Towards Comprehensive Recommender Systems: Time-Aware Unified Recommendations Based on Listwise Ranking of Implicit Cross-Network Data

Dilruk Perera, Roger Zimmermann
School of Computing, National University of Singapore
{dilruk, rogerz}@comp.nus.edu.sg

Abstract

The abundance of information in web applications make recommendation essential for users as well as applications. Despite the effectiveness of existing recommender systems, we find two major limitations that reduce their overall performance: (1) inability to provide timely recommendations for both new and existing users by considering the dynamic nature of user preferences, and (2) not fully optimized for the ranking task when using implicit feedback. Therefore, we propose a novel deep learning based unified cross-network solution to mitigate cold-start and data sparsity issues and provide timely recommendations for new and existing users. Furthermore, we consider the ranking problem under implicit feedback as a classification task, and propose a generic personalized listwise optimization criterion for implicit data to effectively rank a list of items. We illustrate our cross-network model using Twitter auxiliary information for recommendations on YouTube target network. Extensive comparisons against multiple time aware and cross-network baselines show that the proposed solution is superior in terms of accuracy, novelty and diversity. Furthermore, experiments conducted on the popular MovieLens dataset suggest that the proposed listwise ranking method outperforms existing state-of-the-art ranking techniques.

1 Introduction

Recommender systems have been effectively used in various web applications (e.g., search engines, e-commerce and social networks) to mitigate the notorious information overload problem. In addition to the user benefits, where interesting information are automatically recommended to users, recommender systems help web applications to significantly increase their revenue. For example, 35% of Amazon purchases and 75% of Netflix movie views are initiated by their internal recommender systems. Therefore, over the years, constant efforts have been made to improve the quality of recommendations in terms of accuracy, novelty and diversity (Perera and Zimmermann 2018; McLaughlin and Herlocker 2004). However, despite their success, existing systems face two main limitations that continue to hinder their performances.

(1) **Inability to provide timely recommendations for new and existing users:** Due to the high item-to-user ratio in web applications, existing user interactions are highly sparse, and leads to low recommender performance (*data sparsity problem*). It is also infeasible to determine new user preferences due to the absence of their interactions (*cold-start problem*). Moreover, user preferences toward items are also subject to constant change over time (Campos, Dıez, and Cantador 2012; Koren 2009). Therefore, recommender systems should capture such changes and model up-to-date user preferences for effective recommendations for both new and existing users.

(2) **Not fully optimized to use implicit feedback data for listwise ranking:** Unlike interactions with explicit feedback (1-5 ratings), most user interactions are implicit in nature (e.g., watching a video or clicking on a link). Therefore, recent years have seen a shift in recommender systems from using explicit ratings to widely and cheaply available implicit ratings (Kelly and Teevan 2003). In practice, recommender systems output a ranked list of items, based on user preferences. However, existing implicit feedback based systems, for example, Matrix Factorization (MF) (Koren, Bell, and Volinsky 2009), Factorization Machines (FM) (Rendle 2010) and vast majority of deep learning models (He and Chua 2017) are not fully optimized to rank a list of items. They are commonly optimized for predicting user preferences for a single item (*pointwise*) or, ranking preferences for a pair of items (*pairwise*). Therefore, the achievable recommender quality is limited.

We propose a time aware unified cross-network solution with a novel personalized listwise loss function to address the above limitations and create a consolidated recommender solution. User engagements on multiple social networks showcase their preferences from various perspectives. Therefore, such preferences across networks can be used to mitigate cold start and data sparsity issues, while improving recommender novelty and diversity. Therefore, for new users, at the time of registration, their preferences are obtained from other source networks. For existing users, more comprehensive preferences are obtained by integrating their preferences with other source networks.

To capture the dynamic nature of user preferences, we

modeled preferences under three temporal levels, namely short, long and long short term preferences. Short term preferences reflect most recent preferences and are highly likely to drive immediate future interactions. Long term preferences are practised over an extended period and reflect general preferences. Unlike short term preferences, long term preferences are less affected by the current context and recent trends. However short term preferences are highly sensitive to preference outliers. For example, a user may be drawn to watch athletics videos during the hype of the Olympics season. Once the Olympics is over, he may no longer be interested in watching such videos. Therefore, we introduced a novel long short term preference component to effectively remove such outliers. The main intuition is to exploit historical user preferences that are similar to the current preferences, and compute a new (long short term) preference component based on the preference similarities and frequencies, which is less affected by the current outliers and highly relevant for immediate user interactions.

We summarize our main contributions as follows:

- We proposed a novel deep learning based time aware unified cross-network model that captures user preferences in three temporal levels for effective recommendations.
- We consider the ranking problem under implicit feedback as a classification task, and proposed a novel, generic listwise optimization criterion.
- We conducted extensive experiments to demonstrate that the proposed model and optimization criterion consistently outperform the state-of-the-art baselines.

2 Literature Review

User preferences from multiple networks have increased the robustness of recommendations against cold start and data sparsity issues (Mehta et al. 2005; Yan, Sang, and Xu 2015). For example, YouTube recommender accuracy was increased using information from Google+ (Sang et al. 2015), Twitter (Wanave and Takale 2016), and Amazon recommender accuracy was increased with information from Cheetah Mobile (Hu, Zhang, and Yang 2018). However, most cross-network solutions are non-unified and solve either cold-start or data sparsity issues.

Recently, a handful of efforts have been made to utilize auxiliary information from source networks to develop unified recommender solutions (Yan, Sang, and Xu 2015; Yan et al. 2016). However, these solutions do not model the dynamic nature of user preferences and recommender performance degrades over time unless the models are retrained frequently. To the best of our knowledge, the first time aware unified solution was proposed in (Perera and Zimmermann 2017), where the authors used a decaying function to prioritize most recent preferences when conducting recommendations. However, the model is based on a linear MF model and optimized using a pointwise criterion.

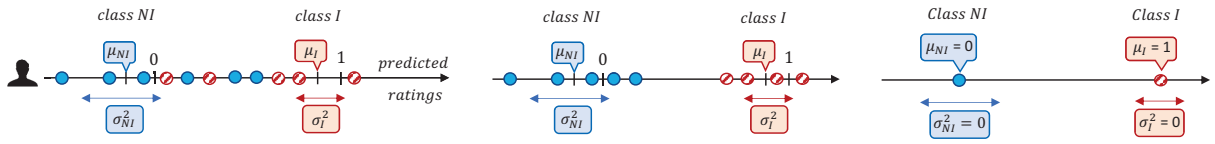
Widely used pointwise optimization criteria are less effective for implicit feedback based approaches due to the absence of negative feedback. As a solution, Bayesian Personalized Ranking (BPR) was proposed to maximize the difference between predicted ratings for interacted and non-interacted items (Rendle et al. 2009). However, BPR and its variants (Yu et al. 2018; Loni et al. 2016), were not fully optimized to rank a list of items. Similar work on learning to rank with non-collaborative models have also been proposed (e.g., modeling distributions on permutations (Kondor, Howard, and Jebara 2007; Huang, Guestrin, and Guibas 2008). Nevertheless, these solutions are not personalized and are not well suited for the recommendation task. Listwise optimization approaches based on explicit feedback showed superiority over pointwise and pairwise approaches for recommending a ranked list of items (Cao et al. 2007).

3 Model Preliminaries

Intuition for the Listwise Loss: Given a list of items with corresponding implicit feedback given by a user, the optimization criterion aims to classify item feedback into 2 classes (interacted and non-interacted). The goal is to maintain high inter-class distances and low intra-class distances. Considering ground truth values for interacted and non-interacted feedback as 1 and 0, the optimization criterion should lead the mean values of the corresponding classes (μ_I and μ_{NI}) to 1 and 0, and the variance values of both classes (σ_I^2 and σ_{NI}^2) to 0. Accordingly, a non optimized list of items will not have clearly separated classes, a partially optimized list will have a clear separation, and an ideally optimized list will have all values of each class merged into a single point (see Figure 1). Note that, the proposed personalized loss function is generic and can be used with popular collaborative filtering approaches (e.g. MF, FM and kNN).

Pairwise Loss verses Listwise Loss: Unlike explicit feedback, implicit feedback lacks the notion of preference levels due to its binary nature. Therefore, popular pointwise loss functions that are optimized to predict preference levels are ineffective for implicit feedback. Pairwise loss is the widely adopted optimization criterion for learning with implicit feedback. Although recommendation is widely considered to be a ranking task, we argue that under implicit feedback, it is a binary classification task. Considering the list of items for a user, the aim is to classify the items into 2 classes (interacted and non-interacted). The pairwise approach is not suitable for this task, since it is only optimized to rank a pair of items, and the ranking for the item list is indirectly obtained through transitivity ($i_1 > i_2$, and $i_2 > i_3$ means $i_1 > i_3$). In the proposed listwise loss, the optimization is more realistic in nature as it directly optimizes for the entire list in each training instance.

It is also infeasible to run pairwise optimizations for all item pairs, since it requires ${}^nC_2$ (n is the no. of items) runs per user, per training epoch. Hence, pairwise optimization is executed on a randomly selected subset of item pairs for each user, at each instance by assuming that multiple executions would generalize to rank the entire item list. Thus, effectiveness largely depends on the quality of the randomly selected item pairs, and no of pairs. Also, since ranking is obtained through transitivity, item predictions need to be computed multiple times to obtain item list rankings. For example, to obtain $i_1 > i_3$, from $i_1 > i_2$ and $i_2 > i_3$, prediction for i_2 need to be computed in two runs. This is highly



(a) Non optimized list (b) Partially optimized list (c) Ideally optimized list

Figure 1: Stages of the listwise loss optimization process.

inefficient considering the number of items in applications. In contrast, the listwise approach only computes the prediction for an item once and has a similar complexity to the inferior pointwise approach. Furthermore, since pairwise approach does not compute ratings for the entire list in a single run, during validation and testing, predictions for each user-item pair need to be computed in different runs and ranked in a separate process. In the listwise approach, the model is already trained to provide a classified list for testing.

Problem Formulation: We denoted each existing user on the target network at a time interval t as $u^t_e \in U = [Sr^t_e, Tg^t_e]$, using his source and target network interaction histories (Sr^t_e and Tg^t_e) over $T = \{1, ..., t\}$ time intervals. Similarly, each new user is denoted as $u^t_n \in U = [Sr^t_n]$, using only source network interactions (Sr^t_n), since new user interactions are unavailable on the target network. Given the interaction histories, at the end of each interval t, the goal is to determine their preferences and predict future interactions at $t + 1$, on the target network.

4 Methodology

The proposed model conducts recommendations for new and existing users over four stages (see Figure 2). First, historical interactions are transferred and integrated in the cross-network topical layer. Second, three levels of user preferences are extracted from the computed topical representations on both source and target networks. Third, extracted preferences are integrated to obtain overall preferences of new and existing users. Fourth, the overall preferences are used to conduct unified recommendations.

4.1 Cross-Network Topical Layer

User interactions on different networks are often multimodal and heterogeneous in nature (e.g., tweets and liked YouTube videos). Since comparison of heterogeneous interactions across networks is challenging, we used topic modeling to transform user interactions onto a homogeneous topical space (see Figure 2, stage 1). We assumed that each user interaction is associated with multiple topics and extracted related topics from textual metadata (i.e., tweet contents, video titles and descriptions). We considered each tweet and interacted YouTube video as a *document* and extracted topics using Twitter-Latent Dirichlet Allocation (Twitter-LDA) (Zhao et al. 2011), which is highly effective against short and noisy textual contents.

Consequently, each interaction is represented as a topical distribution over all possible topics. Source network interactions of new users within a given time interval t are represented as $\boldsymbol{x^t_{Sr}} = \{f^{t,1}_{Sr}, \ldots, f^{t,K^t}_{Sr}\} \in \mathbb{R}^{K^t}$, the summation over all topical distributions during t, where K^t is the number of topics. Each element $f^{t,k^t}_{Sr} \in \boldsymbol{x^t_{Sr}}$ in the resulting topical distribution is a topical frequency representing user preference level towards the corresponding topic. Therefore, topical distributions are a good representation of user preferences. Similarly, existing users within time interval t are represented as $\{\boldsymbol{x^t_{Sr}}; \boldsymbol{x^t_{Tg}}\} \in \mathbb{R}^{2K^t}$, using source and target network topical distributions. Thus, new and existing users are represented using their interaction histories over T intervals as $u_n = \{\boldsymbol{x^1_{Sr}}, \ldots, \boldsymbol{x^t_{Sr}}\} \in \mathbb{R}^{T \times K^t}$ and $u_e = \{\boldsymbol{x^1_{Tg}}, \ldots, \boldsymbol{x^t_{Tg}}; \boldsymbol{x^1_{Sr}}, \ldots, \boldsymbol{x^t_{Sr}}\} \in \mathbb{R}^{T \times 2K^t}$, which form the inputs to the model.

4.2 User Preference Extraction

We extracted user preferences as short, long and long short term preferences (see Figure 2, stage 2). This extraction process is further illustrated in Figure 3.

Embedding Layer: Given a source network topical distribution $\boldsymbol{x^t_{Sr}} = \{f^{t,1}_{Sr}, \ldots, f^{t,K^t}_{Sr}\} \in \mathbb{R}^{K^t}$, the embedding layer projects each input topic to a dense vector representation and computes its corresponding embeddings as $E^t_{Sr} = \{\boldsymbol{e^{t,1}_{Sr}}, \ldots, \boldsymbol{e^{t,K^t}_{Sr}}\} \in \mathbb{R}^{K^t \times k}$, where $\boldsymbol{e^{t,c}_{Sr}} = f^{t,c}_{Sr} \cdot \boldsymbol{v^c_{Sr}} \in \mathbb{R}^k$ is the embedding for topic c, $f^{t,c}_{Sr}$ is the topical frequency, $\boldsymbol{v^c_{Sr}}$ is the learnt global vector representation of topic c, and k is the dimensionality of the embedding space. We only learnt embeddings for non-zero topical frequencies since $f^{t,c}_{Sr} = 0$ leads to $\boldsymbol{e^{t,c}_{Sr}} = 0$. Similarly, a separate set of vector representations $\boldsymbol{v^c_{Tg}}$ are learnt for topics on the target network. Therefore, for n and m non-zero frequencies on source and target networks, resulting embedding vectors are $E^t_{Sr} \in \mathbb{R}^{k \times n}$ and $E^t_{Tg} \in \mathbb{R}^{k \times m}$. The embeddings are used to compute short, long and long short term user preferences. The intuition for learning separate sets of topical vector representations for each network is to distinctly capture the network-level topical properties. For example, entertainment related topics are more prominent on YouTube than Twitter (Perera and Zimmermann 2017). Therefore, the same topic will be differently represented on each network.

Short Term Preferences: We extracted short term preferences from interactions within the most recently completed time interval as it is the closest representation of the current context. At the end of time interval t, short term preferences of new users are computed using source network em-

beddings within the time interval $\left(E^t_{Sr} \in \mathbb{R}^{k \times n}\right)$, and short term preferences of existing users are computed using both source and target network embeddings $\left(E^t_{Sr} \in \mathbb{R}^{k \times n}\right.$ and $\left.E^t_{Tg} \in \mathbb{R}^{k \times m}\right)$ as follows:

$$\boldsymbol{sp^t_n} = \{\boldsymbol{s^t_{Sr}}\} = \sum_{c \in C_n} \boldsymbol{e^{t,c}_{Sr}} \qquad \in \mathbb{R}^k \qquad (1)$$

$$\boldsymbol{sp^t_e} = \{\boldsymbol{s^t_{Sr}}; \boldsymbol{s^t_{Tg}}\}, = \sum_{c \in C_n} \boldsymbol{e^{t,c}_{Sr}}\ ; \sum_{c \in C_m} \boldsymbol{e^{t,c}_{Tg}} \qquad \in \mathbb{R}^{2k} \qquad (2)$$

where $\boldsymbol{sp^t_n}$ and $\boldsymbol{sp^t_e}$ are short term preferences of new and existing users, C_n and C_m are sets of indices of the non-zero topical frequencies in source and target networks.

Long Term Preferences: Long term preferences are modeled using the collection of short term preferences over time. Hence, at the end of time interval t, long term preferences for new and existing users are computed as follows:

$$\boldsymbol{lp^t_n} = \sum_{t=1}^{t-1} \boldsymbol{s^t_{Sr}} \qquad \in \mathbb{R}^k \qquad (3)$$

$$\boldsymbol{lp^t_e} = \sum_{t=1}^{t-1} \boldsymbol{s^t_{Sr}}\ ; \sum_{t=1}^{t-1} \boldsymbol{s^t_{Tg}} \qquad \in \mathbb{R}^{2k} \qquad (4)$$

where $\boldsymbol{lp^t_n}$ and $\boldsymbol{lp^t_e}$ are long term preferences of new and existing users.

Long Short Term Preferences: We used a neural network based attention mechanism to compute long short term preferences as a weighted sum of long term preferences. Specifically, the neural network computes attention scores (i.e., similarity based on relevancy and frequency) between past and current preferences. Thereby, only the relevant past preferences are used to compute the long short term preferences which reduces the outlier effects on short term preferences. Thus, for each user, given a pair of current and past preference embeddings on source ($\boldsymbol{s^t_{Sr}}$ and $\boldsymbol{s^{t'}_{Sr}}$) or target networks ($\boldsymbol{s^t_{Tg}}$ and $\boldsymbol{s^{t'}_{Tg}}$), the attention scores are computed as follows:

$$\alpha^{t'}_{Sr} = \phi_{Sr}(\boldsymbol{s^t_{Sr}}, \boldsymbol{s^{t'}_{Sr}}) \qquad (5)$$

$$\alpha^{t'}_{Tg} = \phi_{Tg}(\boldsymbol{s^t_{Tg}}, \boldsymbol{s^{t'}_{Tg}}) \qquad (6)$$

where ϕ_{Sr} and ϕ_{Tg} are neural attention functions for source and target networks, $\alpha^{t'}_{Sr} \in \mathbb{R}$ and $\alpha^{t'}_{Tg} \in \mathbb{R}$ are attention scores between past (t') and current (t) preferences. Once attention scores are computed for all past intervals, long term preferences are weighted as follows:

$$\boldsymbol{lsp^t_n} = \sum_{t'=1}^{t-1} \alpha^{t'}_{Sr} \cdot \boldsymbol{s^{t'}_{Sr}} \in \mathbb{R}^k \qquad (7)$$

$$\boldsymbol{lsp^t_e} = \sum_{t'=1}^{t-1} \alpha^{t'}_{Sr} \cdot \boldsymbol{s^{t'}_{Sr}} \sum_{t'=1}^{t-1} \alpha^{t'}_{Tg} \cdot \boldsymbol{s^{t'}_{Tg}} \in \mathbb{R}^{2k} \qquad (8)$$

where $\boldsymbol{lsp^t_n}$ and $\boldsymbol{lsp^t_e}$ are long short term preferences of new and existing users.

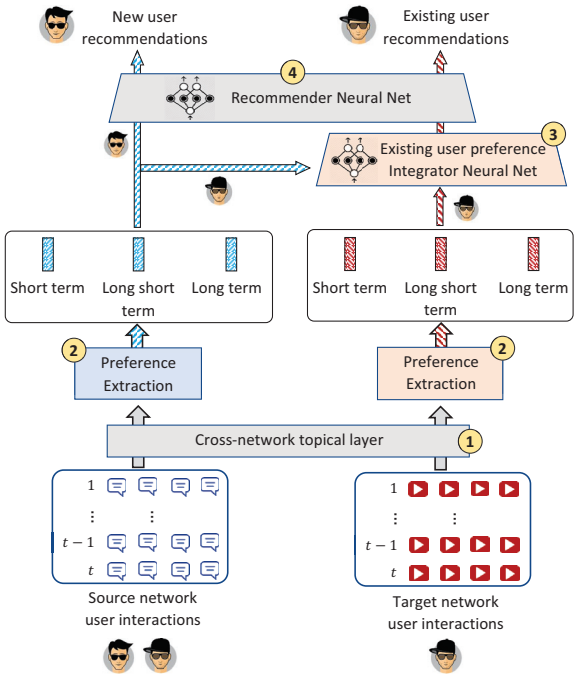


Figure 2: High-level overview of the recommendation process for new and existing users.

4.3 User Preference Integration

We integrated short, long and long short term preferences of new and existing users to obtain a final preference representation that reflects their near future preferences.

New User Preferences: The temporal preferences for new users are captured from the same network space. Therefore, the final new user preference representation at the end of time interval t ($\boldsymbol{p^t_n}$) is obtained using a direct summation over the computed preference components as follows:

$$\boldsymbol{p^t_n} = \boldsymbol{sp^t_n} + \boldsymbol{lp^t_n} + \boldsymbol{lsp^t_n} \qquad \in \mathbb{R}^k \qquad (9)$$

Existing User Preferences: Unlike for new users, the temporal preferences for existing users are captured on heterogeneous network spaces. Therefore, we introduced a neural integration function to effectively integrate these user preferences across networks (see Figure 2, stage 3). Hence, the final preference representation for existing users at the end of time interval t ($\boldsymbol{p^t_e}$) is computed as follows:

$$\boldsymbol{p^t_e} = \Phi_E(\boldsymbol{v_e}, \boldsymbol{sp^t_e}, \boldsymbol{lp^t_e}, \boldsymbol{lsp^t_e}) \qquad \in \mathbb{R}^k \qquad (10)$$

where Φ_E is the neural integration function. We also introduced an additional latent embedding $\boldsymbol{v_e} \in \mathbb{R}^{D'}$ to represent unique characteristics of each existing user (e.g., rating styles) to further support personalized recommendations. For example, some users may normally give higher ratings when they enjoy items, others may have high expectations and tend to give lower ratings on average. Hence, these highly personalized characteristics can be latently encoded in their embeddings.

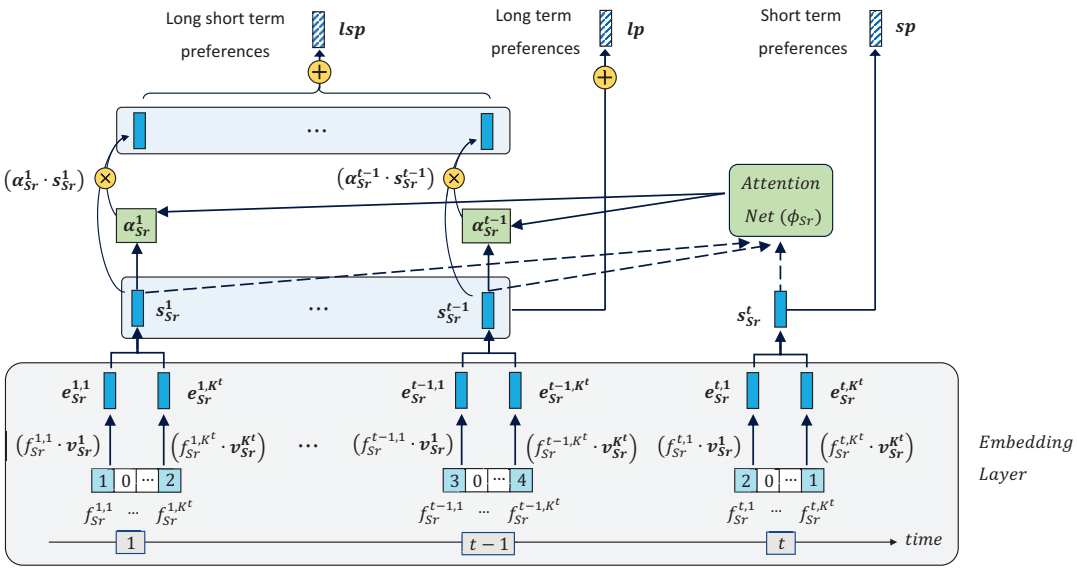


Figure 3: Overview of the user preference extraction process on the source network, which computes short, long and long short term user preferences. Note that, this is analogous to the extraction process on the target network.

4.4 Recommendation

Many early recommender solutions modeled linear relationships between user-item interactions (e.g., MF and FM). However, recent neural network based recommender solutions showcased that neural functions better model complex user-item interactions (He et al. 2017). Therefore, we used a feed forward neural network (see Figure 2, stage 4) to model user-item pairs and obtain rating predictions as follows:

$$\hat{r}_{ni}^{t+1} = \Phi_R(\boldsymbol{p_n^t}, \boldsymbol{v_i}) \in \mathbb{R} \tag{11}$$

$$\hat{r}_{ei}^{t+1} = \Phi_R(\boldsymbol{p_e^t}, \boldsymbol{v_i}) \in \mathbb{R} \tag{12}$$

where $\hat{r}_{n,i}^{t+1}$ and $\hat{r}_{e,i}^{t+1}$ are predicted ratings for new and existing users for an item i during $t+1$, $\boldsymbol{v_i} \in \mathbb{R}^D$ is the latent item embedding which represents the unique characteristics of each item i, and Φ_R is the neural recommender function.

4.5 Optimization

The optimization function contains a novel listwise loss and an attention loss component.

Listwise Loss Component: The model is trained at the end of each time interval t, using listwise training instances. These instances contain actual (ground truth) interacted and non-interacted items at the next interval $t+1$. Hence, we define each listwise training instance for a new user u_n^t at t as a triplet $H_n^t = \{(u_n^t, I_{n+}^{t+1}, I_{n-}^{t+1}) | u_n^t \in U_{N+}^{t+1} \wedge I_{n-}^{t+1} \in I \setminus I_{n+}^{t+1}\}$, where U_{N+}^{t+1} is the set of new users with at least one interaction at $t+1$, I denotes the set of all items, and I_{n+}^{t+1} is the set of interacted items. Analogously, each listwise training instance for an existing user u_e^t is denoted as $H_e^t = \{(u_e^t, I_{e+}^{t+1}, I_{e-}^{t+1}) | u_e^t \in U_{E+}^{t+1} \wedge I_{e-}^{t+1} \in I \setminus I_{e+}^{t+1}\}$.

According to the classification approach, during the training process, the recommender should assign rating values of 1 for interacted items (I_{n+}^{t+1} and I_{e+}^{t+1}) and 0 for non-interacted items (I_{n-}^{t+1} and I_{e-}^{t+1}). Hence, the listwise loss functions to be minimized for new ($L_{lw,n}$) and existing ($L_{lw,e}$) user predictions are defined as follows:

$$L_{lw,n} = \sum_{u_n^t} \sum_{i \in I_{n+}^{t+1}} \sum_{j \in I_{n-}^{t+1}} \left(1 - \mu_{ni}\right)^2 + \mu_{nj}^2 + \sigma_{ni}^2 + \sigma_{nj}^2$$

$$L_{lw,e} = \sum_{u_e^t} \sum_{i \in I_{e+}^{t+1}} \sum_{j \in I_{e-}^{t+1}} \left(1 - \mu_{ei}\right)^2 + \mu_{ej}^2 + \sigma_{ei}^2 + \sigma_{ej}^2$$

where μ_{ni} and μ_{nj} (μ_{ei} and μ_{ej}) are mean predicted ratings of interacted and non-interacted items for new (existing) users, and σ_{ni}^2 and σ_{nj}^2 (σ_{ei}^2 and σ_{ej}^2) are corresponding variance values for new (existing) users.

Attention Loss Component: The attention neural functions for both source and target networks compute attention scores (similarity scores of [0,1]) between pairs of past and current preferences. Therefore, we introduced attention loss components to better train the attention neural networks (ϕ_{Sr} and ϕ_{Tg}). During each training instance, the computed current preference of a user on a given network is duplicated to create an extra pair of inputs to the attention function of the corresponding network. Thus, a well trained attention function should output 1 to indicate the perfect match between the duplicated preferences. Thereby, the attention loss components to be minimized are defined as follows:

$$\begin{aligned} L_{at,n} &= \left(1 - \phi_{Sr}(\boldsymbol{s_{Sr}^t}, \boldsymbol{s_{Sr}^t})\right)^2 \\ L_{at,e} &= \left(1 - \phi_{Sr}(\boldsymbol{s_{Sr}^t}, \boldsymbol{s_{Sr}^t})\right)^2 + \\ &\quad \left(1 - \phi_{Tg}(\boldsymbol{s_{Tg}^t}, \boldsymbol{s_{Tg}^t})\right)^2 \end{aligned} \tag{13}$$

where $L_{at,n}$ and $L_{at,e}$ are attention loss components for new and existing users. Note that, the duplicated inputs are not

used for recommendations since they are auxiliary inputs used to train the attention networks and are not valid inputs.

Total Loss: The effectiveness of the proposed model depends on two main tasks: (1) accurate modeling of user preferences, and (2) effective optimization of the classification task. Since the attention loss and listwise loss components represent these two tasks, the final loss functions to be optimized are formed by combining the two loss components. Accordingly, the final loss functions for new and existing users (L_n and L_e) are as follows:

$$L_n = L_{lw,n} + L_{at,n} \tag{14}$$

$$L_e = L_{lw,e} + L_{at,e} \tag{15}$$

Note that, since both tasks are equally important for the overall effectiveness, we used a simple addition to combine the two loss components. We could also exploit other methods to model the contribution from each task (e.g., learning a hyperparameter weight for the attention loss or learning a complex neural function to integrate the two components). However, the use of addition reduces the number of hyperparameters and manages the overall complexity of the model. The empirical results also proved it to be highly effective.

5 Experiments

5.1 Datasets

We conducted experiments on two datasets, CrossNet and MovieLens[1] to evaluate the proposed cross-network model and listwise optimization criterion, respectively.

CrossNet: We extracted overlapped users on Twitter and YouTube from two public datasets (Yan, Sang, and Xu 2014; Lim et al. 2015) and scraped timestamps of interactions from 1st March 2015 to 29th February 2017. We used tweets as source network interactions and YouTube videos either liked or added to playlists as target network interactions and extracted their associated textual contents (e.g., tweet contents, video titles and descriptions). In line with common practices, we filtered out users with less than 10 interactions on Twitter and YouTube. The final dataset contained 2372 users, 12,782 YouTube videos, and the sparsity of the user-video matrix was 99.62%.

MovieLens: We used a popular version of the movie ratings dataset with 1 million ratings from 6000 users for 4000 movies. Similar to other studies based on implicit feedback data (Koren 2008; He et al. 2017), we transformed explicit ratings to implicit ratings where each user-item pair is 1 or 0 indicating an interaction or non-interaction.

Table 1: Experimental settings.

Exp.	Granularity	Training set (%)	Testing set (%)
1	Biweekly	First 20 (83%)	Last 2 (17%)
2		First 22 (92%)	Last 2 (8%)
3	Monthly	First 10 (83%)	Last 2 (17%)
4		First 11 (92%)	Last 1 (8%)

[1]https://grouplens.org/datasets/movielens/1m/

5.2 Experimental Setup

Recommender systems are often evaluated on training and testing datasets with temporal overlaps [9]. This leads to biases in predictions as it does not reflect a realistic recommender environment. Therefore, at each time interval, we trained the proposed model only on interactions in past time intervals to predict interactions in future time intervals. We sorted all users on CrossNet based on their total number of YouTube interactions, and selected the first and second half of users as new and existing users, respectively. We ignored all YouTube interactions of new users, and designed four main experiments with two temporal granularities and train/test ratios (see Table 1). Note that, training for both new and existing users is conducted simultaneously. Furthermore, we randomly selected 10% of interactions and 10% of non-interactions from MoviLens as the test set to evaluate the proposed listwise optimization criterion.

We formulated video recommendation as a Top-N recommender task and predicted a ranked set of 10 videos that the user is most likely to interact with, at the next time interval. We calculated Hit Ratio (HR) (He et al. 2015) to evaluate the accuracy of the proposed model. Similar to other ranking approaches (Rendle et al. 2009), we calculated Area Under the ROC Curve (AUC) to evaluate the optimization criterion. Both HR and AUC metrics are calculated for each participating user and results are averaged across users.

5.3 Baselines

We used the following time aware, single network and cross-network based baselines to evaluate the proposed model.

- **TimePop**: Recommends the most popular N items in the most recent time interval to all users.
- **TBKNN** (Campos et al. 2010): Time-Biased KNN computes K neighbors for each user and recommends their recent interactions to the target user. Similar to the original experiments, we used multiple K values from 4 to 50 and averaged the results for comparisons.
- **Time-LSTM** (Zhu et al. 2017): A single network-based LSTM model that computes short and long term user preferences based on timestamps of user interactions. The authors proposed three versions, and we selected the best performing model (TimeLSTM3) for comparisons.
- **Unified** (Yan, Sang, and Xu 2015): A non-temporal cross-network model, which uses Twitter auxiliary information for new and existing user recommendations on YouTube.
- **TDCN** (Perera and Zimmermann 2017): Time Dependent Cross-Network is an MF based model, which also uses Twitter timestamped interactions as source information to conduct recommendations on YouTube.

Furthermore, we used several popular baselines to evaluate the proposed listwise loss criterion.

- **Pop**: Recommends the most popular set of items to all users.
- **MF** (Koren, Bell, and Volinsky 2009): Learns low rank embeddings for users and items and conducts user-item rating predictions based on their inner product. MF is used

for implicit and explicit rating predictions. We employed the widely used pointwise loss optimization for training.

- **BPR-MF** (Rendle et al. 2009): Uses a pairwise loss function for optimization. Each training instance is a triplet (u, i, j), where user u has interacted (not interacted) with item i (j). The optimization maximizes the difference between predictions $(\hat{r}_{ui} - \hat{r}_{uj})$.

5.4 Model Parameters

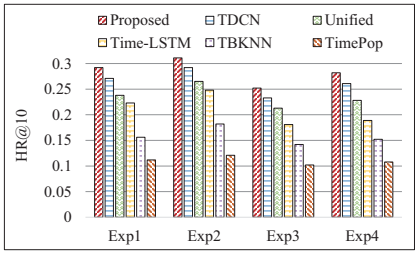
Adaptive Moment Estimation (Adam) (Kingma and Ba 2014) is an extension to the popular stochastic gradient descent algorithm, and is widely used for optimizations in various deep learning applications. We used Adam for training since it adaptively updates the learning rate. All neural network architectures used in the model were restricted to only one hidden layer, to manage the overall complexity and parameters. Given the size of the output layer as L, the size of the preceding hidden layer was set to $2 \times L$ to minimize the number of hyperparameters. All neural layers were regularized using the dropout technique, and dropout was experimentally set to 0.3 during training to prevent overfitting. We set the number of topics, K^t to 64 using a standard grid search algorithm when using Twitter-LDA to project user interactions to a cross-network topical space. The model performance was not highly sensitive to small changes in $K^t (\approx \pm 5)$.

6 Discussion

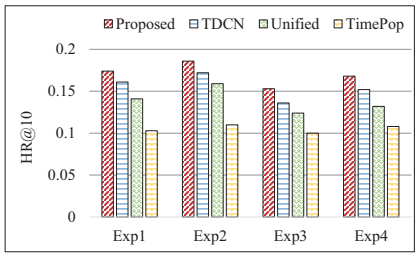
6.1 Model Evaluation

Prediction Accuracy: We plot HR@10 under the different experimental settings for existing and new users (see Figures 4a and 4b). In general, HR is higher when models are trained in biweekly intervals (Exp. 1 and Exp. 2), compared to monthly intervals (Exp. 3 and Exp. 4). This is intuitive because smaller time intervals capture more recent preferences and finer level preference dynamics over time. Larger training sets (Exp 2. and Exp 4.) also obtained slightly higher accuracy, perhaps due to better trained models. Compared to single-network based models (TimePop, TBKNN and Time-LSTM), cross-network models (Unified, TDCN and Proposed) show higher accuracy, across all experimental settings, since auxiliary information mitigates data sparsity issues. Furthermore, the accuracy improvements of TDCN over the Unified approach shows the benefits of incorporating temporal information. The Proposed model consistently outperforms all baselines in all experimental settings. Note that, only cross-network and TimePop models are able to provide recommendations for new users. The inability to utilize auxiliary information makes single-network baselines nonfunctional for new user recommendations. In general, new user recommender accuracy is lower than existing user accuracy, since new user recommendations only rely on source network information.

Effects of Three Level Temporal Preferences: To evaluate the effectiveness of short, long and long short term preferences, we removed each component individually and



(a) Existing user



(b) New user

Figure 4: Recommendation accuracy comparisons against baselines.

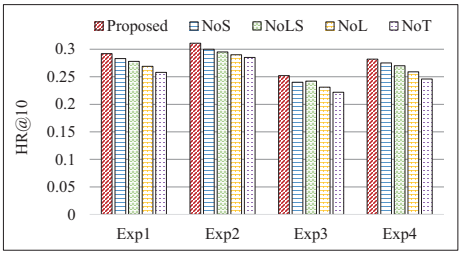
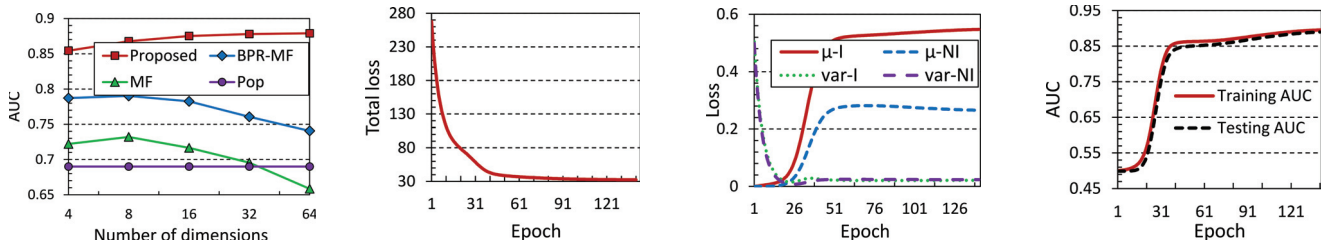


Figure 5: Effects of the proposed short, long and long short term preference components.

created several variations of the model. The resulting variations NoS, NoLS, NoL and NoT represent baselines *without* short, long short, long and all three components, respectively (see Figure 5). Apart from NoT which does not model user preference dynamics, the highest performance drop is attained when the long term preferences component (NoL) is removed. This is intuitive since long term preferences tend to contain the majority of users preferences. Compared to short term preferences, the proposed long short term preferences also have a higher impact on accuracy, which illustrates the effectiveness of the proposed long short term preference component.

Novelty and Diversity: Accuracy alone is insufficient to comprehensively evaluate recommender quality. For example, recommending a similar set of interesting items to a user would result in higher accuracy, but he would lose interest in recommendations over time. Therefore, we also measured novelty (Zhang 2013) and diversity (Avazpour et al. 2014) of recommended items. We observed that cross-network solutions outperform single network solutions as they exploit diverse user preferences from source networks. The proposed



(a) AUC against baseline loss functions.

(b) Convergence of listwise loss when optimizing the standard MF model.

(c) Mean and variance for interacted and non-interacted classes.

(d) Training and testing AUC during the convergence.

Figure 6: Loss function evaluation

model outperforms the closest TDCN approach by 8.2% and 9.1% in novelty and diversity. Further experiments show that compared to long term preferences, the use of short and long short term preferences improved novelty and diversity measures.

6.2 Loss Function Evaluation

The proposed listwise loss criterion is generic and can be used with multiple recommendation approaches. However, we optimized MF using the proposed loss function for direct comparisons against the MF based baselines as follows:

$$\hat{r}_{ui} = \sum_{f=1}^{k} \boldsymbol{w_{uf}} \cdot \boldsymbol{h_{if}} \in \mathbb{R}$$

$$L_{lw} = \sum_{u} \sum_{i \in I^+} \sum_{j \in I \setminus I^+} \left(1 - \mu_{ui}\right)^2 + \mu_{uj}^2 + \sigma_{ui}^2 + \sigma_{uj}^2$$

where $\hat{r}_{ui}$ is the rating prediction function in MF for a user-item pair, computed by taking the inner product between d-dimensional user and item latent vectors ($\boldsymbol{w_{uf}} \in \mathbb{R}^d$ and $\boldsymbol{h_{if}} \in \mathbb{R}^d$). Note that, the same function is used across all MF based baselines to evaluate the loss functions. The only difference is that we used the proposed L_{lw} loss function to better optimize the $\boldsymbol{w_{uf}}$ and $\boldsymbol{h_{if}}$ factors for the ranking task. As stated above, the baselines use their own popular loss functions (both pointwise and pairwise), instead.

Performance Comparison: We compared AUC scores for all models under varying number of dimensions for latent user and item representations (d). The non-personalized baseline (Pop) shows the lowest performances. Among the rest of the personalized MF based approaches (Proposed, BPR-MF and MF), pairwise optimizations consistently outperform pointwise optimizations due to their natural support for ranking. However, since none of them are fully optimized for listwise ranking, the proposed listwise optimization criterion consistently outperformed all baselines and is also the least affected by the small variations of d (see Figure 6a).

Model Convergence: The total loss converges at around 60 epochs as the mean values of interacted and non-interacted ratings (μ_I and μ_{NI}) approach their maximum and minimum values (see Figures 6b and 6c). Simultaneously, the variance values for the corresponding classes (σ_I and σ_{NI}) lead to 0 at early epochs resulting in very low inter-class distances. After the overall loss reaches its minimum, the mean value of the interacted class slowly increases, while the mean value of the non-interacted class slowly decreases. At this stage, the model is still being marginally optimized, and it is visible from the continuous increase in AUC (see Figure 6d).

7 Conclusion and Further Work

Typical recommender systems fail to conduct recommendations for both new and existing users by considering the dynamic nature of user preferences. In addition, existing implicit feedback based systems are not fully optimized to rank a list of items. Thus to the best of our knowledge, we proposed the first time aware unified cross-network recommender solution with a generic listwise loss function. The proposed model first transfers and integrates user interactions from multiple networks in a cross-network topical layer. Second, captures dynamic user preferences in three temporal levels, namely short, long and long short term preferences. The proposed deep learning model directly utilized the computed source network user preferences for new user recommendations and for existing users, they are integrated with target network preferences. We used two datasets, CrossNet and MovieLens to evaluate the proposed model and the listwise optimization criterion and they consistently outperformed multiple state-of-the-art baselines. As future work, the proposed model can be extended with social (e.g., follower and followee) and item information. The intuition for the proposed listwise optimization criterion can also be extended for ranking under explicit feedback. Overall, we believe the paper makes a significant contribution to the cross-network recommender field and the use of implicit feedback.

8 Acknowledgments

This research has been supported by the Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number MOE2018-T2-1-103. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the Titan Xp GPU used in this research.

References

Avazpour, I.; Pitakrat, T.; Grunske, L.; and Grundy, J. 2014. Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering*. Springer. 245–273.

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

Campos, P. G.; Dıez, F.; and Cantador, I. 2012. A performance comparison of time-aware recommendation models.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, 129–136. ACM.

He, X., and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 355–364. ACM.

He, X.; Chen, T.; Kan, M.-Y.; and Chen, X. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 1661–1670. ACM.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182. International World Wide Web Conferences Steering Committee.

Hu, G.; Zhang, Y.; and Yang, Q. 2018. Conet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 667–676. ACM.

Huang, J.; Guestrin, C.; and Guibas, L. J. 2008. Efficient inference for distributions on permutations. In *Advances in neural information processing systems*, 697–704.

Kelly, D., and Teevan, J. 2003. Implicit feedback for inferring user preference: a bibliography. In *SIGIR forum*, volume 37, 18–28.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kondor, R.; Howard, A.; and Jebara, T. 2007. Multi-object tracking with representations of the symmetric group. In *Artificial Intelligence and Statistics*, 211–218.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 426–434. ACM.

Koren, Y. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 447–456. ACM.

Lim, B. H.; Lu, D.; Chen, T.; and Kan, M.-Y. 2015. # mytweet via instagram: Exploring user behaviour across multiple social networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 113–120. ACM.

Loni, B.; Pagano, R.; Larson, M.; and Hanjalic, A. 2016. Bayesian personalized ranking with multi-channel user feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 361–364. ACM.

McLaughlin, M. R., and Herlocker, J. L. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 329–336. ACM.

Mehta, B.; Niederee, C.; Stewart, A.; Degemmis, M.; Lops, P.; and Semeraro, G. 2005. Ontologically-enriched unified user modeling for cross-system personalization. In *International conference on user modeling*, 119–123. Springer.

Perera, D., and Zimmermann, R. 2017. Exploring the use of time-dependent cross-network information for personalized recommendations. In *Proceedings of the 25th ACM international conference on Multimedia*, 1780–1788. ACM.

Perera, D., and Zimmermann, R. 2018. Lstm networks for online cross-network recommendations. In *IJCAI*, 3825–3833.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, 452–461. AUAI Press.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*, 995–1000. IEEE.

Sang, J.; Deng, Z.; Lu, D.; and Xu, C. 2015. Cross-osn user modeling by homogeneous behavior quantification and local social regularization. *IEEE Transactions on Multimedia* 17(12):2259–2270.

Wanave, T. T., and Takale, S. A. 2016. Youtube video recommendation via cross-network collaboration. *International Journal of Computer Applications* 146(11).

Yan, M.; Sang, J.; Xu, C.; and Hossain, M. S. 2016. A unified video recommendation by cross-network user modeling. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 12(4):53.

Yan, M.; Sang, J.; and Xu, C. 2014. Mining cross-network association for youtube video promotion. In *Proceedings of the 22nd ACM international conference on Multimedia*, 557–566. ACM.

Yan, M.; Sang, J.; and Xu, C. 2015. Unified youtube video recommendation via cross-network collaboration. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 19–26. ACM.

Yu, R.; Zhang, Y.; Ye, Y.; Wu, L.; Wang, C.; Liu, Q.; and Chen, E. 2018. Multiple pairwise ranking with implicit feedback. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1727–1730. ACM.

Zhang, L. 2013. The definition of novelty in recommendation system. *Journal of Engineering Science & Technology Review* 6(3).

Zhao, W. X.; Jiang, J.; Weng, J.; He, J.; Lim, E.-P.; Yan, H.; and Li, X. 2011. Comparing twitter and traditional media using topic models. In *European conference on information retrieval*, 338–349. Springer.

Zhu, Y.; Li, H.; Liao, Y.; Wang, B.; Guan, Z.; Liu, H.; and Cai, D. 2017. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, 3602–3608.