Make It a Chorus: Knowledge- and Time-aware Item Modeling for Sequential Recommendation

Chenyang Wang, Min Zhang*, Weizhi Ma, Yiqun Liu, and Shaoping Ma
Department of Computer Science and Technology, Institute for Artificial Intelligence,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing, 10084, China
wangcy18@mails.tsinghua.edu.cn,z-m@tsinghua.edu.cn

ABSTRACT

Traditional recommender systems mainly aim to model inherent and long-term user preference, while dynamic user demands are also of great importance. Typically, a historical consumption will have impacts on the user demands for its relational items. For instance, users tend to buy complementary items together (iPhone and Airpods) but not substitutive items (Powerbeats and Airpods), although substitutes of the bought one still cater to his/her preference. To better model the effects of history sequence, previous studies introduce the semantics of item relations to capture user demands for recommendation. However, we argue that the temporal evolution of the effects caused by different relations cannot be neglected. In the example above, user demands for headphones can be promoted after a long period when a new one is needed.

To model dynamic meanings of an item in different sequence contexts, a novel method *Chorus* is proposed to take both item relations and corresponding temporal dynamics into consideration. Chorus aims to derive the embedding of target item in a knowledge-aware and time-aware way, where each item will get its basic representation and relation-related ones. Then, we devise temporal kernel functions to combine these representations dynamically, according to whether there are relational items in history sequence as well as the elapsed time. The enhanced target item embedding is flexible to work with various algorithms to calculate the ranking score and generate recommendations. According to extensive experiments in three real-world datasets, Chorus gains significant improvements compared to state-of-the-art baseline methods. Furthermore, the time-related parameters are highly interpretable and hence can strengthen the explainability of recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

Recommender system; Item relations; Knowledge-aware recommendation; Temporal dynamics

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401131

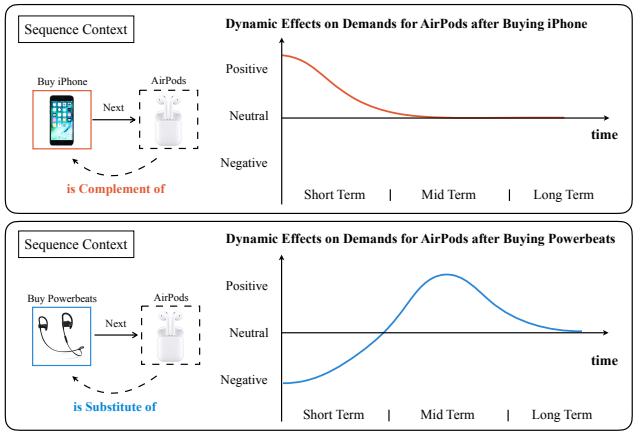


Figure 1: Illustration for temporal effects of previous relational consumptions on the demands for AirPods. Current consumption for an iPhone may have positive effects in the short term, but the impact is oppositely negative if the user has just purchased Powerbeats. The positive effect of complement decays with time, while the negative effect of substitute can turn to positive after a period of time.

ACM Reference Format:
Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Make It a Chorus: Knowledge- and Time-aware Item Modeling for Sequential Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401131

1 INTRODUCTION

With the overload of information on the Internet, the recommender system has been playing an increasingly important role in daily life. It not only provides information catering to users' tastes but also helps to discover their intrinsic preference [1]. Traditional recommendation methods mainly focus on user preference modeling [11, 13, 19, 33]. For example, the latent factor model [21] embeds both users and items into a latent space, where the embeddings of users represent the preference in various aspects, which will not change when making recommendations at different times.

However, although user preference is static most of the time, user consuming demands are actually dynamic and changeable. The same item will have different meanings to a user in different

* Corresponding author.

contexts. Actually, sequential consuming behavior can be seen as a process to fulfill user demands in distinct aspects. A consumption for an item may have impacts on other relational items, and such effects of distinct relations also differ from each other. Take complementary and substitutive relations as example, Figure 1 illustrates how the effects of purchasing different relational items change with time. For **complements**: assuming a user purchases an iPhone currently, he/she may want to buy AirPods (i.e. a complement of the iPhone) in the short term. But the positive effect will be lowered after a period of time (the user may already have headphones, and the recommender system should not present AirPods continually). For **substitutes**: if what he/she has just consumed is a substitute for AirPods, such as Powerbeats, the short-term impacts are mainly expected to be negative because the user does not need another headphone immediately. While the negative effect can turn to positive in the mid term, since the user may need to buy a new headphone, and the newly released AirPods could be an attraction. Such positive effects will also gradually decrease, in which case the user may have lost interests in this kind of headphones or have purchased another one by other means.

From the above illustration, we can see that current consumptions for distinct relational items have different impacts on the target item. More crucially, the temporal trends also differ from each relation. There are also studies introducing item relations into recommender systems [16, 40, 44, 45], but they do not take temporal dynamics of different relations into consideration. Although some work addresses long-term and short-term preference [16], the item relations are only used to model short-term item transitions, lacking in the modeling of continuous evolution of different relations' effects. Another recent work investigates the temporal dynamics of repeat consumption [41]. However, a consumption will influence not only the same item itself but also relational items. Thus, both item relations and corresponding temporal dynamics are critical to capture the dynamic meanings of an item in different contexts.

In this paper, we propose a novel method *Chorus*, aiming to get knowledge- and time-aware target item embeddings. To the best of our knowledge, we are the first to explicitly model the evolution of different relations' effects with time, which helps to better capture the meaning of each item in different sequence contexts. In particular, Chorus assigns a basic representation and various relational ones for each item based on translation-based graph embedding methods. Then, these representations are combined dynamically by temporal kernel functions, depending on the elapsed time since relational consumptions, which is the reason why it is named Chorus. The proposed temporal kernel functions enable relational representations to contribute differently to the final item embedding. As a result, Chorus is capable of obtaining knowledge- and time-aware item embeddings dynamically, which are easy to be leveraged by various recommendation algorithms. Furthermore, the highly interpretable time-related parameters make it possible to explain recommendation results at different time periods. The main contributions of this work can be summarized as follows:

- We propose to take both item relations and corresponding temporal dynamics into consideration. To the best of our knowledge, we are the first to explicitly model the continuous temporal evolution of different relations' effects.
- We devise a novel and flexible method *Chorus*, which enhances target item modeling by dynamically combining different representations when the target item acts as distinct roles in the sequence. The final item embedding can easily work with various recommendation algorithms.
- Comparative experiments in three real-world datasets show the effectiveness of Chorus, and the highly interpretable parameters further help to enhance model explainability.

2 RELATED WORK

2.1 Sequential Recommendation

Different from traditional recommendation methods, sequential recommendation utilizes sequential data to predict a user's next consumption based on Markov chains, which assume that the next action depends on the previous action sequence [34, 37]. Rendle et al. [34] combines Matrix Factorization (MF) [21] and factorized Markov Chains to make next-basket recommendation given the previous basket items. More recently, there has been a lot of work utilizing Recurrent Neural Network (RNN) [36] to encode interaction history to hidden vectors [6, 12, 23, 27, 32, 38]. Hidasi et al. [12] first introduce RNN to sequential recommendation and achieve impressive performance gain. Loyola et al. [27] and Pei et al. [32] both apply attention mechanism [39] to RNN for more effective recommendation. Besides, numerous follow-up studies focus to extend the capacity of the RNN-based model.

Despite the great expressiveness of RNN-based sequential recommendation methods, they still cannot well model sophisticated user demands for the lack of external knowledge, and substantially suffer from the interpretability issue [26]. Differently, our method obviously addresses both item relations and corresponding temporal dynamics to better capture user demands.

2.2 Item Relation Modeling

In real-world applications, there are typically multiple relations between items that have concrete semantics. Some recent work has been focusing on how to introduce item relations into recommender systems [16, 28, 30, 40, 44, 45], most of which utilize Knowledge Graph (KG) [42] to represent item relations. CFKG [45] introduces *user* to item relation graph as entities and view the action *buy* as another relation, and then uses TransE [3] to represent the heterogeneous information network and makes recommendations. Xin et al. [44] propose a general recommendation task that incorporates multiple relations between items, and integrate relational data to Collaborative Filtering (CF) [35]. Ma et al. [28] propose a joint learning framework to integrate the induction of explainable rules from knowledge graph.

However, all these methods assume the effects of relational item consumptions are static and independent with temporal information, in which case complements may be persistently recommended after a long time, even if the user does not need them.

2.3 Temporal Dynamics Modeling

There are mainly two lines of work taking temporal information into consideration. On the one hand, some work aims to take temporal information as context features. TimeSVD++ [20] divides time into slots and devises time-related parameters. TransFM utilizes

FM to include timestamp as an extra context feature [31]. Besides, tensor factorization is also a major method [2, 17], where time is viewed as the third dimension of user-item interaction cube. On the other hand, some work focuses to model temporal decay effects of historical interactions. In this line of work, Hawkes Process (HP) [8] is always utilized to model mutual-exciting characteristics of user consumption sequence [7, 22, 24, 41]. Du et al. [7] first apply Hawkes Process to time-sensitive recommendation. SLRC [41] combines Hawkes Process and Collaborative Filtering to model temporal dynamics of repeat consumption.

However, these methods do not consider the temporal dynamics of different relations. As a result, to better model dynamic user demands, we creatively take both item relations and corresponding temporal evolution into consideration.

3 PRELIMINARIES

3.1 Task Definition

Definition 3.1 (Problem Definition). Given user $u \in \mathcal{U}$ and interaction history $S_u = \{(i_1, t_1), (i_2, t_2), \cdots, (i_{N_u}, t_{N_u})\} \in \mathcal{S}$ with N_u interactions ($t_n < t_{n'}$ for any $n < n' < N_u$), the recommendation task is: considering the interaction sequence before the target time t, denoted as S_u^t, generating an ordered list containing k items that the user may be interested in at t.

Besides, let $\mathcal{R}$ be the set of all item relations, and each relation $r \in \mathcal{R}$ has a matrix $\boldsymbol{I}_r \in \mathbb{N}^{M \times M}$, where M is the total number of items and $\boldsymbol{I}_r(i, j) = 1$ if relation r holds for item i and j, otherwise 0. Relation r can be *is_complement_of*, *is_substitute_of* and so on.

3.2 Knowledge Graph Embedding

The information of item relations can be viewed as a knowledge graph, and the component is a set of triplets (i, r, j), where i and j denote different items and r denotes relation types. For instance, $(AirPods, is\_complement\_of, iPhone)$ means Airpods is a complement of iPhone. Note that sometimes the opposite for a triplet may not hold (e.g. iPhone is not a complement of AirPods), hence the relation graph is directional.

To introduce structural information of relation graph into recommender systems, it is important to get embeddings with semantic meanings of item relations. Among various embedding methods, translation-based models [3, 25, 43] stand out for their efficiency and effectiveness. The inherent idea is embedding items and relations into the same latent space and finding a translation function to minimize the scoring function:

$$\min_{\Theta} \ f(i, r, j) = D\left(\text{Trans}(\boldsymbol{i}, \boldsymbol{r}), \boldsymbol{j}\right) \ , \tag{1}$$

where $D(\cdot)$ is a metric function to measure the distance (usually l_2-norm). $\text{Trans}(\boldsymbol{i}, \boldsymbol{r})$ is an arbitrary translation function, which can be a simple translation operation or a specially designed neural network. A lot of work has been focusing on extending the capacity of translation function, such as TransE [3], TransH [43], TransR [25] and so on. In the case of TransE [3], the translation function is $\text{Trans}(\boldsymbol{i}, \boldsymbol{r}) = \boldsymbol{i} + \boldsymbol{r}$, and the scoring function for any triplet (i, r, j) when applying l_2-norm is $f(h, r, t) = ||\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}||_2$.

To learn item and relation embeddings from relation graph, a margin-based loss [45] is minimized as follows:

$$\begin{aligned} \mathcal{L}_{rel} = &\sum_{(i,r,j,j')} \left[f(i, r, j) - f(i, r, j') + \gamma \right]_+ \\ &+ \sum_{(i,r,j,i')} \left[f(i, r, j) - f(i', r, j) + \gamma \right]_+ \ . \end{aligned} \tag{2}$$

For every triplet, the tail item is replaced by a random sampled item j' to make sure (i, r, j') is not observed in the knowledge graph. Similarly, the head item is replaced by i' and (i', r, j) does not hold. The above objective function aims to discriminate the observed triplets from the corrupted ones, and the embeddings will be forced to retain relations between items.

3.3 Base Methods for Recommendation

The item modeling method proposed in this work is flexible to work with various recommendation algorithms. For the reason that Bayesian Personalized Ranking (BPR) [33] is a widely used matrix factorization method and Generalized Matrix Factorization (GMF) [11] is a state-of-the-art method based on neural networks, we choose them as base recommendation models to verify the effectiveness of our method.

Here we briefly review these two collaborative filtering methods. CF methods assume similar users like similar items. In the case of BPR, there is a K-dimensional latent factor for each user and item, and the ranking score is calculated as follows:

$$\hat{y}_{ui} = \boldsymbol{u}^T \boldsymbol{i} + b_u + b_i \ , \tag{3}$$

where b_u and b_i are bias of each user and item, respectively.

In the case of GMF, the ranking score is derived by a multi-layer neural network, which can be formulated as

$$\hat{y}_{ui} = \phi_{out}\left(\phi_X\left(\cdots \phi_2\left(\phi_1\left(\boldsymbol{u}^T, \boldsymbol{i}^T\right)\right)\cdots\right)\right) \ , \tag{4}$$

where ϕ_{out} and ϕ_x respectively denote the mapping function for the output layer and x-th neural collaborative filtering layer, and there are X neural CF layers in total. Then, candidate items are ranked according to the predicted score $\hat{y}_{ui}$.

To learn parameters in recommendation model, a pair-wise ranking loss [33] can be optimized as follows:

$$\mathcal{L}_{rec} = -\sum_{u \in \mathcal{U}} \sum_{i=1}^{N_u} \log \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) \ , \tag{5}$$

where σ denotes the sigmoid function and a negative item $j \notin S_u$ is randomly sampled for each training instance.

4 CHORUS MODEL

4.1 Model Overview

Chorus is a two-stage model, which integrates both item relations and their specific temporal effects. Figure 2 demonstrates the overall model structure. At the first stage (**Relation Modeling**), graph embedding is utilized to encode structural information of item relations into embeddings. Various translation-based methods described in Section 3.2 are flexible to be leveraged here. The results of relation graph embedding will be used to derive the basic and relational representations in our Chorus model.

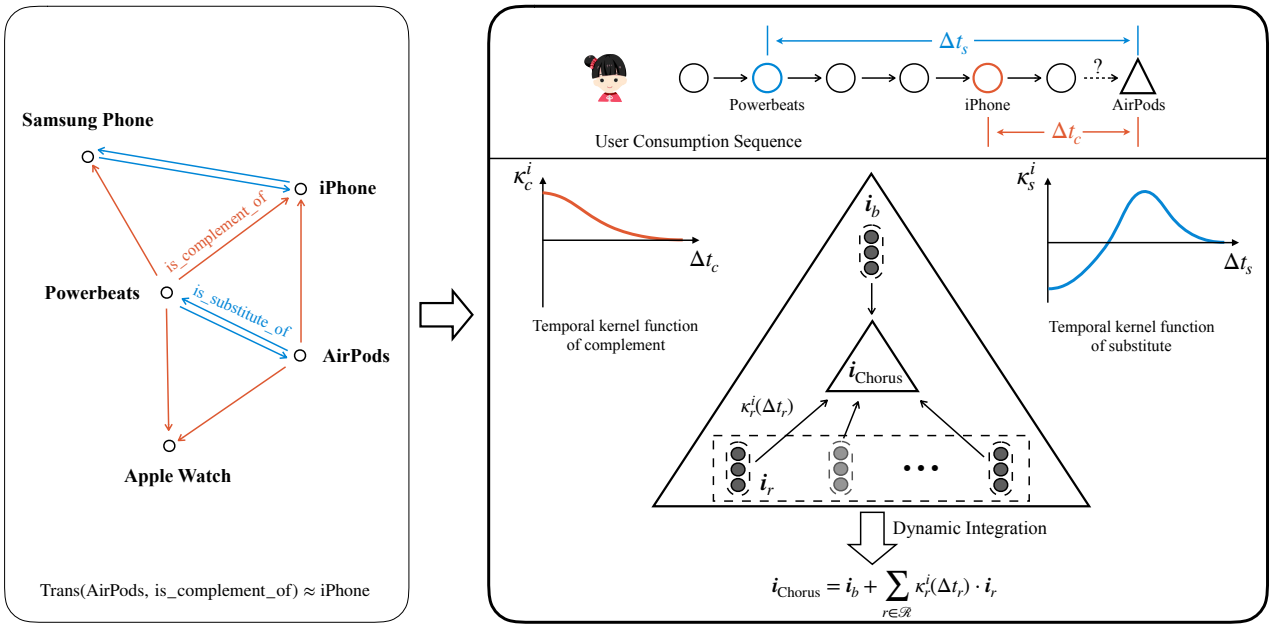


Figure 2: Chorus model overview. At the first stage (Relation Modeling), graph embedding is utilized to learn basic embeddings of items and relations. At the second stage (Dynamic Item Representation), Chorus gives each item extra relational representations. Then, these representations are dynamically combined according to the sequence contexts. The final knowledge-aware dynamic item embedding can be used for recommendation in various algorithms (e.g. BPR and GMF).

At the second stage (**Dynamic Item Representation**), there are two key modules: (1) dynamic integration, and (2) design of temporal kernel functions. First, each item will get $|\mathcal{R}|$ relational representations besides the basic one based on the translation function, which denote the representations of the target item when acting as different roles in the context. Then, these representations are dynamically integrated according to whether there are corresponding relational consumptions in the history sequence as well as the elapsed time. To incorporate temporal dynamics of each relation, we propose relation-specific temporal kernel function to control the polarity and intensity of effects. As a result, relational representations contribute differently to the final item embedding in distinct contexts, leading to knowledge-aware dynamic item embeddings. Finally, the enhanced item embeddings can be leveraged by many algorithms to calculate ranking scores and make recommendations. In the remainder of this section, we elaborate the key modules of Chorus at the second stage.

4.2 Dynamic Integration

First, we define basic representation (denoted as $\boldsymbol{i}_b$) and relational ones (denoted as $\boldsymbol{i}_r$ for relation $r \in \mathcal{R}$) for each item based on the result of relation graph embedding. The basic representation encodes the inherent characteristics of an item, so the item embeddings learned at the first stage are used to initialize $\boldsymbol{i}_b$. Then, the translation function is utilized to get relational representations:

$$\boldsymbol{i}_r = \text{Trans}(\boldsymbol{i}_\text{b}, \boldsymbol{e}_\text{r}) , \tag{6}$$

where $\boldsymbol{e}_r$ is the relation embedding for $r \in \mathcal{R}$. In this way, the relational representations integrate the semantic information corresponding to each relation.

After getting basic and relational item representations, here we focus on how to combine them dynamically according to different contexts, which is the core idea of our Chorus model. Note that the relational representations are knowledge-aware but still static. We aim at deriving a context-aware coefficient f_r for each relational representation, reflecting the actual degree of effects in current context. The final context- and knowledge-aware item embedding $\boldsymbol{i}_\text{Chorus}$ is proposed to be represented as follows:

$$\boldsymbol{i}_\text{Chorus} = \boldsymbol{i}_b + \sum_{r \in \mathcal{R}} f_r(S_t^u, t, i) \cdot \boldsymbol{i}_r \ . \tag{7}$$

It consists of two parts: a basic item representation and scaled relational ones, where the contexts (history sequence S_t^u, time t, and target item i) serve as the input of coefficient f_r. Next we focus on how to get reasonable f_r given the context.

Intuitively, some relational representations may have no effect or even negative effects in some cases. Figure 3 gives some examples on how these representations are expected to contribute to the final embedding in different contexts. The three angles of the triangle stand for different representations of the target item. When there are no relational consumptions (context A), the final embedding is just basic item representation, and the other two relational ones take no effects. When Powerbeats or iPhone is just bought (context B and C), the corresponding relational representation should

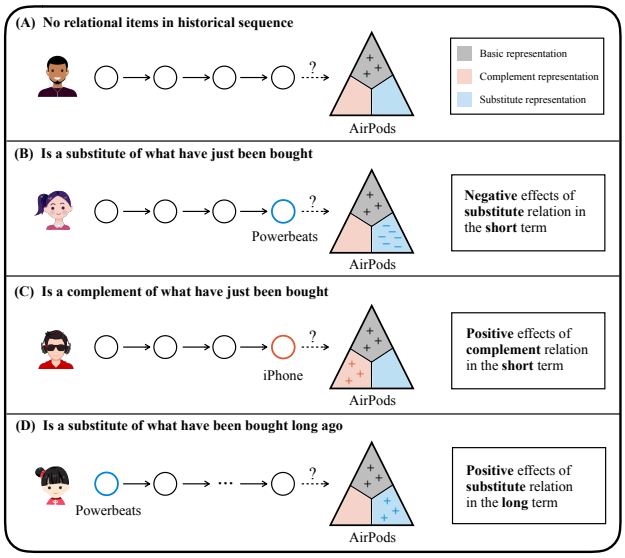


Figure 3: Illustration of how different representations are combined dynamically depending on the sequence context.

have negative and positive effects, respectively. While if the substitute is bought long ago (context D), the substitute representation may pose positive effects on the final embedding. Besides, when there are many different relational items in the sequence, the three representations will all take effect in different degrees.

To incorporate such temporal dynamics of different relations, we innovatively devise *temporal kernel function* for each relation, which is a continuous function of the lag time between consumptions. Temporal kernel function aims to control the degree of influence for each previous relational consumption. The polarity of function value denotes the polarity of effects. Assuming we have obtained temporal kernel function $\kappa_r^i(\Delta t)$, indexed by item i and relation r (the concrete design and related discussions are left in the next Section), we propose to define the relational coefficient f_r as follows:

$$f_r(S_t^u, t, i) = \sum_{(i',t') \in S_u^t} \boldsymbol{I}_r(i, i') \kappa_r^i(t - t') \,, \tag{8}$$

where $\boldsymbol{I}_r$ is the relation matrix. Each previous consumption with relation r to the target item i will have additive effects to the coefficient f_r, controlled by the kernel function $\kappa_r^i(\cdot)$. In this way, the relational coefficients enable distinct representations to contribute to the final embedding in different degrees, unlike previous studies that assign static item embeddings. Due to the temporal kernel function, a relational representation may barely take effect because of the long-time gap, or even has a negative effect under some circumstances. As a result, the Chorus embeddings can better capture the meanings of items in different contexts, and thus better model users' changing demands with time.

Besides, for simplicity and efficiency, we can consider only the latest relational item in history sequence, in which case Equ. (7) can be directly represented as:

$$\boldsymbol{i}_{\text{Chorus}} = \boldsymbol{i}_b + \sum_{r \in \mathcal{R}} \kappa_r^i(\Delta t_r) \cdot \boldsymbol{i}_r \,, \tag{9}$$

where Δt_r represents the elapsed time since the latest item consumption having relation r with the current item. If there is no relational item in history sequence for a relation, we assume there is one with a positive infinity time gap (i.e. $\Delta t_r = +\infty$). Provided the temporal kernel function approaches zero with time, corresponding relational embedding will take no effects.

4.3 Design of Temporal Kernel Function

Next, we focus on how to design the temporal kernel function for each relation. Actually, the concrete form of temporal kernel function can be seen as a kind of human intervention to the model. On the one hand, we can design the function according to the characteristics of each relation. For example, as shown in Figure 1, complementary relation has positive impacts in a short period, and such effects decay with time. On the other hand, we can design it based on subjective requirements on the system. If we want substitutes to also appear in the recommendation list in the short term, the temporal kernel function can be designed to have positive initial value and decay quicker. In this work, we mainly focus on two relations: *is_complement_of* and *is_substitute_of*. As an example, we subsequently design corresponding temporal kernel functions according to general perceptions and characteristics of these two relations.

For complements, besides the overall decrease trend, the positive effects generally hold for a while and then begin decaying in daily life. As a result, instead of the intuitive exponential distribution, which decays too quickly, we choose normal distribution with zero mean to be its temporal kernel function:

$$\kappa_c^i(\Delta t) = N(\Delta t | \, 0, \sigma_c^{z(i)}) \,, \tag{10}$$

where $N(\Delta t | \mu, \sigma)$ is a normal distribution of Δt with μ mean and σ standard deviation. Note that the parameter $\sigma_c^{z(i)}$ here is related to $z(i)$, which denotes the category of item i. We do not estimate item-specific parameters because category is usually a more suitable level to model characteristics of a cluster of items. And item-specific parameters may also suffer from data sparsity issues.

For substitutes, the impact is expected to turn from negative to positive, because we do not need another item with similar utility in the short term but would like to change to a new one when its lifetime runs out [41]. As a result, we use two opposite normal distributions to model such characteristics:

$$\kappa_s^i(\Delta t) = \overbrace{-N(\Delta t | \, 0, \sigma_{s1}^{z(i)})}^{negative} + \overbrace{N(\Delta t | \, \mu_s^{z(i)}, \sigma_{s2}^{z(i)})}^{positive} \,, \tag{11}$$

which is a superposition of (1) short-term restrain (negative) and (2) life-time promotion (positive). The negative normal distribution is designed to have zero mean because the retraining effect is generally the strongest just after the substitute consumption. In the positive one, the parameter $\mu_s^{z(i)}$ represents the lifetime of this category in some way, meaning the effect will peak at this time.

Illustrations of these two temporal kernel functions are shown in Figure 2. One can also design other forms of temporal kernel functions to meet different needs. Besides, Chorus is not restricted to these two relations. Many relations can be incorporated, such as *same_brand*, *same_producer* and so on. The only thing to do is designing a corresponding temporal kernel function based on prior

Table 1: Comparison between baselines and Chorus.

Model	Temporal Dynamics	Sequential Info	Relation Modeling
Tensor	✓		
GRU4Rec		✓	
CFKG			✓
SLRC'	✓	✓	
Chorus	✓	✓	✓

knowledge, and we find an exponential distribution basically works well for general relations.

Here we have got the final knowledge-aware dynamic item embeddings. Then various algorithms can be utilized to make recommendations by replacing origin target item embeddings with ours. Different from previous models, Chorus simultaneously integrates sequential information, item relations modeling, and corresponding temporal dynamics. The recently proposed CFKG and SLRC either focus on only item relations or temporal dynamics in consumption sequence. Table 1 lists the difference between related methods and our Chorus model. More details about these baselines will be described in Section 5.1.3.

4.4 Parameter Learning

To get better and robust performance, a two-stage training procedure is applied to learn model parameters: first, we optimize $\mathcal{L}_{rel}$ to get item and relation embeddings with structural information, which are utilized to initialize basic item representations and relation embeddings at the second stage; then we minimize $\mathcal{L}_{rec}$ to learn all parameters of the model. At the second stage, we do not freeze the embeddings learned before. Experiments show that optimizing with $\mathcal{L}_{rec}$ yields better results. On the other hand, it is also possible to corrupt the meaningful embeddings at the beginning of training. Therefore, we scale down the learning rate by 0.1 for basic item representations and relation embeddings at the second stage. Adam [18] is utilized as the learning algorithm at each stage due to its success in many recommendation models.

5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 ***Datasets.*** Experiments are conducted on publicly accessible Amazon dataset [9]. Besides the user interaction sequences with timestamps, it also has item metadata, including the list of *also_view*, *also_buy* and category information. Following previous studies [28, 29], we take *also_view* as substitutes and *also_buy* as complement relation. Differently, in our work the relation means *is_complement_of* and *is_substitute_of*. Hence the direction of the original *also_view*, *also_buy* relation should be reversed.

We adopt three representative sub-datasets: *Grocery and Gourmet Food* (Grocery), *Cellphones and Accessories* (Cellphones), and *Home and Kitchen* (Home). Table 2 summarizes the statistics of the three datasets. Note that in the Home dataset, the ratio of test cases that are relational to historical items is low and the relational data is comparatively sparse.

Table 2: Statistics of datasets.

Dataset	#user ($\lvert\mathcal{U}\rvert$)	#item ($\lvert\mathcal{I}\rvert$)	#entry ($\sum_u N_u$)	#triplet ($\sum_r \lvert\lvert I_r \rvert\rvert_1$)	relational ratio in test set
Grocery	14.7k	8.5k	145.8k	372.1k	27.8%
Cellphones	27.9k	10.3k	193.2k	247.5k	30.0%
Home	66.5k	27.2k	541.6k	924.6k	16.6%

5.1.2 ***Evaluation Protocols.*** We adopt the leave-one-out evaluation, which is widely used in the literature [4, 10, 15]. For each consumption sequence $S_u \in \mathcal{S}$, we use the most recent interaction of each user for testing, the second recent item for validation, and the remaining items for training. Considering it is time-consuming to rank all items for some methods when the dataset is large, we randomly sample 99 items that the target user has not interacted with and rank the ground-truth item together with these negative items. This method is also widely adopted [11, 41, 44].

To evaluate recommendation quality, we use *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) [14] as evaluation metrics. HR@k measures whether the ground-truth item appears in the top-k recommendation list, while NDCG@k concerns about the position in the ranking list. We repeat each experiment 5 times with different random seeds and report the average score.

5.1.3 ***Baseline Methods.*** We compare our Chorus model to seven baseline methods in different aspects, including traditional collaborative filtering, sequential recommendation, and methods incorporating item relations or temporal dynamics:

- **BPR** [33]: This method proposes to apply a pairwise ranking loss to optimize matrix factorization model.
- **GMF** [11]: This is a state-of-the-art collaborative filtering method that utilizes a multi-layer neural network.
- **Tensor** [17]: This method splits time into bins and factorizes a three-dimensional tensor (user-item-time).
- **GRU4Rec** [12]: This is a sequential recommendation model that applies GRU [5] to derive the ranking score.
- **NARM** [27]: This model utilizes GRU and attention mechanism to improve the performance of sequential recommendation, which is a state-of-the-art session-based method.
- **CFKG** [45]: This method takes various item relations into consideration and views *buy* as another relation between users and items. Then, TransE is utilized to learn graph embeddings and make recommendations.
- **SLRC'** [41]: SLRC combines Hawkes and CF to model temporal dynamics of repeat consumption. Considering that repeat consumptions have been removed in Amazon dataset, we extend its setting to the effects of relational items, named SLRC'. But it still lacks semantic modeling of item relations.

5.1.4 ***Implement Details.*** We implement all the models in *PyTorch*. The implementation codes have been released[1]. For fair comparison, the embedding size is set to 64 for all models. All the hyper-parameters are tuned to get the best results in the validation dataset. For CFKG, we consider *also_view* and *also_buy* relations

[1] https://github.com/THUwangcy/ReChorus

Table 3: Test results in three datasets. We repeat each experiment five times with different random seeds and report the average score. Best baseline on each metric is underlined, and ** means significantly better than the strongest baseline ($p < 0.01$).

Method	*Grocery and Gourmet Food*				*Cellphones and Accessories*				*Home and Kitchen*			
	k=5		k=10		k=5		k=10		k=5		k=10	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
BPR	0.3242	0.2223	0.4315	0.2571	0.3260	0.2349	0.4364	0.2705	0.2542	0.1718	0.3701	0.2091
GMF	0.3051	0.2089	0.4100	0.2429	0.2866	0.2030	0.3910	0.2367	0.2500	0.1671	0.3693	0.2055
Tensor	0.3478	0.2623	0.4471	0.2943	0.3560	0.2489	0.4888	0.2917	0.2897	0.1941	0.4216	0.2366
GRU4Rec	0.3704	0.2643	0.4721	0.2972	0.4112	0.2956	0.5453	0.3389	0.2953	0.2025	0.4187	0.2423
NARM	0.3590	0.2573	0.4634	0.2910	0.4092	0.2938	0.5440	0.3373	0.2901	0.1976	0.4137	0.2375
CFKG	0.4337	0.3081	0.5628	0.3499	0.4465	0.3264	0.5677	0.3656	0.2609	0.1760	0.3801	0.2144
SLRC'	0.4513	0.3329	0.5649	0.3698	0.4440	0.3433	0.5414	0.3747	0.3275	0.2452	0.4346	0.2797
$\text{Chorus}_{\text{BPR}}$	**0.4754****	0.3448**	**0.5998****	0.3852**	0.4593**	0.3439	0.5784**	0.3824**	**0.3405****	**0.2473****	**0.4572****	**0.2849****
$\text{Chorus}_{\text{GMF}}$	0.4748**	**0.3467****	0.5960**	**0.3861****	**0.4623****	**0.3481****	**0.5809****	**0.3863****	0.3350**	0.2461*	0.4433**	0.2811**

consistent with ours. For SLRC' and Chorus, we find there are few interactions that have two or more relational items in the history sequence. Thus, for simplicity and efficiency, we consider the latest relational interaction in the sequence without loss of generality and performance. Besides, TransE is utilized as the translation function in Chorus. All the time-related parameters are initialized with 1 for numeric stability, and the other parameters are normally initialized with 0 mean and 0.01 standard deviation.

5.2 Overall Performance

Table 3 shows the performance of all baselines and our Chorus model when utilizing BPR and GMF to calculate the ranking score, denoted as $\text{Chorus}_{\text{BPR}}$ and $\text{Chorus}_{\text{GMF}}$ respectively.

First, different kinds of baselines demonstrate obvious performance gaps. For collaborative filtering methods (i.e. BPR and GMF), they serve as benchmarks because the only information they have is user-item interactions. Tensor outperforms basic CF methods by taking temporal dynamics into consideration. Sequential recommendation methods (i.e. GRU4Rec and NARM) further achieve better performance, which demonstrates the importance of dynamic user demands conveyed by recently consumed items. CFKG gets fair results and becomes the best baseline in terms of some metrics, which indicates that item relation is indeed helpful to recommendation. With regard to SLRC', it generally gets the best results among baselines because of its explicit modeling of mutual-exciting characteristics in consumption sequence.

Second, our Chorus model performs consistently better than other baselines in all the datasets, which benefits from addressing both item relations and their temporal dynamics. This shows the proposed model is capable of better capturing dynamic user demands and the meaning of items in different contexts. Compared with CFKG, Chorus not only considers item relations but also integrates their temporal dynamics. Compared with SLRC', Chorus is able to model semantic meaning and category-specific temporal effects of each relation. The basic form of Hawkes in SLRC' may focus more on the impacts of relational items, and hence hurts performance on normal cases when there is no previous relational consumption (more discussions in Section 5.4). Differently, Chorus integrates item relations into knowledge-aware dynamic item representations, which is more effective and flexible.

On the other hand, notice that in the Home dataset, the improvement is comparatively small. The possible reason is that the relational information is too sparse and not that reliable. We utilize similar relation graph embedding method of CFKG and it also performs badly in this dataset. Although TransE works well in the other datasets, the relations in the Home dataset may be so sophisticated that TransE is insufficient to model accurately. More evidences are provided in related discussion in Section 5.3.

5.3 Ablation Study

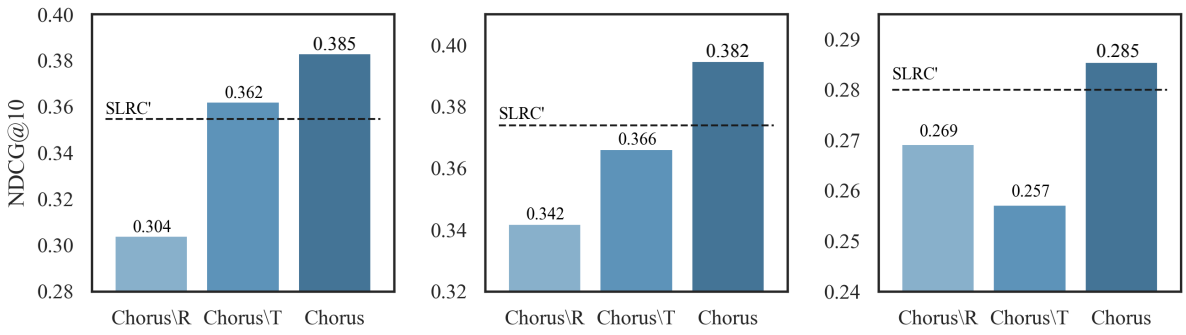
To verify the effects of relation modeling and temporal dynamics addressed in our model, we compare Chorus with two variants:

- Chorus\R. This model assigns separate item embeddings for each relation and estimates all parameters by optimizing $\mathcal{L}_{rec}$. The results of graph embedding are not used to initialize basic representation and derive relational ones.
- Chorus\T. This model does not take temporal dynamics of relations into consideration and assumes all the temporal kernel functions (i.e. $\kappa_r^i(\Delta t)$) are constant with value 1.

Figure 4 shows NDCG@10 of $\text{Chorus}_{\text{BPR}}$ and its variants, as well as SLRC'. It can be concluded that both relation modeling and temporal dynamics are of great importance. The lack of any module of Chorus results in performance loss. Furthermore, we have the following observations:

First, item relations are indeed helpful. Chorus\R brings the greatest performance loss in the Grocery and Cellphones datasets, which indicates the importance of modeling structural information of relations, as well as our translation-based method of deriving relational representations through graph embeddings.

Second, it is important to model the temporal dynamics of different relations. Without temporal information, Chorus\T results in a moderate loss of performance in the first two datasets. This does not mean the temporal dynamics addressed in our model is



(a) Grocery and Gourmet Food **(b) Cellphone and Accessories** **(c) Home and Kitchen**

Figure 4: Ablation study. Performance comparison between Chorus and its variant without relation modeling (Chorus\R), and variant without temporal dynamics (Chorus\T).

not important. Literally, relations between items have a greater influence on users when making consumption decisions. Thus, it is reasonable that modeling item relations leads to larger improvements than modeling temporal dynamics. On the other hand, Chorus achieves consistent improvements compared to Chorus\T, especially in *Home*, which shows the usefulness of moving forward to take temporal dynamics of item relations into consideration.

Third, the relation graph embedding does not work well in the *Home* dataset, where Chorus\T leads to the greatest performance loss, but not Chorus\R. This is another evidence of the inadequate relation modeling. Although TransE is a natural choice for translation function and generally works well on the other two datasets, we find it may be insufficient under scenarios in the Home dataset, where CFKG also performs badly with TransE as its graph embedding method. Figure 4 (c) shows the performance does not decline too much without relation modeling (Chorus\R). But if there is relation modeling but no temporal kernel functions (Chorus\T), the performance will be hurt by inappropriate embeddings of items and relations. This also shows the temporal dynamics addressed in our model can help to adaptively avoid probably bad effects of messy relational ones, which demonstrates the usefulness and necessity of taking temporal dynamics into consideration.

5.4 Performance in Different Scenarios

Besides the overall performance improvement, we also want to figure out where does the improvement comes from. Here we investigate the performance of models in different scenarios. Specifically, we construct three subsets of the test dataset according to whether there are corresponding relational consumptions in the history sequence. *Normal* means there are no relational items previously. *Complement* means the target item is a complement of some items in the history sequence. Similarly, *Substitute* is the case when there are previous consumptions serving as substitutes. A test case may be in Complement and Substitute group simultaneously when there are both kinds of relational items before. Figure 5 shows NDCG@10 of different models (lines) and the number of cases (bars) in the three subsets of *Cellphones* dataset. We can see although there are fewer cases in Complement and Substitute groups, models tend to perform better on these cases. They may inherently demonstrate some patterns, so that all the models achieve better performance compared to normal cases, even for BPR that does not explicitly take item relations into consideration.

Besides, Chorus is able to combine the advantages of different kinds of methods, and hence achieve the best performance on average. Notice that for SLRC' and CFKG, they both have their strengths and weakness. Although SLRC' performs well on relational cases, especially in the Substitute group, it is even worse than BPR on normal cases. This shows SLRC' is prone to overfit the relational cases, which in turn hurts the performance of normal cases. On the other hand, although CFKG performs well in the Normal group, it is less powerful than SLRC' on relational cases because SLRC' explicitly models the temporal characteristics of each relation. As for Chorus, it captures both item relations and their category-specific temporal dynamics. It is noteworthy that on normal cases, Chorus is similar to CFKG; and on complement cases, Chorus is a little better than SLRC', which are both the best baseline in each scenario. Although Chorus is not as strong as SLRC' in the Substitute group, it gets obvious improvement compared to CFKG. As a result, Chorus gains significantly better results on average, which shows the importance of integrating both item relations and fine-grained temporal dynamics.

5.5 Parameter Interpretability

Here we want to validate whether the time-related parameters have interpretable meanings as we design the temporal kernel function. Note that these parameters are indexed by item category, which represents how the impacts of previous relational consumptions on this category drift with time. Although the overall trends for a specific relation are similar because of the functional form, their concrete forms reveal the characteristics of the category. Figure 6 shows the temporal kernel functions for some representative categories learned in the *Cellphones* dataset.

The left figure is corresponding to *is_complement_of* relation in terms of *Headsets* and *Replacement Parts*. As shown in the figure, the effect on headsets decays much quicker than replacement parts. On the one hand, after purchasing a cellphone, it is reasonable to recommend headsets to the user as a complement. But if the user does not interact with the recommended headsets, he/she may

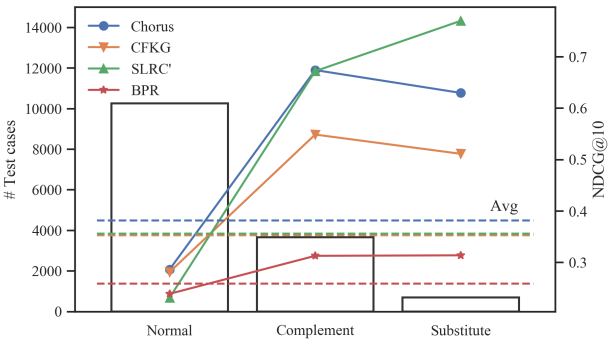
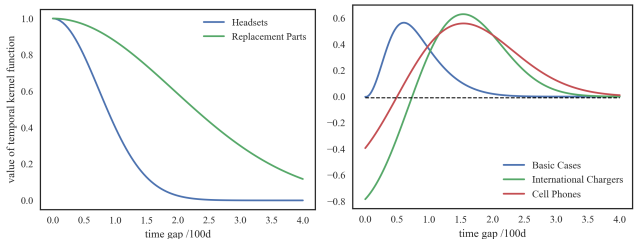


Figure 5: Performance comparison when test cases are in different scenarios (*Normal*: no historical relational consumptions; *Complement*: the target item is a complement of some items in the history sequence; *Substitute*: the target item is a substitute of some items in the history sequence).



(a) is_complement_of **(b) is_substitute_of**

Figure 6: Case study of the learned temporal kernel functions of is_complement_of (left) and is_substitute_of (right) relation. Each line represents that when a corresponding relational item is consumed at the origin point, how its effect on items of this category changes with time.

already have a headset or have purchased one from somewhere else. Therefore, the positive effect of complement is expected to decay quickly, otherwise persistent recommendation for headsets may be a bother to the user. On the other hand, for replacement parts like backup batteries, the positive effect of complementary consumption will last for a while. Because users often purchase a backup battery after a period of time when the original device's battery runs out.

The right figure shows the impact of *is_substitute_of* relation for *Basic Cases*, *International Chargers*, and *Cellphones*. Despite the general forms all consist of two opposite normal distributions, the temporal kernel function for basic cases is quite different from the other two, where the component for the restraining effect is almost flatten. This indicates that when users buy a case, there will not exist strong negative effects because we often change phone cases for various reasons, such as a broken edge or just wanting to try a new style. For international chargers and cellphones, their temporal kernel functions both demonstrate obvious negative effects and positive peaks. It is reasonable because we generally do not need another charger or cellphone if we have just bought one. Interestingly, the time gaps corresponding to the peak of these two kinds of items are similar, which reflects the fact that the change of cellphone often leads to the change of matching chargers. Moreover, compared to chargers, the curve is smoother for cellphones. The reason may be that we could change our cellphones for various causes but seldom change chargers if one works. Thus a new cellphone can be consumed after a wide range of time gaps.

To sum up, the time-related parameters in our Chorus model are highly interpretable and well reflect the characteristics of items in different categories. These parameters can help recommender systems to give explanations to the recommendation results. For example, when a user bought an iPhone previously and it is time when the temporal kernel function for cellphones peaks, the recommendation for a new cellphone can be explained as "Your cellphone has been on service for a long time, how about having a look at some new products?"

6 CONCLUSION AND FUTURE WORK

In this work, we propose a novel method Chorus for knowledge- and time-aware item modeling. To the best of our knowledge, we are the first to explicitly model the evolution of different relations' effects with time, and incorporate such information into item embeddings. We use graph embedding to learn structural information from item relation graph, and then derive different relational representations for each item. The temporal dynamics of relations are controlled by specifically designed temporal kernel functions, and the relational representations are dynamically combined based on time gaps between the target item and relational items in the history sequence, leading to a knowledge-aware dynamic item representation. Besides, the design of temporal kernel function can be seen as a kind of human intervention to the model and can be used to meet different requirements on the recommendation results. Due to the flexibility of Chorus, it can easily leverage various embedding-based algorithms to calculate ranking scores and make recommendations. Extensive experimental results show that Chorus is superior to state-of-the-art baselines, which demonstrates that both item relations and their temporal evolutionary effects are of great importance. Moreover, the time-related parameters in Chorus are highly interpretable and hence can help to enhance the explainability of recommendation.

This model still has some limitations, such as predefined temporal functions and the two-stage learning proces. Besides, although translation-based methods generally work well, we find they are insufficient in some cases, which need to be further studied. In the future, we will investigate how to adaptively estimate the temporal evolutionary effects of different relations, and try to design more suitable methods to tightly integrate item relation modeling and recommendation.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2018YFC0831900) and Natural Science Foundation of China (Grant No. 61672311, 61532011). Dr Weizhi Ma has been supported by Shuimu Tsinghua Scholar Program. We would like to thank Maarten de Rijke for his valuable comments of this work.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering* 6 (2005), 734–749.

[2] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. 2015. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 130–140.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 335–344.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1555–1564.

[7] Nan Du, Yichen Wang, Niao He, and Le Song. 2015. Time-sensitive recommendation from recurrent user activities. In *International Conference on Neural Information Processing Systems*. 3492–3500.

[8] Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58, 1 (1971), 83–90.

[9] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 507–517.

[10] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[15] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 659–667.

[16] Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2018. Recommendation Through Mixtures of Heterogeneous Item Relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1143–1152.

[17] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM conference on Recommender systems*. ACM, 79–86.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Computer Science* (2014).

[19] Yehuda Koren. 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*. ACM, 426–434.

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

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[22] Takeshi Kurashima, Tim Althoff, and Jure Leskovec. 2018. Modeling Interdependent and Periodic Real-World Action Sequences. *arXiv preprint arXiv:1802.09148* (2018).

[23] Guokun Lai, Wei Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. (2018).

[24] Sha Li, Xiaofeng Gao, Weiming Bao, and Guihai Chen. 2017. FM-Hawkes: A Hawkes Process Based Approach for Modeling Online Activity Correlations. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1119–1128.

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[26] Juntao Liu and Caihua Wu. 2017. Deep learning based recommendation: a survey. In *International Conference on Information Science and Applications*. Springer, 451–458.

[27] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling user session and intent with an attention-based encoder-decoder architecture. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 147–151.

[28] Weizhi Ma, Min Zhang, Yue Cao, Chenyang Wang, Yiqun Liu, Shaoping Ma, Xiang Ren, et al. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. *arXiv preprint arXiv:1903.03714* (2019).

[29] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 785–794.

[30] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. 2017. Do Also-Viewed Products Help User Rating Prediction?. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1113–1122.

[31] Rajiv Pasricha and Julian McAuley. 2018. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 63–71.

[32] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David MJ Tax. 2017. Interacting attention-gated recurrent networks for recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management*. ACM, 1459–1468.

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[34] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[35] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1 (2001), 285–295.

[36] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[37] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

[38] Elena Smirnova and Flavian Vasile. 2017. Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*. ACM, 2–9.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[40] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1133–1142.

[41] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling Item-Specific Temporal Dynamics of Repeat Consumption for Recommender Systems. In *The World Wide Web Conference*. ACM, 1977–1987.

[42] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[43] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[44] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon Jose. 2019. Relational Collaborative Filtering: Modeling Multiple Item Relations for Recommendation. *arXiv preprint arXiv:1904.12796* (2019).

[45] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).