Cumulative Gain (NDCG) [18, 27]. Finally, we use Wilcoxon Rank Sum tests to identify statistical differences of the results generated by the models [29]. Evaluation of time-aware recommender algorithms is challenging since the time ordering of the ratings needs to be considered and thus the use of cross validation approaches are not suitable. Campos et al. [5] describe in detail the issues regarding time-aware recommender systems in their 2014 UMUAI paper. Instead of k-fold cross-validation, we applied a sliding-window approach taking snapshots along the timeline [12]. Taking a snapshot at time t means using the ratings within a certain number of days before t for training and the ratings within a certain number of days after t for testing. In total we took five of these snapshots over the timeline in each dataset and report the overall means in the results section. Regarding complexity, the running times of the proposed methods are more than the baselines, in which we used a subsampled version of datasets on a compute cluster. We considered a maximum running time of five days, in which TCNSVD and CNSVD models had higher running times compared to NSVD and TNSVD models. As for future works, we plan to perform time complexity analysis of the models, and ignore individual learning parameters to find a compromise between accuracy and time complexity.

5 RESULTS

In the following section, we present the results of our offline simulations. We did preliminary experiments on the current state-of-the-art community detection algorithms regarding time complexity and number of found communities. We chose DMID [26] and Walktrap [24] as two alternatives that can identify overlapping and disjoint communities. They not only scale well on large amount of data but can also handle weighed and directed networks. To use Walktrap, a step size input parameter needs to be set that was 2 and 5 in our case. Figure 1 presents the results with respect to RMSE, Prec@10 and Rec@10 for the NSVD, TNSVD, CNSVD and TCNSVD models on the MovieLens dataset (for space reasons we omitted the Netflix results here which show similar tendencies). As shown, the RMSE values are quite close to each other for both of the two community detection algorithms investigated. However, as also

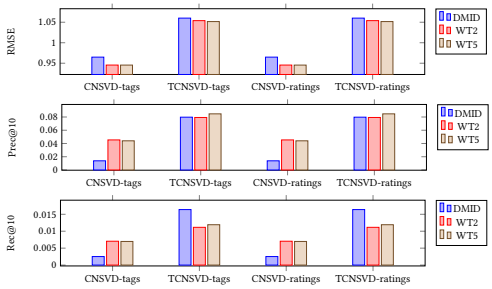

Figure 1: Comparison of model performance using DMID and Walktrap as community detection algorithms on ratings-based and tags-based graph construction on MovieLens dataset. For the Walktrap we use two stepping parameters namely, 2 = WT2 and 5 = WT5.

shown, Walktrap performs slightly better compared to the DMID algorithm. Moreover regarding RMSE, algorithms based on the TCNSVD model - using tags and ratings constructions - achieve higher values than the CNSVD model. This trend also holds for precision as well as recall. In general though, the results show that Walktrap (WT5) yields the better performance regarding RMSE, recall as well as precision. The only exception is TCNSVD recall results based on tags and ratings, in which DMID slightly outperforms Walktrap versions. To verify the overall performance of the proposed algorithms, online user studies need to be done.

Figure 2 illustrates how the models perform when using different similarity metrics using the Netflix dataset (we omit for space reasons the results of the MovieLens dataset, which though are comparable). k was set to 10 in this case. As shown, there are observable differences with respect to the measures chosen for both models. In general we can observe the following: Pearson achieves in four cases the best results for RMSE, Prec@10 and Rec@10, followed by Cosine and JMSD. This pattern is also emerging when testing with different ks and the different similarity metrics at the same time. Table 3 shows the best performing parameters for the ratings-based graph construction regarding RMSE, precision and recall and the best performing parameters for k. To figure out the practical performance of the models and similarity metrics, we need to deploy them online and study users' feedback.

Table 3: Best performing k-NN graph construction parameter values with respect to RMSE, precision and recall.

Model	RMSE	Precision@10	Recall@10
CNSVD	k = 20, Pearson	k = 10, Cosine	k = 20, Cosine
TCNSVD	k = 20, Pearson	k = 10, Pearson	k = 10, Pearson

To illustrate how the models perform compared to some baselines, a series of experiments have been performed. First, the models were compared with the baseline methods NSVD and TNSVD on the MovieLens (ML) and Netflix (NF) datasets. Thereafter, we compared them to current state-of-the-art item-ranking models such as WRMF and ItemKNN as present in the LibRec library.

Table 4 shows the first set of results in this respect. In general we can observe that the TCNSVD model achieves the best results here.

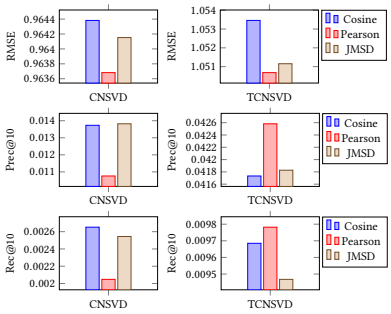

Figure 2: Comparison of model performance employing different similarity metrics for the k-NN graph construction using the Netflix dataset: Pearson correlation, cosine similarity and Jaccard mean squared distance. The parameter k was set to 10 in this case.

Table 4: Performance differences of the community-aware models to their baseline models for the MovieLens and Netflix datasets. Statistically significant differences are denoted with *p<0.05, **p<0.01, ***p<0.001 based on a Wilcoxon Ranked Sum test.

DS	Model	Graph	Baseline	ΔRMSE	ΔNDCG	ΔPrec@10	ΔRec@10
ML	TCNSVD	Ratings	TNSVD	+3.91%	+45.45 % ***	+675.32 % ***	+443.90% ***
		Tags	TNSVD	+2.50 % **	+3.23 % *	+35.78% **	-13.62% *
	CNSVD	Ratings	NSVD	+0.27%	+1.26 %	+1.07% ***	+2.63% ***
		Tags	NSVD	+0.59%	-2.50%	-20.55%	-19.60% *
NF	TCNSVD	Ratings	TNSVD	+28.14 % ***	+15.67% ***	+126.73%	+115.90% ***
	CNSVD	Ratings	NSVD	-0.01 %	+2.57 % ***	+35.33 %	+44.49 %

For instance in the MovieLens dataset, compared to its baseline (TNSVD), the method increases NDCG, Precision and Recall with 45.45 %, 675 % and 443 % in the best case relying on a ratings-based graph. Similar trends are also observed for the Netflix dataset. Here, TCNSVD can also improve NDCG, Prec@10 and Rec@10 with 15.67 %, 126.73 % and 115.90 %, compared to it's baseline method. The result "patterns" on the other hand for the CNSVD model are not that clear as RMSE and NDCG values are sometimes decreased, while Prec@10 and Rec@10 are not. This is actually an interesting behaviour which we need study further in future work.

Figure 3 provides an overview of the computed item ranking and precision metrics. Regarding the MovieLens dataset and the NDCG metric, the TCNSVD model could achieve the best results, that is statistically better than the baselines WRMF, ItemKNN, CNSVD, NSVD and TNSVD. TCNSVD surpasses the other baselines also for Prec@10 and Rec@10 with even higher differences. As for Prec@10, TCNSVD achieves 0.28359, which is higher than 0.18842 and 0.17433 as obtained by WRMF and ItemKNN. The results and the ranking of the methods are consistent with other benchmarks run on the MovieLens dataset, although our evaluation protocol is time-based [13]. As for the Netflix dataset, again TCNSVD gives us the best results with notable statistically significant differences to the other models.

6 SUMMARY & FUTURE WORK

The main findings of the paper can be summarized as follows:

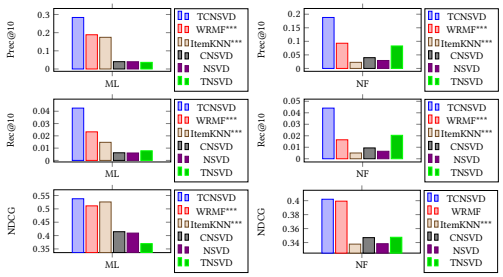

Figure 3: Performance comparison of proposed CNSVD and TCNSVD models with the baselines and other item ranking algorithms based on Prec@10, Rec@10 and NDCG using LibRec[2]. Statistically significant differences between TCNSVD and WRMF or TCNSVD and ItemKNN are denoted with **p<0.01, ***p<0.001 based on a Wilcoxon Ranked Sum Test.

- We proposed a community-aware model named CNSVD based on neighborhood and factor models of recommendation.
- Furthermore, we introduced TCNSVD as a model that considers temporal community structure and dynamics.
- We showed the effect of community detection algorithms on the recommendation performance and found that Walktrap is the better option.
- Also it was shown that Pearson correlation as the similarity metric for graph construction achieves the best performance when considering temporal dynamics of community structures in the recommendation task.
- Finally, we show that TCNSVD as a temporal and community-aware recommender model performs sign better than CNSVD and compared state-of-the-art baseline recommendation approaches on the MovieLens and Netflix datasets.

One limitation of the proposed CNSVD and TCNSVD models are their high dimensionality. As such, it is planned for future work to improve the models in such a way that less parameters need to be set, e.g. by omitting individual user and item parameters. In our experiments we selected the optimum learning rate and regularization based on RMSE. Future work needs to assess whether further improvements can be found when optimizing the models e.g. with respect to NDCG.

Although temporal dynamics of overlapping community structures are considered by TCNSVD, modelling the effect of community life cycles, i.e. birth, death, atrophy, grow, split and merge, on recommendation systems still need to be studied. Moreover, we plan to study the impact of time bins as well as speed of community changes in the proposal of a recommender system. In addition, more community detection algorithms need to be investigated with our models, to identify best performing ones. Finally, we plan to investigate the effect of explicit community structures as the current ones are based on implicit ones and already achieving remarkable results.

ACKNOWLEDGMENTS

This work is in part supported by BIT research school.

REFERENCES

[1] Backstrom, L., Huttenlocher, D., Kleinberg, J. M., and Lan, X. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06* (2006), ACM Press, pp. 44–54.

[2] Baltrunas, L., and Amatriain, X. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems* (2009).

[3] Bellogin, A., and Parapar, J. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the 6th ACM conference on recommender systems* (2012), ACM, pp. 213–216.

[4] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems 46* (2013), 109–132.

[5] Campos, P. G., Díez, F., and Cantador, I. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction 24*, 1-2 (2014), 67–119.

[6] Candillier, L., Meyer, F., and Boullé, M. Comparing state-of-the-art collaborative filtering systems. In *Proceedings of the 5th international conference on machine learning and data mining in pattern recognition* (2007), Springer, pp. 548–562.

[7] Cao, C., Ni, Q., and Zhai, Y. An improved collaborative filtering recommendation algorithm based on community detection in social networks. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (2015), ACM, pp. 1–8.

[8] Charlin, L., Ranganath, R., McInerney, J., and Blei, D. M. Dynamic poisson factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems* (New York, NY, USA, 2015), RecSys '15, ACM, pp. 155–162.

[9] Choo, E., Yu, T., Chi, M., and Sun, Y. Revealing and incorporating implicit communities to improve recommender systems. In *Proceedings of the 15th ACM conference on economics and computation* (2014), ACM, pp. 489–506.

[10] Daneshmand, S. M., Javari, A., Abtahi, S. E., and Jalili, M. A time-aware recommender system based on dependency network of items. *The Computer Journal 58*, 9 (2015), 1955–1966.

[11] Dolgikh, D., and Jelínek, I. Graph-based music recommendation approach using social network analysis and community detection method. In *Proceedings of the 16th international conference on computer systems and technologies* (2015), ACM, pp. 221–227.

[12] Eberhard, L., and Trattner, C. Recommending sellers to buyers in virtual marketplaces leveraging social information. In *Companion of the 25th international conference on World Wide Web* (2016), ACM, pp. 559–564.

[13] Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)* (2011).

[14] Gauvin, L., Panisson, A., and Cattuto, C. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *PLoS ONE 9*, 1 (2014).

[15] Guo, G., Zhang, J., Sun, Z., and Yorke-Smith, N. Librec: A java library for recommender systems. In *Posters, demos, late-breaking results and workshop proceedings of the 23rd conference on user modeling, adaptation and personalization* (2015).

[16] Helic, D., Strohmaier, M., Trattner, C., Muhr, M., and Lerman, K. Pragmatic evaluation of folksonomies. In *20th International World Wide Web Conference (WWW2011), Hyderabad, India, March 28 - April 1, ACM* (2011).

[17] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM conference on research and development in information retrieval* (1999), ACM, pp. 230–237.

[18] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. *Transactions on Information Systems 22*, 1 (2004), 5–53.

[19] Jesús Bobadilla, J., Serradilla, F., and Bernal, J. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems 23*, 6 (2010), 520–528.

[20] Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM international conference on knowledge discovery and data mining* (2008), ACM, pp. 426–434.

[21] Koren, Y. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM international conference on knowledge discovery and data mining* (2009), ACM, pp. 447–456.

[22] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (New York, NY, USA, 2005), KDD '05, ACM, pp. 177–187.

[23] Park, Y., Park, S., Lee, S.-g., and Jung, W. Greedy filtering: A scalable algorithm for k-nearest neighbor graph construction. In *Proceedings of the 19th international conference on database systems for advanced applications* (2014), Springer, pp. 327–341.

[24] Pons, P., and Latapy, M. Computing communities in large networks using random walks. In *Proceedings of the 20th international symposium on computer and information sciences* (2005), Springer, pp. 284–293.

[25] Sahebi, S., and Cohen, W. W. Community-based recommendations: A solution to the cold start problem. In *Proceedings of the 3rd workshop on recommender systems and the social Web* (2011), ACM, pp. 40–44.

[26] Shahriari, M., Krott, S., and Klamma, R. Disassortative degree mixing and information diffusion for overlapping community detection in social networks (DMID). In *Proceedings of the 24th international conference on World Wide Web* (2015), ACM, pp. 1369–1374.

[27] Shani, G., and Gunawardana, A. Evaluating recommendation systems. In *Recommender Systems Handbook*. Springer, 2011, pp. 257–297.

[28] Su, X., and Khoshgoftaar, T. M. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* (2009), 421425:1–421425:19.

[29] Trattner, C., Kowald, D., Seitlinger, P., Ley, T., and Kopeinik, S. Modeling activation processes in human memory to predict the use of tags in social bookmarking systems. *The Journal of Web Science 2*, 1 (2016), 1–16.

[30] Yanxiang, L., Deke, G., Fei, C., and Honghui, C. User-based clustering with top-n recommendation on cold-start problem. In *Proceedings of the 3rd international conference on intelligent system design and engineering applications* (2013), IEEE, pp. 1585–1589.