Toward Dynamic User Intention: Temporal Evolutionary Effects of Item Relations in Sequential Recommendation

CHENYANG WANG, WEIZHI MA, MIN ZHANG, CHONG CHEN, YIQUN LIU, and SHAOPING MA, Tsinghua University

User intention is an important factor to be considered for recommender systems, which always changes dynamically in different contexts. Recent studies (represented by sequential recommendation) begin to focus on predicting what users want beyond what users like, which are better at capturing user intention and have attracted a surge of interest. However, user intention modeling is non-trivial, because it is generally influenced by various factors, among which item relations and their temporal evolutionary effects are of great importance. For example, consumption of a cellphone will have varying impacts on the demands for its relational items: For complements, the demands are likely to be promoted in the short term; while for substitutes, the long-term effect may take advantage, because users do not need another cellphone immediately. Moreover, the temporal evolutions of different relational effects vary across different domains, which makes it challenging to adaptively take them into consideration. As a result, most existing studies only loosely incorporate item relations by encoding their semantics into embeddings, neglecting fine-grained time-aware effects.

In this work, we propose Knowledge-aware Dynamic Attention (KDA) to take both relational effects and their temporal evolutions into consideration. Specifically, to model dynamic impacts of historical relational interactions on user intention, we aggregate the history sequence into relation-specific embeddings, where the attention weight consists of two parts. First, we measure the relational intensity between historical items and the target item to model the absolute degree of influence in terms of each relation. Second, to model how the relational effects drift with time, we innovatively introduce Fourier transform with learnable frequency-domain embeddings to estimate temporal decay functions of different relations adaptively. Subsequently, the self-attention mechanism is leveraged to derive the final representation of the whole history sequence, which reflects the dynamic user intention and will be applied to generate the recommendation list. Extensive experiments in three real-world datasets indicate the proposed KDA model significantly outperforms the state-of-the-art methods on the Top-*K* recommendation task. Moreover, the proposed Fourier-based method opens up a new avenue to adaptively integrate temporal dynamics into general neural models.

CCS Concepts: • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: Sequential recommendation, dynamic user intention, knowledge enhanced model, time-aware model, Fourier transform, attention mechanism

This work is supported by the National Key Research and Development Program of China (2018YFC0831900), Natural Science Foundation of China (Grant No. 62002191, 61672311, 61532011), Tsinghua University Guoqiang Research Institute, China Postdoctoral Science Foundation (2020M670339) and Dr. Weizhi Ma has been supported by Shuimu Tsinghua Scholar Program. This paper is based upon work supported by an IBM Global Academic Award.

Authors' addresses: C. Wang, W. Ma, M. Zhang (corresponding author), C. Chen, Y. Liu, S. Ma, Tsinghua University, 30 Shuangqing Road, Haidian District, Beijing, China, 10084; emails: wangcy18@mails.tsinghua.edu.cn, mawz@tsinghua.edu.cn, z-m@tsinghua.edu.cn, cc17@mails.tsinghua.edu.cn, yiqunliu@tsinghua.edu.cn, msp@tsinghua.edu.cn.

1046-8188/2020/12-ART16 $15.00
https://doi.org/10.1145/3432244

ACM Reference format:
Chenyang Wang, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2020. Toward Dynamic User Intention: Temporal Evolutionary Effects of Item Relations in Sequential Recommendation. *ACM Trans. Inf. Syst.* 39, 2, Article 16 (December 2020), 33 pages.
https://doi.org/10.1145/3432244

1 INTRODUCTION

Recommender systems have been widely applied to various web services to address the issue of information overload, such as e-commerce platforms, news feeds, social media applications, and so on. It not only can largely reduce users' efforts in finding things of interest but also bring more profits to the platforms [8, 53]. Traditional studies about recommender systems [10, 26, 33, 35, 47] mainly concentrate on predicting intrinsic user preference, which is consistent as time goes by. For example, the typical latent factor models [35, 47] embed both users and items into the same latent space. The embeddings of users represent their preferences in various aspects, which will not change when making recommendations in different contexts, while in real-world scenarios, user intention is also an influential factor beyond user preference, which is influenced by recent interactions. The same item can have different meanings to a user in different contexts. Thus, more and more recommender systems begin to focus on not only what users like (user preference) but also what users want (user intention).

Sequential recommendation algorithms [16], which aim to predict a user's next action based on the context of recent interactions, are good at capturing user recent intention and recently have been gaining growing interest. Representative methods include Markov Chain–based approaches [19, 20, 48] and **Recurrent Neural Network (RNN)** [12] based models [24, 25, 51]. Inspired by Transformer [53] in machine translation, attention-based methods also begin to emerge in recent years [30, 38, 45]. There is also research aiming to understand user intention by incorporating extra information, such as item relations and timestamps. Knowledge-enhanced recommendation introduces different relations between items (e.g., complements and substitutes), which contribute to better recommendation performance [41, 55, 60]. Time-aware recommendation takes the timestamp of each interaction into consideration, which is capable of modeling the drift of user intention. Some studies directly utilize timestamp as an additional context feature [44], and some others try to use continuous functions to model the temporal evolution [56].

However, when considering the effects of previous relational interactions on user intention, we find the relational intensity and adaptive temporal evolution are seldom taken into consideration in current methods. The relational intensity refers to the intensity that there exists a specific relation between interacted items and the target item. For example, AirPods are generally perceived to be a complement of iPhone (high relational intensity), and use of an iPhone will greatly promote the demand for AirPods. With regard to consumption of a HUAWEI phone, it will also have positive effects on AirPods (even if the relation between them could be missing in the knowledge base). But such positive effects are tiny, since the relational intensity between them is comparatively low. In addition, the effects of previous relational interactions change dynamically with time, and the temporal evolution can differ from each other for different relation types. For example, after buying a cellphone, the demands for accessories are in the short term. Such positive effects will decay quickly, since the user may already have these complementary items, while for substitutes, the long-term effect may become more prominent, because users do not tend to consume substitutes immediately. Therefore, relational intensity and adaptive temporal evolution are of great importance to understand dynamic user intention.

Although there are previous studies that considers both item relations and corresponding temporal dynamics [55, 56], there are two main weaknesses: (1) they do not take relational intensity into consideration and rigidly determine whether there exist relations between items, which ignore potentially missing relations in the relation graph. (2) Previous studies cannot adaptively estimate temporal evolutions in distinct domains but rely on manually defined functions, which makes them face challenge to be directly applied to real-world scenarios with different temporal dynamics. For example, SLRC [56] only considers repeat consumption as a relation and uses a combination of exponential and normal distributions to control the time-sensitive effects. Chorus [55] further incorporates various item relations but ignores relational intensities. The temporal decay functions also need to be predefined by prior knowledge.

To model relational intensity and achieve adaptive estimation of temporal evolution, there are mainly two challenges: (1) Though existing datasets or knowledge graphs contain some item relations, the exact relational intensities of seen and unseen item pairs are unknown. (2) The temporal evolutionary effects of different item relations vary dramatically in different scenarios, which needs domain-free approaches to capture the dynamic effects automatically. To deal with these challenges, we present **Knowledge-aware Dynamic Attention (KDA)** to achieve a better understanding of dynamic user intention. The main idea of KDA is to adaptively aggregate the history sequence into multiple relation-specific embeddings (named relational dynamic history embedding). The attention weights depend on both relational intensities and time intervals between interacted items and the target item. Then user intention is captured by modeling the mutual influence of these relational dynamic history embeddings.

First, we model item–item relations and their intensities via a knowledge graph embedding task. The matching scores of relational item pairs are maximized to encode semantic meanings of item relations into embeddings. Then, the score function will be used to measure the relational intensities between interacted items and the target item, which subsequently determine partial attention weights of historical items. More importantly, we innovatively introduce Fourier transform to model the varying temporal effects of different relational interactions. In particular, based on our empirical study, we assume that for each relation type in the dataset, there is a latent and continuous temporal decay function of time intervals, which determines the temporal evolution of relational effects on the target item. Then, we parameterize the frequency domain representation of each temporal decay function via **Discrete Fourier Transform (DFT)**, called frequency embeddings. In this way, the temporal decay functions can be estimated adaptively without predefined functional forms. The function values of unseen time intervals will be easily obtained by applying **Inverse Discrete Fourier Transform (IDFT)** on corresponding frequency embeddings, which provides a general approach to incorporate latent and continuous temporal effects in neural models.

In addition to the design of knowledge-aware dynamic attention, we use self-attention and pooling layer to mutually absorb the information captured in each relational dynamic history embedding. As a result, the final history representation well reflects the dynamic user intention regard for the target item. Finally, we optimize the recommendation task and the knowledge graph embedding task with a joint-learning framework. Extensive experiments in three real-world public datasets demonstrate the effectiveness of our proposed model. The consistent improvements in different datasets indicate that the adaptive modeling of item relations and their temporal evolutionary effects in KDA can scale to different application scenarios.

Our main contributions are summarized as follows:

- Based on the empirical study in three datasets with different recommendation scenarios, we reveal that the effects of previous relational interactions have different temporal evolutions

across datasets and relations, which is important for better understanding dynamic user intention in the sequential recommendation.
- KDA is presented to model item relations and their temporal evolutionary effects. We devise relational intensity and frequency-domain embeddings to adaptively determine the importance of historical interactions. The Fourier-based estimation of temporal decay functions further opens up a new avenue to integrate temporal dynamics into neural models.
- Comparative experiments are performed in three real-world datasets, covering different application scenarios and interaction densities. The results demonstrate that KDA significantly and consistently outperforms state-of-the-art methods by more than 7.1%, 13.6%, and 2.4% in the three datasets, respectively.

The rest of this article is organized as follows. In the next section, some related work is reviewed. In Section 3, we introduce the preliminaries about this work. Then we conduct an empirical study to analyses the temporal evolution of relational effects in Section 4. Next our Knowledge-aware Dynamic Attention model is elaborated in Section 5. In Section 6, we present the experimental results and corresponding analyses. Subsequently, some limitations and future directions are discussed in Section 7. We conclude this work in Section 8.

2 RELATED WORK

In this section, some classical methods of sequential recommendation are introduced first. Then, we review two lines of work relevant to ours, namely knowledge-enhanced recommendation and time-aware recommendation.

2.1 Sequential Recommendation

Different from traditional recommendation methods [9, 23, 47], sequential recommender systems leverage the user's historical interaction sequence to better capture current user intention. Some models view the sequential data as Markov chains, which assume that the next action depends on previous action sequence [48, 50]. For example, FPMC [48] combines **Matrix Factorization (MF)** [35] and factorized Markov Chains to make next-basket recommendation given the previous basket items. FISM [29] and SVD++ [33] use the summation of historical item embeddings to represent the user. The main ideas behind these methods are similar, they focus on capturing item-item transition matrices to predict the next item.

More recently, with the development of deep learning, there has been a lot of work utilizing RNN or **Convolution Neural Network (CNN)** to encode interaction history to hidden vectors [14, 25, 36, 45, 51, 64]. GRU4Rec [25] first introduces RNN to the sequential recommendation and achieves impressive performance improvements. NARM [37] further applies attention mechanism [53] to RNN for more effective recommendation. NextItNet [64] utilizes holed CNN layers to increase the receptive fields and model long-range dependencies. Besides, the attention mechanism, which is shown to be effective in various tasks [3, 61], have also been used in the sequential recommendation. SASRec [30] applies self-attention to model the mutual influence between historical items, achieving state-of-the-art results. MARank [63] considers the most recent items and applies multi-order attention to capture individual- and union-level item dependency. Despite the great expressiveness of deep-learning sequential recommendation methods, they still cannot well model sophisticated user intention for the lack of external knowledge, and most of them substantially suffer from the interpretability issue [40].

2.2 Knowledge-enhanced Recommendation

In real-world scenarios, there are multiple relations between items that have concrete semantics. Recent work has been focusing on introducing item relations into recommender systems [31, 41, 43, 54, 60, 66], most of which leverage **Knowledge Graph (KG)** [57] to represent relations between different items.

On the one hand, some work introduces users into KG to build a user-item graph. CFKG [66] views the action *buy* as another relation and then uses TransE [5] to represent the heterogeneous information network and make recommendations. However, most studies focus on modeling the item-item graph solely as a supplement to the recommendation task. In this line of work, there are mainly two kinds of methods: path-based and embedding-based method. The path-based method introduces meta-path to extract knowledge from KG, which is more explicable and is popular in the beginning. For instance, Ma et al. [41] propose a joint learning framework to integrate the induction of explainable rules from KG. The embedding-based method gains more attractions in recent years, which leverages KG embedding methods to obtain embeddings in the recommendation task. Some representative studies include CKE [65] and KSR [27]. Recently, RCF [60] is proposed as a joint-learning framework that incorporates multiple relations between items, achieving remarkable performance improvements. But item embedding is generally the only connection between recommendation task and KG embedding task in these studies, which ignore relational intensities between items and yield loose incorporation of item relations.

2.3 Time-aware Recommendation

Users' preference and items' popularity may change dynamically, which is known as concept drift [59]. Time-aware recommendation tries to incorporate temporal information (e.g., timestamps) to model dynamic user intention. There are typically two lines of work taking temporal information into consideration. On the one hand, some studies view time as context information, aiming to model user preference at different times. TimeSVD++ [34] partitions timelines into periods and designs time-related parameters on the basis of SVD++. Tensor factorization is also a major approach in this line of work [4, 31], where discretized time is viewed as the third dimension of user-item interaction cube. Some other work utilizes content-based models like **Factorization Machine (FM)** to include the timestamp as an extra context feature [22, 44]. On the other hand, some studies focus on modeling temporal decay effects of historical interactions. Intuitively, interactions that occur long ago are less important than recent ones and should be underweighted somehow. Ding et al. [13] first introduces a user-specific and item-specific time weighting scheme for a similarity-based collaborative filtering approach. Sanchez et al. [49] further improve the similarity metric between users with a temporal decay function and sequential information. Some other methods introduce Hawkes Process [17] to the recommendation and model temporal decay with the kernel function [14, 15, 56]. SLRC [56] combines Hawkes Process and collaborative filtering to model temporal dynamics of repeat consumption. The short-term and life-time effects are modeled with exponential and normal distributions, respectively. TiSASRec [38] encodes time intervals into embeddings based on self-attention mechanism. Most recently, Chorus [55] proposes to model the temporal effects of different relations with specifically designed temporal kernel functions.

However, either of these two lines of work has its own drawbacks. Taking time as context information naturally faces the challenge of predicting future behavior at unseen periods. Although estimating temporal effects is more applicable, most of these models need a predefined functional form of the temporal decay, which makes it inflexible to capture different temporal dynamics in different domains. As a result, we introduce Fourier transform to adaptively learn temporal decay functions, which greatly enlarges the application scope of the proposed model.

Table 1. Notations

Notation	Description
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
$\mathcal{R}$	the set of item relation triplets
S_u	the interaction sequence of user u
M	the number of relation types
N	the point of DFT
$\mathbf{p}_u \in \mathbb{R}^d$	the embedding for user u
$\mathbf{q}_i \in \mathbb{R}^d$	the embedding for item i
$\mathbf{x}_\tau \in \mathbb{R}^d$	the embedding for relation type τ
$\mathbf{z}_v \in \mathbb{R}^d$	the embedding for relation value v
$\mathbf{s}_{i,\tau} \in \mathbb{R}^d$	the relational dynamic history embedding in terms of the target item i and relation type τ
$\mathbf{m}_{u,i} \in \mathbb{R}^d$	the final KDA history representation of user u in terms of the target item i
$f_\tau(\cdot)$	the temporal decay function for relation type τ
$\mathbf{F}_\tau \in \mathbb{C}^N$	the complex frequency embedding of $f_\tau(\cdot)$

3 PRELIMINARIES

In this section, we first formulate the time-aware sequential recommendation problem with item relations and introduce some definitions. Then we describe the datasets we use for the empirical study and experiments. Our main notations are summarized in Table 1.

3.1 Problem Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ represent the user and item set, respectively. For each user $u \in \mathcal{U}$, we are given the interaction history $S_u = \{(i_1, t_1), (i_2, t_2), \ldots, (i_{N_u}, t_{N_u})\}$, and each element contains the interacted item $i \in \mathcal{I}$ and the timestamp $t \in \mathbb{N}^+$ ($t_n \leq t_{n'}$ for any $1 \leq n < n' \leq N_u$). Besides, a set of relational triplets $\mathcal{R}$ is also given, representing the relations between items, where each element is an item pair (i, j) and their relation r, denoted as (i, j, r). Then our task is as follows: Considering the interaction sequence before the target time T, denoted as S_u^T, as well as the item relations set $\mathcal{R}$, generating an ordered list that the user may be interested in at T.

With regard to the external knowledge $\mathcal{R}$, there are mainly two kinds of relations between items: (1) *natural item relations* and (2) *attribute-based item relations*. The former naturally exists in the datasets in general, such as *also_buy* and *also_view*. While the latter is extracted from item metadata. Any item pair with the same discrete value of an attribute can be seen to have a relation between them, such as *shared_brand* and *similar_price*. In the meantime, compared to natural item relations, attribute-based item relations usually have more detailed semantics (e.g., the concrete brand *Apple* that the two items share). To capture such fine-grained meanings of a relation, we follow the setting in previous study [60] and define the item relation with a concept of two-level hierarchy $r =< \tau, v >$, where τ denotes the *relation type* and v is the *relation value*.

Definition 3.1 (Relation Type). The relation type describes how items are related to each other in an abstract way, such as *shared_brand*, *similar_price*, *also_buy*, and so on.

Definition 3.2 (Relation Value). The relation value provides details or attributes of the relation, such as *Apple* for the relation type *shared_brand*.

Note that the relation value may be not applicable for some item relations (e.g., *also_buy*, *also_view*), because they do not have related details. For these relation types without fine-grained values, we set their relation values to a special value *None*. In this way, all kinds of item relations can be represented with a tuple $< \tau, \upsilon >$.

With this two-level hierarchy, relations with the same type will keep similar representations, and the relation values empower the representation fidelity of different relations. It is beneficial to capture fine-grained user preference, since a user could weigh different values of a relation type differently. For example, for the same relation type *shared_brand*, a user may be more loyal to *Apple* than other relation values. More discussions and limitations of this representation method will be presented in Section 6.4 and Section 7.

To facilitate the understanding of our empirical study, we additionally define the *relational neighbor* of a given interaction in the user sequence, which refers to the most recent relational historical interaction for a specific relation r.

Definition 3.3 (Relational Neighbor). For relation r and a user's interaction sequence S_u, $(i, t) \in S_u$ is a relational neighbor of the target interaction $(i^*, t^*) \in S_u$ if and only if $t < t^*$ and $(i, i^*, r) \in \mathcal{R}$, meanwhile there is no interaction $(i', t') \in S_u$ satisfying $t < t' < t^*$ and $(i', i^*, r) \in \mathcal{R}$.

3.2 Dataset Description

We choose three real-world public datasets with various scenarios and different densities. The basic information of these datasets is described below.

- **MovieLens-100k** [1]: This is a widely adopted benchmark dataset for evaluating recommendation algorithms, which contains users' ratings of movies as well as side information about users and items. We use the MovieLens-100k dataset and extract item relations based on the year and genre of movies.
- **Amazon Electronics** [2] [21, 42]: This is an e-commerce dataset containing a large corpus of product ratings, reviews, and metadata, collected from Amazon.com from May 1996 to July 2014. It includes natural item relations such as *also_buy*, *also_view*, *buy_together*. And we facilitate the brand, price, and top-level category attributes of items.
- **RecSys2017** [3]: RecSys Challenge 2017 is a competition for job recommendation, whose dataset includes users' interactions with different job postings on XING.com. We use the click records in the offline phase of the competition as user-item interaction data. The career level, discipline, industry, and region of job postings are utilized to generate item relations.

We follow the preprocessing procedure of previous work [19, 60]. For user-item interactions data, we treat a rating or a click as implicit feedback (interaction without explicit rating score), and order interactions by timestamps. Only the last one of duplicate interactions is retained and others are removed. Items with fewer than five interactions will be filtered out.

For item-item relations data, item attributes with continuous values (only price in our datasets) will be discretized into five bins, and the pivots are determined by quantiles of the overall value distribution. For items and relation values that only appear once in the relation set $\mathcal{R}$, the related triplets will be removed. The statistics of these datasets after preprocessing are listed in Table 2.

In summary, the chosen datasets cover three different recommendation scenarios. MovieLens-100k contains fewer interactions and is the densest dataset, while the other two datasets are on a larger scale and much more sparse.

[1] http://grouplens.org/datasets/movielens/100k/.
[2] http://jmcauley.ucsd.edu/data/amazon/.
[3] http://www.recsyschallenge.com/2017/.

Table 2. Dataset Statistics

	Dataset	MovieLens-100k	Amazon Electronics	RecSys2017
User-Item Interactions	#user	943	192,403	382,097
	#item	1,349	63,001	38,496
	#interaction	99,287	1,689,188	1,381,829
	density	7.805%	0.014%	0.009%
Item-Item Relations	#type	2	6	4
	#value	34	1,888	67
	#triplet	886K	2,844M	1,285M

4 EMPIRICAL STUDY

Item relations have been shown to be useful in recommendation algorithms [7, 60], but there are few studies investigating the temporal evolutionary effects of previous relational interactions. In this section, we make several observations about the temporal evolutionary effects of item relations in three real-world datasets, which serve as the foundation of our model design.

Intuitively, recent interactions will have greater impacts on the user intention, which is also known as *recency* effect [2]. For instance, after purchasing a cellphone, the demands for complementary items (e.g., cases, headphones) mainly center in the short term. The positive effects will decay quickly, since users could already have these complementary items, in which case the recommender system should not present these recommendations consistently. However, the effects of some relations may be promoted after a period, such as the substitute relation. Users' demands for substitute items will rise when the lifetime of the previous one runs away.

As a result, here we want to verify how the effects of historical relational interactions vary with time in real-world datasets. Considering that it is hard to directly observe the temporal effects of relational interactions, we resort to study the distribution of time intervals between interactions in the user sequence and their relational neighbors (the most recent relational interaction according to Definition 3.3). Actually, this distribution is a good indicator of temporal evolutionary effects. For example, if (cellphone, cases, *also_buy*) holds in $\mathcal{R}$, and a lot of users purchase cases one day after the previous consumption of a cellphone, the effects in terms of *also_buy* may also have a spike when the time interval is one day.

Due to the fact that time intervals generally demonstrate long-tail distributions, we normalize the time interval Δt (in seconds) with a log transformation:

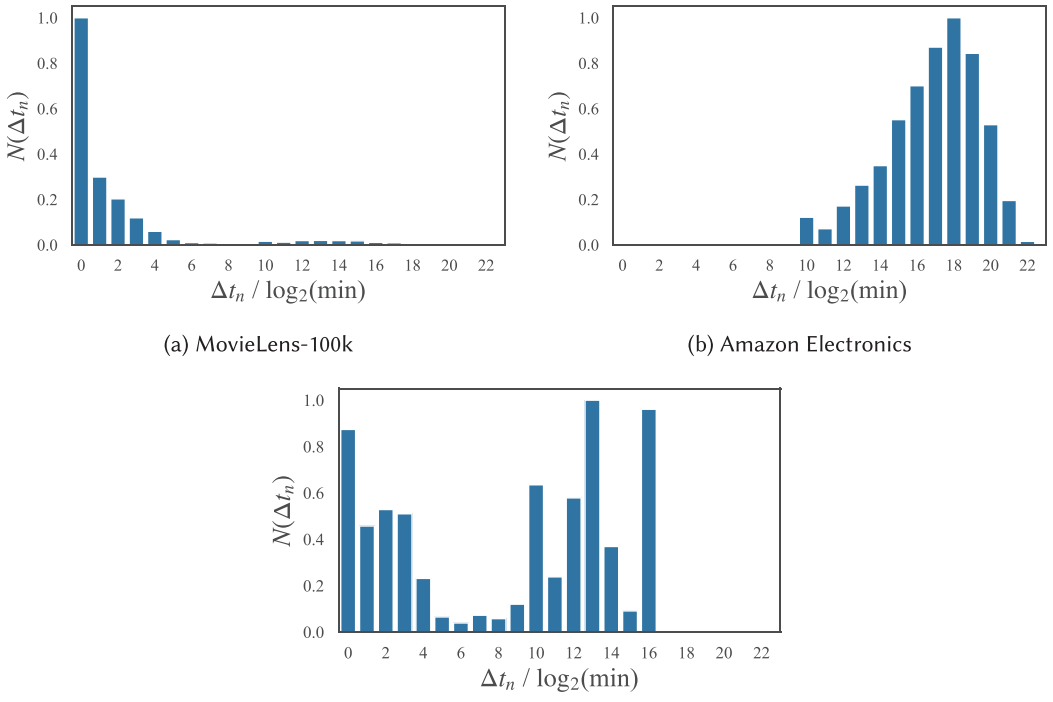
$$\Delta t_n = \max(0, \log_2(\Delta t/60)). \tag{1}$$

All the following time intervals are normalized in the same way. Then, for each relation r in the dataset, we can traverse all the interactions in each user sequence to find their relational neighbors and derive corresponding time interval distribution.

4.1 Temporal Evolution in Different Domains

First, we focus on the global time interval distribution in each dataset, which is derived by integrating the time interval distributions of all the relations in this dataset. The global time interval distribution reflects the characteristics of different recommendation scenarios. Figure 1 shows the distribution[4] in each dataset. We can see the temporal evolution of relational effects in different domains varies dramatically.

[4]The x-axis represents normalized time intervals, and the y-axis values are divided by the maximum number in the dataset.



(a) MovieLens-100k

(b) Amazon Electronics

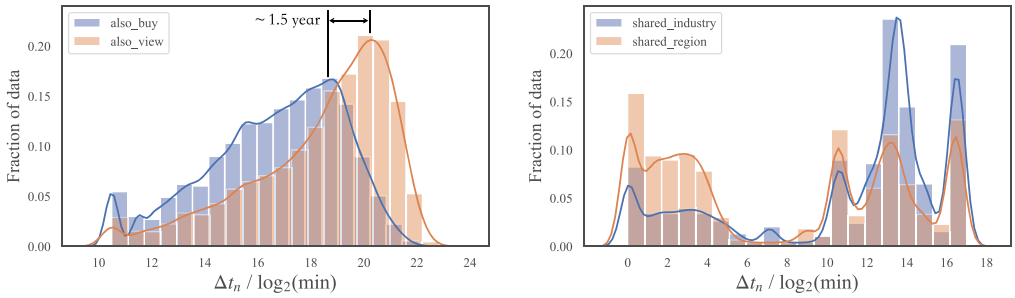
(c) RecSys2017

Fig. 1. The global distribution of time intervals between interactions and their relational neighbors in each dataset. The overall temporal evolution varies dramatically across different domains.

In MovieLens-100k, the time interval distribution exhibits an exponential-decay form, which indicates that users usually watch relational movies in a continuous way. Most users may just browse a series of movies with the same genre/year to examine their interests. This explains the high fraction of time intervals in minutes (the first few bins). Besides, after vanishing in the mid term, a few cases appear around the 13th bin (corresponding time interval is about one week), which shows users also tend to find some similar movies after a week's work. In the context of e-commerce (Amazon Electronics), the time interval distribution looks like a skewed normal distribution (the mean is corresponding to about half a year). It is reasonable, because users seldom consume relational electronic items continuously in the short term. As for the scenario of browsing job postings in RecSys2017, the distribution demonstrates a more complex pattern. There are several spikes in the short, mid, and long term. On the one hand, users may click similar job postings in a single session, especially for jobs with the same career level or region. On the other hand, users could tend to recap or compare with previously clicked postings in the mid term. And browsing relevant postings may be needed after a series of interviews in the long term.

4.2 Temporal Evolution for Different Relation Types

Apart from the obvious differences between datasets, we also investigate the time interval distribution across different relations. Considering that time interval distribution of each relation may have few supports, we integrate distributions of relations with the same relation type (regardless of relation values) to investigate characteristics of different relation types. Figure 2 shows the distributions of some representative relation types in Amazon Electronics and RecSys2017.



(a) Amazon Electronics (b) RecSys2017

Fig. 2. The distribution of time intervals between interactions and their relational neighbors in terms of different relation types. Although the overall trends in the same dataset are similar, the concrete distribution of each relation type differs from each other, which relies on the characteristics of the relation type.

For relation types *also_buy* and *also_view* in Amazon Electronics, although the overall trends are similar, the negative skewness (left tail) of *also_view* is stronger. Note that the time intervals in x-axis are under log transformation. The time interval corresponding to the peak of *also_buy* lies in the range from half a year to a year; while *also_view* centers around the 20th bin (around 2 years). So the realistic time difference between the peaks of these two relation types is about 1.5 year, which demonstrates obviously different temporal patterns. Literally, these two relation types reflect the functionality between items to some extent. For example, if $(i, j,$ *also_buy*$)$ holds, j is likely to be a complement of i; while $(i, j,$ *also_view*$)$ probably indicates j is a substitute of i. Therefore, the difference of distribution skewness between the two relation types conveys the message that the demands for substitutes rise later compared with complements, which is reasonable, because users seldom purchase substitutes immediately.

In RecSys2017, the difference also exists between *shared_industry* and *shared_region*. The short-term effects are much stronger for *shared_region*, because users are usually interested in jobs with the same region in a single attempt but barely deliver several resumes to the same industry. Meanwhile, the temporal effects for *shared_industry* take advantage in the long term. At this time, users may need to compare other jobs in this industry given the new application status.

4.3 Concluding Remarks

In summary, previous relational interactions indeed have time-related impacts on user intention toward the target item. Besides, the temporal evolution of the effects varies dramatically across different domains and even relation types. Therefore, to better understand dynamic user intention, it is essential to adaptively take both item relations and their temporal evolutionary effects into consideration, which calls for a domain-free approach to capture characteristics of different relation types in various scenarios.

5 KNOWLEDGE-AWARE DYNAMIC ATTENTION MODEL

In this section, we elaborate our Knowledge-aware Dynamic Attention model, which mainly consists of two parts: (1) *Item Relation Modeling* and (2) *User Intention Modeling*. Figure 3 demonstrates the overall structure of our KDA model.

First, we organize the relational data between items as a knowledge graph and learn a graph embedding task in *Item Relation Modeling*. This module primarily aims to encode semantics of item relations into embeddings, so that the score function can measure relational intensities between items. Second, to capture various effects of historical relational items, we aggregate history

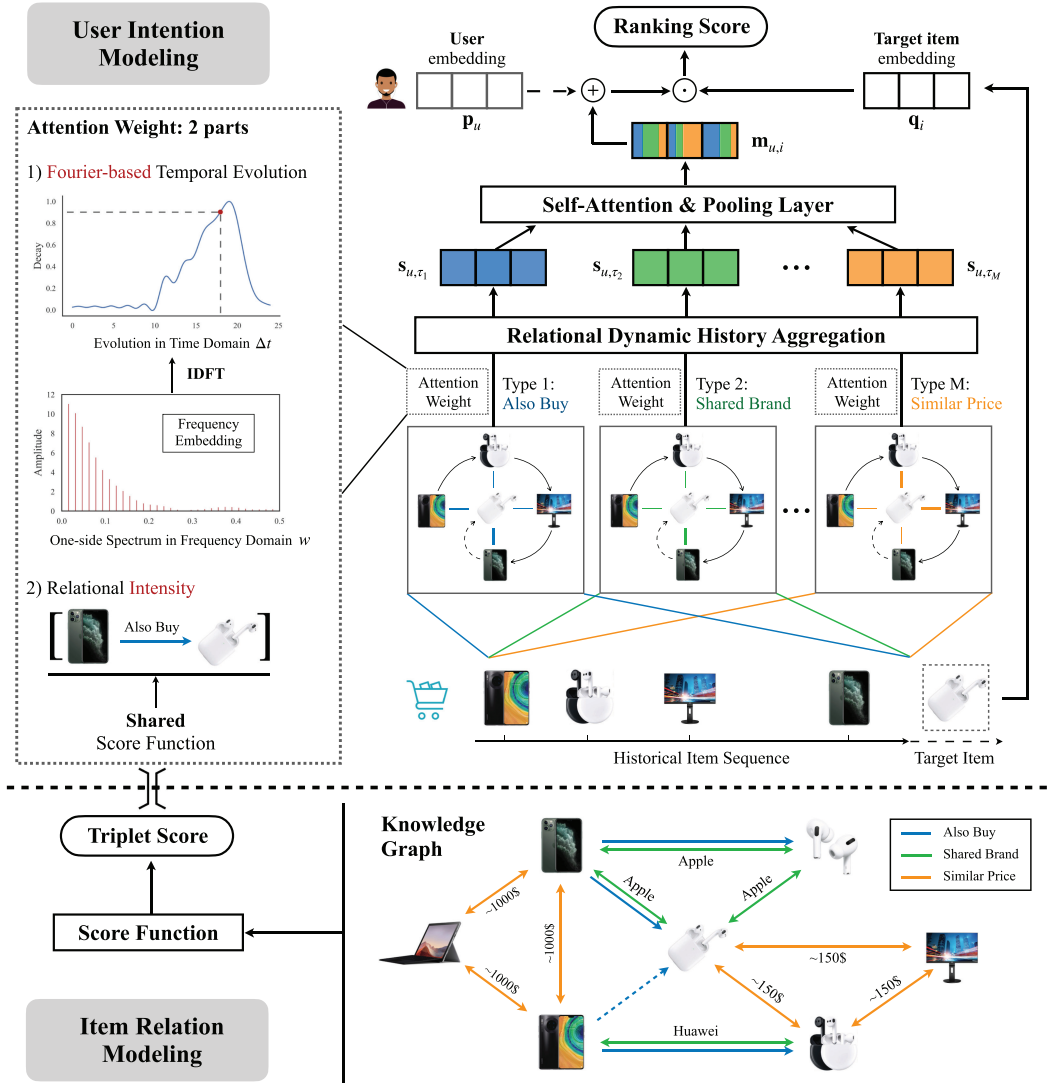


Fig. 3. Overall structure of KDA. There are mainly two modules: (1) *Item Relation Modeling* learns a knowledge graph embedding task to keep semantics of item relations into embeddings; (2) *User Intention Modeling* derives several relation-specific history embeddings, where we devise relational intensity and Fourier-based temporal evolution to determine the dynamic effects of historical interactions. Then self-attention is utilized to capture the mutual influence of different history embeddings and derive the final KDA history representation, which will be used to generate the ranking list.

sequence under the view of different relation types in *User Intention Modeling*. For each relation type, there will be a relational dynamic history embedding and the attentional aggregation weight consists of two parts: (1) relational intensity and (2) Fourier-based temporal evolution. Finally, the self-attention mechanism is adopted to model the mutual influence of different history embeddings, and the final KDA history representation is viewed as a part of the user representation to generate the recommendation list.

To make it easier to understand our KDA model, we first describe the overall structure of *User Intention Modeling* in Section 5.1. As relational dynamic history aggregation is a vital part in *User Intention Modeling* and also the main contribution of this study, we introduce it in detail in Section 5.2. The methods used in *Item Relation Modeling* are introduced in Section 5.3. Next, we describe the multi-task learning strategy and analyze model complexity in Section 5.4. The characteristics of our KDA model and some connections to existing methods are discussed in Section 5.5.

5.1 User Intention Modeling

Given a target item, user intention is largely influenced by his/her recent interaction sequence, especially historical items that are relational to the target item. Besides, for different relations, the degrees and temporal evolution of the effects also differ from each other. Thus, it is important to understand users' perceptions of the history sequence under the view of different relation types, as well as the mutual influence of these relations. In this module, we will introduce how we integrate these factors to calculate the final ranking score.

The basic idea of the KDA model is to derive a history representation $\mathbf{m}_{u,i} \in \mathbb{R}^d$ given the history sequence S_u^T of user u and the target item i, which integrates relational and dynamic effects of historical interactions. Then the final ranking score $\hat{y}_{u,i}$ is calculated as follows:

$$\hat{y}_{u,i} = (\mathbf{p}_u + \mathbf{m}_{u,i})\mathbf{q}_i^T + b_i, \tag{2}$$

where $\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^d$ are embeddings for the user and item, respectively, and b_i is the bias term for each item. The user embedding $\mathbf{p}_u$ reflects the long-term preference, and $\mathbf{m}_{u,i}$ encodes short-term user intention. Subsequently, we focus on how to obtain the KDA history representation $\mathbf{m}_{u,i}$.

To better understand dynamic user intention, a two-level aggregation architecture is devised: We first dynamically aggregate the embeddings of interacted items under the view of different relation types and then move forward to aggregate these relation-specific history embeddings $\mathbf{s}_{i,\tau}$ to obtain the final KDA history representation $\mathbf{m}_{u,i}$.

5.1.1 Relational Dynamic History Aggregation. First, we focus on the relational dynamic history aggregation in terms of each relation type (first-level aggregation). Specifically, for the given history sequence S_u^T and the target item i, we derive a *relational dynamic history embedding* $\mathbf{s}_{i,\tau}$ for each relation type τ as follows:

$$\mathbf{s}_{i,\tau} = \sum_{(j,t_j) \in S_u^T} \text{INT}(i,j,\tau) f_\tau(\Delta t_n) \cdot \mathbf{q}_j, \tag{3}$$

where $\mathbf{q}_j$ is the embedding of the historical item, and the attention weight consists of a relational intensity score $\text{INT}(i,j,\tau) \in [0,1]$ and a temporal decay $f_\tau(\Delta t_n) \in [0,1]$. Here the time interval $\Delta t = T - t_j$ between the historical item and target item is normalized with Equation (1), denoted as Δt_n. The relational dynamic history embedding encodes the specific meaning of the history sequence for each relation type, which will serve as the foundation to obtain the final KDA history representation $\mathbf{m}_{u,i}$. The two components of the attention weight will be elaborated in Section 5.2.

5.1.2 Cross-relation Influence Modeling (Self-attention Layer). Assume we have obtained the relational dynamic history embeddings $\mathbf{s}_{i,\tau}$, the self-attention mechanism is leveraged to capture the mutual influence between relations (second-level aggregation). Generally, the common adopted scaled dot-product attention [53] can be defined as:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \tag{4}$$

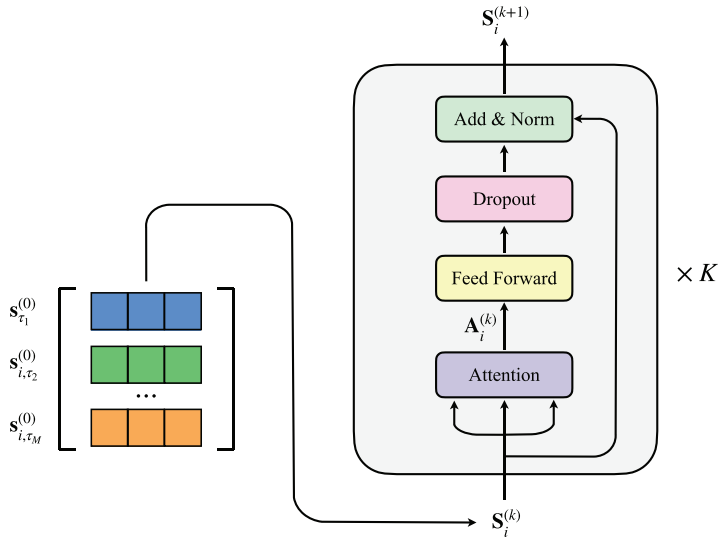


Fig. 4. Illustration of the self-attention mechanism in our KDA model. We leverage self-attention to capture the mutual influence between different relations. There are totally K layers of self-attention, and we adopt dropout, residual connection, and layer normalization in each layer.

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ represent queries, keys and values, respectively, and d is the hidden size of the attention space. It calculates a weighted sum of all values, where the weight depends on the inner product of queries and keys. The scale factor $\sqrt{d}$ is to avoid overly large values of the inner product. As for self-attention, the query, key, and value all come from the same object. Then the new representation of this object is able to absorb the information of other objects. In our case, the objects refer to different relational dynamic history embeddings $\mathbf{s}_{i,\tau}$. We stack $\mathbf{s}_{i,\tau}$ for all the M kinds of relation types as $\mathbf{S}_i \in \mathbb{R}^{M \times d}$:

$$\mathbf{S}_i = \left(\mathbf{s}_{i,\tau_1}^T; \mathbf{s}_{i,\tau_2}^T; \ldots; \mathbf{s}_{i,\tau_M}^T\right). \tag{5}$$

Then the query, key, and value are obtained by linear projections:

$$\mathbf{Q} = \mathbf{S}_i \mathbf{W}^Q, \mathbf{K} = \mathbf{S}_i \mathbf{W}^K, \mathbf{V} = \mathbf{S}_i \mathbf{W}^V, \tag{6}$$

where the projection matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$. Next we can obtain the attentional representations $\mathbf{A}_i \in \mathbb{R}^{M \times d}$:

$$\mathbf{A}_i = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}). \tag{7}$$

Besides, we can stack several self-attention layers to extend the model capacity. To endow nonlinearity to each layer, we apply feed-forward network with a ReLU activation function:

$$\text{FFN}(\mathbf{A}_i) = \text{ReLU}(\mathbf{A}_i \mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{8}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$. We use the same linear transformations in different self-attention layers to achieve parameter sharing.

In the meantime, stacking self-attention layers may cause problems such as over-fitting and vanishing gradients. We adopt dropout regularization, residual connections, and layer normalization techniques to solve these problems inspired by previous work [38, 53] (as shown in Figure 4):

$$\mathbf{S}_i^{(k+1)} = \text{LayerNorm}\left(\mathbf{S}_i^{(k)} + \text{Dropout}\left(\text{FFN}\left(\mathbf{A}_i^{(k)}\right)\right)\right), \tag{9}$$

where $\mathbf{A}_i^{(k)}$ is obtained from $\mathbf{S}_i^{(k)}$ and $\mathbf{S}_i^{(0)}$ is defined as $\mathbf{S}_i^{(0)} = \mathbf{S}_i$. We first apply dropout on the output of the feed-forward network. Dropout [52] is proposed to randomly set the output of the previous layer to zero with probability p, which is shown to be beneficial to alleviate the overfitting problem. Then the representation of the last layer is added by residual connections [18] to propagate low-layer features to higher layers. Finally, layer normalization [1] is used to normalize the outputs of a layer with zero mean and unit variance. It is useful to stabilize and accelerate neural network training. Assume $\mathbf{x}$ is an output vector of the previous layer, layer normalization is defined as:

$$\text{LayerNorm}(\mathbf{x}) = \boldsymbol{\alpha} \otimes \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \boldsymbol{\beta}, \tag{10}$$

where $\otimes$ represents element-wise product, μ and σ are the mean and variance of $\mathbf{x}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are learned scaling factors and bias terms. We do not apply layer normalization on $\mathbf{A}_i^{(k)}$ like Transformer, because this may ruin the meaningful weighted sum of relational dynamic history embeddings in the beginning. Experimental results also demonstrate the intermediate layer normalization leads to a little worse performance.

5.1.3 Pooling Layer. Assume the total number of self-attention layers is K, we can get the final relational dynamic history embeddings $\mathbf{S}_i^{(K)}$. Then we use a pooling layer to derive the final KDA history embedding $\mathbf{m}_{u,i} \in \mathbb{R}^d$ as the target-aware history representation. The concrete method can be (1) average pooling, (2) max pooling, or (3) attention pooling. In the case of average pooling, we simply average the embeddings for all the relation types:

$$\mathbf{m}_{u,i} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{s}_{i,\tau_m}^{(K)}. \tag{11}$$

While max pooling applies element-wise maximum across these embeddings:

$$\mathbf{m}_{u,i} = \max\left(\mathbf{s}_{i,\tau_1}^{(K)}, \mathbf{s}_{i,\tau_2}^{(K)}, \ldots, \mathbf{s}_{i,\tau_M}^{(K)}\right). \tag{12}$$

The attention pooling method considers that users may pay different attentions to different relation types according to users' characteristics:

$$\mathbf{m}_{u,i} = \text{softmax}\left([a(u,\tau_1), a(u,\tau_2), \ldots, a(u,\tau_M)]\right) \mathbf{S}_i^{(K)}, \tag{13}$$

where $a(u,\tau)$ is the attention score between user u and relation type τ. We define it as follows:

$$a(u,\tau) = \mathbf{h}^T(\tanh\left((\mathbf{p}_u \otimes \mathbf{x}_\tau)\mathbf{W}_3 + \mathbf{b}_3\right)). \tag{14}$$

The linear transformation $\mathbf{W}_3 \in \mathbb{R}^{d \times l}, \mathbf{b}_3 \in \mathbb{R}^l$ projects the element-wise product of user and relation type embeddings into the attention space, where l is the hidden size of the attention space. $\mathbf{h} \in \mathbb{R}^l$ subsequently projects the hidden vector into attention score. We will compare the performance when using different pooling methods in the experiments, and average pooling is the default method in our model.

At this stage, we have obtained the final KDA history representation $\mathbf{m}_{u,i}$. Then we can calculate the ranking score according to Equation (2), and $\hat{y}_{u,i}$ will be used to rank candidate items and generate the Top-K recommendation list.

5.2 Relational Dynamic History Aggregation

Here we focus on how to determine the attention weight in Equation (3) (first-level aggregation), which is the key innovative point in KDA. Remind that the aggregation weight consists of a relational intensity score $\text{INT}(i,j,\tau)$ and a temporal decay $f_\tau(\Delta t_n)$. For the first one, we calculate the

relevance degree between interacted items and the target item. For the other, we estimate temporal decay functions based on Fourier transform to control the temporal evolution of different relational effects.

5.2.1 Relational Intensity. First, we discuss the design of relational intensity score. Generally, the relational data is organized as KG, and knowledge graph embedding is a popular method to learn signals from relational data, which aims at embedding knowledge graph into a continuous vector space. In the knowledge graph embedding task, a score function $\text{Rel}_r(i, j)$ is usually learned to measure the plausibility of a triplet (i, j, r). The true triplets in $\mathcal{R}$ will get higher scores compared with corrupted ones that do not exist in the knowledge graph.

From another point of view, the score $\text{Rel}_r(j, i)$ between historical item j and the target item i can also be seen as a sign of relational intensity. Interactions with higher relational intensities in terms of a specific relation type should play more important roles in corresponding relational dynamic history embedding. Thus, we directly use the same score function in the knowledge graph embedding task to calculate the relational intensity score $\text{INT}(i, j, \tau)$:

$$\text{INT}(i, j, \tau) = \frac{\exp\left(\text{Rel}_r(j, i)\right)}{\sum_{j' \in S_u^T} \exp\left(\text{Rel}_r(j', i)\right)}. \tag{15}$$

In this way, historical items can be aggregated according to how much they are relational to the target item. As for the concrete definition of the score function, we leave it to Section 5.3.

Notice that the score function $\text{Rel}_r(j, i)$ relies on relation r, which consists of both relation type τ and relation value v in our setting. But the relation intensity score $\text{INT}(i, j, \tau)$ only determine the relation type. To solve this problem, for relation types with multiple relation values, we introduce the corresponding attributes of the target item i as relation values. Then the relational intensity measures whether the historical item shares the same attribute with the target item. While for relation types with no detailed values (e.g., *also_buy*), the relation value will be the specific value *None* as described in Section 3.1.

Previous knowledge-enhanced studies with a joint-learning framework generally only share embeddings between the recommendation task and knowledge graph embedding task, ignoring the model capacity to predict relational intensities learned in the KG. Here, the shared score function acts as a second bridge besides the shared embeddings, which enables our KDA model to fully exploit the knowledge learned in the KG. Another advantage of relational intensity is that it can handle missing relations in the knowledge graph. For example, in Figure 3, the *also_buy* relation is probably missing between the HUAWEI phone and AirPods 2. But when deriving $\mathbf{s}_{i,\tau}$ for *also_buy* relation, the attention score of the previous consumption for the HUAWEI phone should still be higher than other items (except for iPhone), since HUAWEI phone is in a similar position with iPhone in the KG to some extent. Benefited from relational intensity, the KDA model can better generalize the knowledge learned in the KG to softly aggregate the entire history sequence with knowledge-aware weights.

5.2.2 Fourier-based Temporal Evolution. Besides relational intensities between historical items and the target item, the time interval also plays an important role. In the example above, although the relational intensity of the HUAWEI phone is high, its final contribution to corresponding relational dynamic history embedding will be lowered because of the long time interval.

Based on the empirical study in Section 4, we assume the temporal evolution for each relation type τ can be cultivated by a latent and continuous temporal decay function $f_\tau(\Delta t_n)$. The temporal decay function controls how the relational effects drift as the time interval increases, which determines the attention weight in Equation (3) together with relational intensity. An intuitive solution is estimating a relation-specific function with a predefined functional form. But as analyzed

in the empirical study, the temporal evolutions of different item relations vary dramatically across datasets and relation types. Hence a predefined functional form of temporal evolutions can hardly generalize to different application domains.

The main problem here is that the temporal decay function is assumed to be continuous in the time domain, which cannot be directly optimized in the neural network. If a continuous function can be encoded to a fixed-size vector, then the model can learn the temporal evolution from data without any prior knowledge. Inspired by DFT [6], our key insight is to transform $f_\tau(\Delta t_n)$ to the frequency domain $F_\tau[w]$:

$$f_\tau(\Delta t_n) \leftrightarrow F_\tau[w]. \tag{16}$$

Given the point of DFT N, DFT can convert a finite sequence of equally spaced samples of a continuous function $f(t)$ into a same-length sequence, which is a complex-valued function $F[w]$ of N equal-spaced frequencies:

$$w_k = \frac{2\pi}{N}k, \quad k = 0, 1, \ldots, N-1. \tag{17}$$

$F[w_k]$ contains information about the amplitude and phase of the sinusoid wave of frequency w_k in $f(t)$. In this way, a continuous function in time domain can be represented as a discrete function in frequency domain. This is equivalent to encoding the time-domain function to a complex-valued embedding, which we call *frequency embedding*. Given the frequency embedding, we can get the value of $f(t)$ for any input t via IDFT. Next we describe the details about parameterizing frequency-domain representation $F_\tau[w]$ for each temporal decay function $f_\tau(\Delta t_n)$.

In our KDA model, we give each relation type τ a frequency embedding $\mathbf{F}_\tau \in \mathbb{C}^N$ as the frequency-domain representation, which can be randomly initialized and will be optimized by the loss function together with other parameters. Subsequently, for any time interval Δt_n, we can get $f_\tau(\Delta t_n)$ by applying IDFT on $\mathbf{F}_\tau$:

$$f_\tau(\Delta t_n) = \frac{1}{N}\sum_{k=0}^{N-1} F_\tau[w_k] e^{j w_k \Delta t_n}. \tag{18}$$

To facilitate the advantage of Fast Fourier Transform, we assure N to be the integral power of 2. Besides, since the values of temporal decay functions are real numbers, we only retain the semi-positive frequencies (i.e., the first $N/2$ elements of $\mathbf{F}_\tau$). When transforming back to the time domain, we compute the complex conjugates $\overline{F_\tau[w_k]}$ of these frequencies and pad them at the end. This ensures that the IDFT produces real values as well. Formally, Equation (18) will be changed to:

$$f_\tau(\Delta t_n) = \frac{1}{N}\sum_{k=0}^{N/2-1} \left(F_\tau[w_k] e^{j w_k \Delta t_n} + \overline{F_\tau[w_k]} e^{-j w_k \Delta t_n} \right). \tag{19}$$

With frequency embeddings, we have already got the continuous temporal decay function in the attention weight. Besides, we can also leverage information in data to optimize the initialization of frequency embeddings. Although temporal decay function can not be directly observed from data, the distribution of time intervals between interactions in user sequences and their relational neighbors (shown in Section 4) can be seen as an equidistant sampling of $f_\tau(\Delta t_n)$. Thus, we can apply DFT on the time interval distribution for each relation type to initialize $\mathbf{F}_\tau$ in the beginning. Assume the distribution of time intervals between interactions and their relational neighbors in terms of the relational type τ is $\mathcal{F}_\tau[x]$, where x is the index of the time interval slot and $\mathcal{F}_\tau[x]$ is the corresponding number of cases in data. We discrete the normalized time intervals Δt_n into slots as follows:

$$x = \lfloor \Delta t_n \rfloor = \lfloor \max\left(0, \log_2(\Delta t / 60)\right) \rfloor. \tag{20}$$

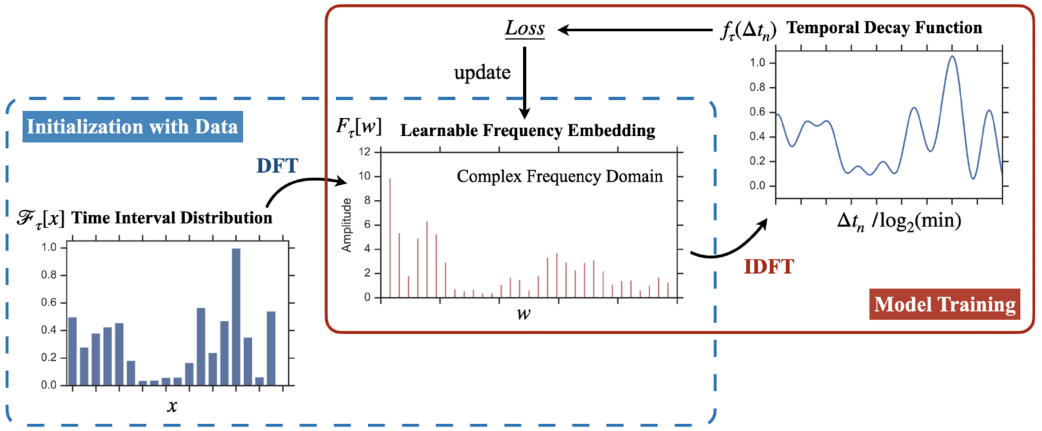


Fig. 5. Illustration of the Fourier-based method to estimate the temporal decay function. The complex-valued frequency embedding serves as a bridge in frequency domain to adaptively learn different temporal evolutions, and it will be optimized together with other parameters according to the objective function.

The total number of slots is denoted as X. Then given the point of DFT $N \geq X$, we can apply DFT on the periodic extension of $\mathcal{F}_\tau[x]$ (i.e., $\mathcal{F}_\tau[x + X] = \mathcal{F}_\tau[x]$):

$$F_\tau[w_k] = \sum_{n=0}^{N-1} \mathcal{F}_\tau[n] e^{-j w_k n}, \tag{21}$$

The above DFT results will be used to initialize $\mathbf{F}_\tau$ before training. Figure 5 gives an illustration of this DFT-based method to estimate temporal decay functions. We normalize the maximum value of the time interval distribution $\mathcal{F}_\tau[x]$ to 1, which ensures the output of IDFT is on the same scale. We will compare the results with different initialization methods in Section 6.3, and the results of DFT on the time interval distribution are used to initialize $\mathbf{F}_\tau$ by default.

In summary, the proposed DFT-based estimation method can be seen as a general approach to encode latent and continuous temporal functions into frequency-domain embeddings, which is easy to be incorporated in most neural models. For instance, if we additionally want to model the user-specific perceptions on temporal evolution, we can define a user-specific embedding $\mathbf{F_u}$ and represent the temporal decay function as:

$$f_{\tau,u}(\Delta t_n) = \frac{1}{N} \sum_{k=0}^{N-1} (F_\tau[w_k] + F_u[w_k]) e^{j w_k \Delta t_n}. \tag{22}$$

In our experiments, we find adding user-specific frequency embeddings yields similar results with Equation (18), hence we choose to only model relation-specific frequency embeddings $\mathbf{F}_\tau$. But similar ideas can be adopted in other applications and model designs.

5.2.3 Remarks. At this stage, we have clearly defined the two components of the first-level aggregation weight. From another point of view, the derivation of relational dynamic history embeddings can also be seen as a variant of multi-head attention. The query is the target item; the key and value are historical items; the head is corresponding to each relation type. Differently, here each head has explicit meanings compared with the original multi-head attention. And attention scores are knowledge-aware and dynamic, which integrate both relational intensity and their temporal evolutionary effects. Meanwhile, unlike directly applying self-attention on historical

interactions that faces the problem of variant sequence lengths, the fixed number of relation types makes the subsequent self-attention layer can fully exploit the mutual influence of relations.

Besides, note that we isolate the temporal decay $f_\tau(\Delta t_n)$ as a scaling factor, instead of including it into the calculation of the relational intensity score with softmax. The rationale behind is that the temporal decay effects are absolute but not relative. If the historical interactions all occurred long ago, then the temporal decay of their relational effects should not be similar to the situation that all the interactions happened recently. Experiments also show that isolating the temporal decay as a scaling factor leads to significant performance improvements.

5.3 Item Relation Modeling

Now the remaining part is the item relation modeling module referred to in Section 5.2.1. In this module, we learn a knowledge graph embedding task to retain semantic meanings into item embeddings. The main target is to define the score function $\text{Rel}_r(i, j)$ to measure the plausibility of a triplet (i, j, r). Following the setting in previous work [60], we use the summation of the two-level hierarchy components of the relation as its embedding. Precisely, the representation of relation $r =< \tau, v >$ is formulated as:

$$\mathbf{r} = \mathbf{x}_\tau + \mathbf{z}_v. \tag{23}$$

In this way, relations with the same type keep similar representations, and the value embeddings empower the model fidelity to tackle the situation that the same relation type has different values. Meanwhile, many popular methods [5, 39, 58, 62] for knowledge graph embedding task can be adopted under this setting, such as translation-based methods (e.g., TransE [5]) and semantic matching methods (e.g., DistMult [62]). The difference between these two methods mainly lies in the symmetry of the score function. For TransE, $\text{Rel}_r(i, j) \neq \text{Rel}_r(j, i)$, while DistMult is symmetric. Considering the fact that most item relations extracted from attributes are undirected (i.e., the shared brand of Apple holds for both (iPhone, AirPods) and (AirPods, iPhone)), we use DistMult by default. Besides, many other state-of-the-art knowledge embedding methods can be leveraged, we leave this as the future work.

In the case of DistMult, it captures relation-specific interactions between item pairs. The score function is defined as:

$$\text{Rel}_r(i, j) = \mathbf{q}_i^T \text{diag}(\mathbf{r}) \mathbf{q}_j, \tag{24}$$

where $\text{diag}(\mathbf{r})$ denotes a diagonal matrix whose main diagonal elements are equal to $\mathbf{r}$.

To learn semantics in the item relations graph, we want to maximize $\text{Rel}_r(i, j)$ for observed triplets and minimize it for unobserved ones. Based on that, a pairwise loss is optimized:

$$\mathcal{L}_{rel} = - \sum_{(i,j,r) \in \mathcal{R}} \log \sigma \left(\text{Rel}_r(i, j) - \text{Rel}_r(i^-, j^-) \right). \tag{25}$$

For each triplet in $\mathcal{R}$, we randomly corrupt the head item or tail item and make sure $(i^-, j^-, r) \notin \mathcal{R}$. If the head item is corrupted, then i^- is a randomly sampled item and $j^- = j$, vice versa. The probability of corrupting the head item is set to 0.5 in our model.

5.4 Multi-task Learning

To effectively learn parameters for the recommendation, as well as integrate the relational structure between items, we jointly learn the main recommendation task and the knowledge graph embedding task through a multi-task learning framework.

For the recommendation task, we optimize a pairwise ranking loss [47] as follows:

$$\mathcal{L}_{rec} = - \sum_{u \in \mathcal{U}} \sum_{i=2}^{N_u} \log \sigma \left(\hat{y}_{u,i} - \hat{y}_{u,i^-} \right), \tag{26}$$

ALGORITHM 1: Learning algorithm for KDA

Input: user-item interactions data $\bigcup_{u \in \mathcal{U}} S_u$; item-item relational data $\mathcal{R}$; point of DFT N; self-attention layer number K; learning rate η; embedding size d; attention hidden size l; knowledge embedding loss coefficient γ; l2-normalization coefficient λ

Output: model parameters Θ

Randomly initialize all parameters Θ
for *relation type* τ **do**
    Calculate time interval distributions $\mathcal{F}_\tau[x]$
    $\mathbf{F}_\tau \leftarrow DFT(\mathcal{F}_\tau[x], N)$
end for
while *stopping criteria is not met* **do**
    Draw a mini-batch (u, i, i^-, T, S_u^T) from $\bigcup_{u \in \mathcal{U}} S_u$
    Draw a mini-batch (i, j, i^-, j^-, r) from $\mathcal{R}$ with the same batch size
    Compute $\mathcal{L}_{rec}$ according to Equation (26)
    Compute $\mathcal{L}_{rel}$ according to Equation (25)
    $\mathcal{L} \leftarrow \mathcal{L}_{rec} + \gamma \mathcal{L}_{rel} + \lambda ||\Theta||_2$
    Update model parameters Θ according to $\mathcal{L}$ and optimizer
end while
return Θ

where σ denotes the sigmoid function and we randomly sample a negative item $i^- \notin S_u$ for each training instance. The index of i starts from 2, because sequential recommendation should have at least one historical interaction.

Then the joint objective function is defined as:

$$\min_{\Theta} \quad \mathcal{L} = \mathcal{L}_{rec} + \gamma \mathcal{L}_{rel} + \lambda ||\Theta||_2, \tag{27}$$

where Θ is the parameter space, γ is the coefficient of the knowledge graph embedding task, and λ is the regularization coefficient.

With regard to the learning procedure, we first calculate the time interval distribution of each relation type and save the DFT results. The frequency-domain embeddings $\mathbf{F}_\tau$ are initialized with these DFT results before training. Since $||\mathcal{R}||$ can be very large under our two-level hierarchy setting, we iterate the training data based on user-item interactions in the recommendation task. At each training step, we draw a mini-batch from $\mathcal{R}$ with the same batch size of the recommendation task. The overall training procedure of KDA is illustrated in Algorithm 1.

Finally, we analyze the time complexity of our model. Now consider all the computations for a single training instance. For the recommendation task, if the maximum length of history is H, aggregating the historical interactions for a specific relation type takes $O((N + d)H)$, corresponding to the IDFT operation and relational intensity calculation for each previous interaction. While the subsequent self-attention with K layers takes $O(KM^2d)$. As for the prediction layer and knowledge graph embedding task, they are all controlled within $O(d)$, which can be ignored on the whole. Therefore, the total time complexity for a training instance is

$$O(M(N + d)H + KM^2d). \tag{28}$$

Generally, the number of relation type M is not large (less than 10 in our cases), hence the time complexity for learning KDA is acceptable compared to other state-of-the-art models.

5.5 Discussion

Here we discuss the connections between KDA and other related recommendation models.

5.5.1 Conventional Sequential Recommendation. For conventional sequential models such as FISM [29] and SVD++ [34], the main idea is to capture item-based collaborative similarity; thus, they use the dot product between historical and target item embeddings to make recommendations. Our KDA model can easily generalize these conventional CF methods. If there is only a single latent relation type τ_0 whose aggregation weight is $1/\sqrt{|S_u^T|}$, and there is no self-attention layer, then the KDA history representation will become

$$\mathbf{m}_{u,i} = \frac{1}{\sqrt{|S_u^T|}} \sum_{j \in S_u^T} \mathbf{p}_j, \tag{29}$$

which leads to the same ranking function as FISM. In fact, compared to traditional sequential recommendation methods, the item relations and their temporal evolutionary effects addressed in history aggregation weights empower KDA to be better at understanding dynamic user intention.

5.5.2 Knowledge-enhanced Recommendation. Item relations have been shown to be useful in many studies, and embedding-based methods attract more and more attention due to the scalability. Most popular knowledge-enhanced methods loosely connect the knowledge graph embedding task and recommendation task with shared entity embeddings, such as CKE [65], KTUP [7], and RCF [60]. Some of them move one step forward to consider the relationship between historical items and the target item, such as RCF and Chorus [55]. However, they need to explicitly determine whether the historical item is relational to the target one, which is not only inefficient but also easy to be affected by the incompleteness of the knowledge graph (missing relations). Differently, we introduce relational intensity and share the score function in the knowledge graph embedding task. This enables our KDA model to generalize the knowledge learned in KG, so as to softly determine the relational effect of each previous interaction. Previous methods that directly divide the history sequence according to relations to the target item can also be seen as a special case of our relational intensity attention. As a result, the relational intensity connects the two tasks more tightly, which is a better approach to exploit the information in the item relations graph.

5.5.3 Time-aware Recommendation. As for the models with temporal information, KDA follows the idea of underweighting the effects of historical interactions. Unlike incorporating temporal information as contextual features, the temporal decay effects can handle unseen time periods and hence gain better generalization ability. However, most existing methods predefine the functional form of the temporal decay effects (e.g., exponential and normal distribution in SLRC and Chorus), which is not scalable and needs domain-specific knowledge. Our key insight about temporal evolution is to transform the continuous temporal decay function to a discrete representation in frequency domain based on DFT. The learnable frequency embedding for each relation type enables KDA to capture the temporal evolution of different item relations in a domain-adaptive way. This can also serve as a general approach to estimate continuous functions with discrete embeddings in neural models, because it is easy to be integrated into the learning procedure and achieves promising results even with randomly initialized values.

6 EXPERIMENTS

In this section, we present our experimental settings and results. Our experiments are designed to answer the following research questions:

- **RQ1:** How effective is our proposed KDA model in different datasets compared to state-of-the-art methods on the Top-K recommendation task?

- **RQ2:** Does our Fourier-based method successfully model the temporal evolutionary effects of historical relational interactions?
- **RQ3:** What are the impacts of the relational intensity and different knowledge graph embedding methods?
- **RQ4:** How do the hyper-parameter settings influence the performance, such as the coefficient of the KG embedding loss and the number of self-attention layers?

6.1 Experimental Settings

6.1.1 Datasets and Evaluation Protocols. We use the same datasets described in Section 3.2, covering various recommendation scenarios. Following the leave-one-out strategy in the literature [11, 29, 53], 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 the items for some methods when the dataset is large, we randomly sample 99 negative items (i.e., not in S_u) and rank the ground-truth item together with these items. This approach is also widely adopted in other work [38, 55].

To evaluate the quality of the recommendation, we use **Hit Ratio (HR)** and **Normalized Discounted Cumulative Gain (NDCG)** [28] as evaluation metrics. HR@K measures whether the ground-truth item appears in the Top-K recommendation list, while NDCG@K concerns the ranking position of the ground-truth item. Let $g_u \in [1, 100]$ denote the rank of the ground-truth item for each user u, then HR@K and NDCG@K can be defined as follows under our experimental settings:

$$\begin{aligned} \text{HR@K} &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} I(g_u \leq K), \\ \text{NDCG@K} &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{I(g_u \leq K)}{\log_2(g_u + 1)}, \end{aligned} \tag{30}$$

where $I(\cdot)$ is an indicator function and returns 1 when the condition is true, otherwise 0. We repeat each experiment 5 times with different random seeds and report the average score.

6.1.2 Baseline Methods. To evaluate the performance of our KDA model, we compare it with 8 baselines. These methods include traditional factorization-based methods (BPR, FISM), neural sequential recommendation (NARM, SASRec), and knowledge-enhanced/time-aware methods (TiSASRec, TransFM, Chorus, RCF).

- **BPR** [47]. This method applies Bayesian Personalized Ranking objective function to optimize the Matrix Factorization model.
- **FISM** [29]. This is a factorization-based sequential recommendation model with mean aggregation of the interacted item embeddings.
- **NARM** [37]. It is a neural model with attention mechanism applied on the hidden states of Gated Recurrent Units [12] to better capture the sequential dependency.
- **SASRec** [29]. This method utilizes self-attention [53] to exploit the mutual influence between historical interactions, which is a state-of-the-art neural sequential model.
- **TiSASRec** [38]. This is a improved version of SASRec, which considers time intervals between historical interactions when performing self-attention.
- **TransFM** [44]. This is a knowledge-enhanced method with combination of translation-based approaches and FM [46]. It incorporates temporal information as context features, leading to time-aware recommendation.

Table 3. Comparison of Baseline Methods and KDA

Characteristics	BPR	FISM	NARM	SASRec	TiSASRec	TransFM	Chorus	RCF	**KDA**
Sequential Model		✓	✓	✓	✓	✓	✓	✓	✓
Neural Model			✓	✓	✓			✓	✓
Domain Adaptive	✓	✓	✓	✓	✓	✓		✓	✓
Knowledge-aware						✓	✓	✓	✓
Time-aware					✓	✓	✓		✓

- **Chorus** [55]. It is a state-of-the-art method with item relations and temporal evolution. But it only concerns whether there is relational consumption previously, and needs hand-crafted forms of temporal decay functions.
- **RCF** [60]. This model presents a joint-learning framework to incorporate item relations in the sequential recommendation with a two-level neural attention network.

The comparison of our KDA model and these baseline methods are shown in Table 3.

6.1.3 Implement Details. We implement our model with *PyTorch* and the implementation codes are publicly available.[5] All the models are optimized by Adam optimizer, which is shown to be effective in neural model training. Early stop is adopted if NDCG@5 on the validation dataset does not improve for 10 epochs. For fair comparisons, the batch size is set to 256; the embedding size is set to 64, and the attention hidden size is set to 10. We consider a maximum of 20 most recent interactions as the historical sequence for all models.

We tune hyper-parameters according to the performance in the validation set. All baseline methods are carefully tuned to achieve optimal performances. The learning rates η are tuned within $[10^{-2}, 10^{-3}, 10^{-4}]$. The l2-normalization coefficients λ are tuned amongst $[10^{-4}, 10^{-5}, 10^{-6}, 0]$. For SASRec, TiSASRec and KDA, the number of self-attention layers is tuned within $[1, 2, 3, 4, 5]$. For RCF and KDA, the coefficient of knowledge graph embedding task γ is tuned between $[0.01, 0.1, 1, 5, 10]$. For KDA, the point of DFT is set to 64. DistMult and average pooling are used by default for the score function and pooling method, respectively. All the parameters are normally initialized with 0 mean and 0.01 standard deviation. The sensitivity of some key hyper-parameters will be further explored in Section 6.5. All the experiments are conducted with a single GTX-1080 Ti GPU.

6.2 Performance Comparison (RQ1)

Table 4 shows the Top-K recommendation performance of all methods, as well as the relative improvements of the best-performing KDA model over corresponding baseline. For our KDA model, we report the results when utilizing three kinds of pooling methods (Section 5.1.3): average pooling, max pooling, and attention pooling, denoted as KDA_{avg}, KDA_{max}, and KDA_{attn}, respectively.

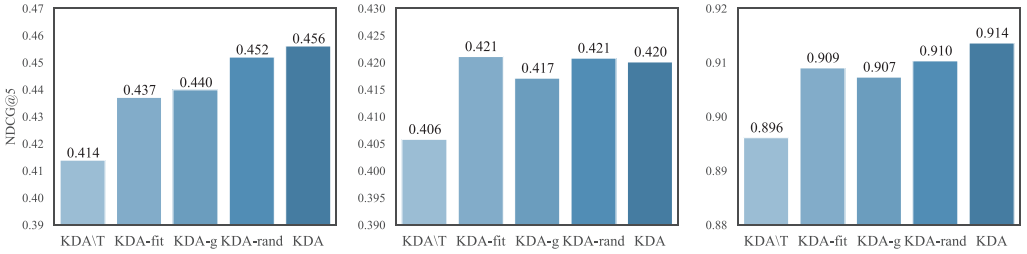
Among all baselines, RCF generally achieves promising performance due to the target-aware and knowledge-enhanced history aggregation. In dense dataset (i.e., MovieLens-100k), SASRec and TiSASRec gain outstanding results because of their strong ability to capture the latent mutual influence of sequential interactions. With regard to item relations, knowledge-aware methods (i.e., TransFM, Chorus, RCF) generally outperform traditional sequential models, especially in Amazon Electronics. But item relations seem to be less useful in RecSys2017. Temporal information also plays an important role, and the performance of TiSASRec is remarkable in all

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

Table 4. Performance of Different Models

	MovieLens-100k				
Model	NDCG@5	HR@5	NDCG@10	HR@10	Improv.
BPR	0.3830	0.5175	0.4320	0.6691	+20.17%
FISM	0.3588	0.5228	0.4135	0.6903	+22.38%
NARM	0.3716	0.5345	0.4300	0.7137	+18.47%
SASRec	0.4285	0.6002	0.4752	0.7434	+7.09%
TiSASRec	0.4271	0.5938	0.4767	0.7444	+7.34%
TransFM	0.3925	0.5525	0.4487	0.7253	+14.13%
Chorus	0.3959	0.5492	0.4462	0.7032	+15.05%
RCF	0.4043	0.5652	0.4514	0.7094	+13.05%
KDA_{avg}	0.4550**	0.6288**	0.5020**	0.7731**	—
KDA_{max}	**0.4703****	**0.6320****	**0.5174****	**0.7762****	—
KDA_{attn}	0.4455**	0.6246**	0.4906**	0.7625**	—
	Amazon Electronics				
Model	NDCG@5	HR@5	NDCG@10	HR@10	Improv.
BPR	0.3003	0.4067	0.3389	0.5262	+35.29%
FISM	0.3432	0.4615	0.3838	0.5872	+19.55%
NARM	0.3399	0.4584	0.3810	0.5855	+20.35%
SASRec	0.3587	0.4805	0.4000	0.6083	+14.82%
TiSASRec	0.3594	0.4813	0.4006	0.6086	+14.66%
TransFM	0.3564	0.4832	0.4000	0.6181	+14.40%
Chorus	0.3635	0.4865	0.4019	0.6154	+13.62%
RCF	0.3635	0.4852	0.4045	0.6121	+13.66%
KDA_{avg}	**0.4201****	**0.5544****	**0.4603****	**0.6795****	—
KDA_{max}	0.4081**	0.5410**	0.4493**	0.6685**	—
KDA_{attn}	0.3947**	0.5223**	0.4358**	0.6495**	—
	RecSys2017				
Model	NDCG@5	HR@5	NDCG@10	HR@10	Improv.
BPR	0.8069	0.8546	0.8164	0.8837	+11.95%
FISM	0.8792	0.9166	0.8859	0.9373	+3.95%
NARM	0.8263	0.8786	0.8363	0.9095	+9.08%
SASRec	0.8863	0.9173	0.8930	0.9381	+3.49%
TiSASRec	0.8961	0.9282	0.9021	0.9465	+2.42%
TransFM	0.8382	0.8994	0.8514	0.9399	+6.70%
Chorus	0.8434	0.9025	0.8543	0.9358	+6.46%
RCF	0.8688	0.9270	0.8779	0.9549	+3.73%
KDA_{avg}	0.9136**	0.9467**	0.9219**	0.9722**	—
KDA_{max}	0.9076**	0.9404**	0.9135**	0.9585**	—
KDA_{attn}	**0.9146****	**0.9505****	**0.9225****	**0.9743****	—

The best-performing method is in bold, and the second best method is underlined. ** means significantly better than the strongest baseline ($p < 0.01$). "Improv." means the relative improvement of the best KDA model over corresponding baseline (averaged across all metrics).



(a) MovieLens-100k (b) Amazon Electronics (c) RecSys2017

Fig. 6. Ablation study about temporal evolution. KDA\T means there is no temporal evolution; KDA-fit denotes utilizing the Weibull distribution to fit temporal decay functions; KDA-g means the temporal evolution is global but not relation-specific; KDA-rand means the frequency embeddings are randomly initialized.

three datasets. TiSASRec becomes the strongest baseline in RecSys2017, probably because of the complex temporal patterns of this dataset shown in Section 4. But it only captures time intervals within historical interactions, ignoring the temporal decay up to the target item. Besides, it is noteworthy that although Chorus introduces both item relations and temporal dynamics, it only performs well in Amazon Electronics. The main reason is that the temporal decay functions in Chorus need a predefined functional form, which may be not scalable to different domains.

Our proposed KDA model improves over the best baseline methods by a large margin in all datasets. The rationale behind lies in three parts: (1) We use relational intensity score to softly aggregate historical interactions, rather than directly dividing the sequence like RCF. We will show that utilizing the method of RCF will lead to poor results in Section 6.4; (2) we devise a Fourier-based method to adaptively incorporate temporal evolutionary effects into model training, which is not only effective but also flexible. (3) The mutual influence between relations is further explored with the self-attention mechanism. Besides, notice that the performances of different pooling methods vary across different datasets, and the best choices are not consistent. The average pooling method generally gets promising results, but max pooling is extremely powerful in MovieLens-100k. And attention pooling is a little better than the others in RecSys2017, but performs worse in the other two datasets (still better than baseline methods). As a result, the concrete method needs to be determined according to the dataset, which may rely on data scale and the number of relations. According to our experiences, average pooling and max pooling are generally stable, hence we use average pooling by default in the following experiments.

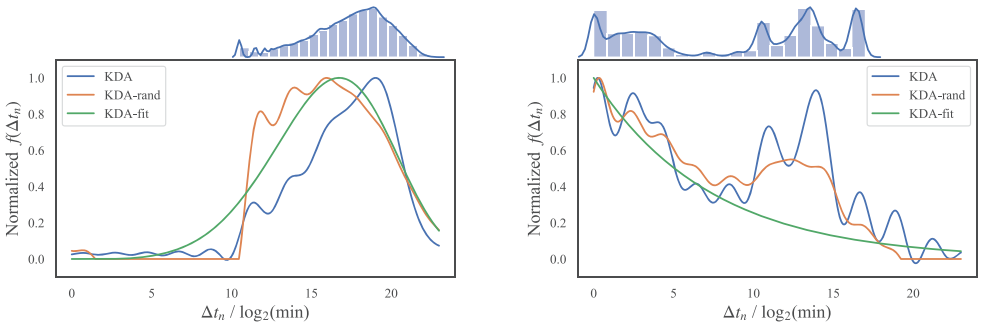
6.3 Temporal Evolution Analyses (RQ2)

In this section, we investigate the impacts of temporal information and our Fourier-based estimation method. Some modifications of our KDA model are compared here:

- **KDA\T.** This model removes the temporal decay $f_\tau(\Delta t_n)$ in Equation (3). The historical interactions are aggregated only by the relational intensity score.
- **KDA-fit.** This model estimates temporal decay functions by fitting a specific distribution with relation-specific parameters. Weibull distribution is shown to be flexible and effective in our experiments, in which case Equation (18) will become

$$f_\tau(\Delta t_n; \lambda_\tau, k_\tau) = \frac{k_\tau}{\lambda_\tau}\left(\frac{\Delta t_n}{\lambda_\tau}\right)^{k_\tau - 1} e^{-(\Delta t_n/\lambda_\tau)^{k_\tau}}.$$

- **KDA-g.** This model assumes the temporal evolution is not relevant to relation types, and only has a global frequency embedding for all relation types.



(a) Amazon Electronics (b) RecSys2017

Fig. 7. Visualization of the averaged temporal decay functions learned by different methods, and global time interval distributions in data (above). The temporal decay functions of KDA are the most consistent with data. Randomly initializing frequency embeddings (KDA-rand) yields similar results with KDA, while fitting a predefined distribution (KDA-fit) can hardly capture multi-modal effects and scale to different datasets.

- **KDA-rand.** This model does not initialize frequency embeddings with the DFT results of time interval distributions in data. The real and imaginary part of $\mathbf{F}_\tau$ are normally initialized with 0 mean and 0.01 standard deviation like other embeddings.

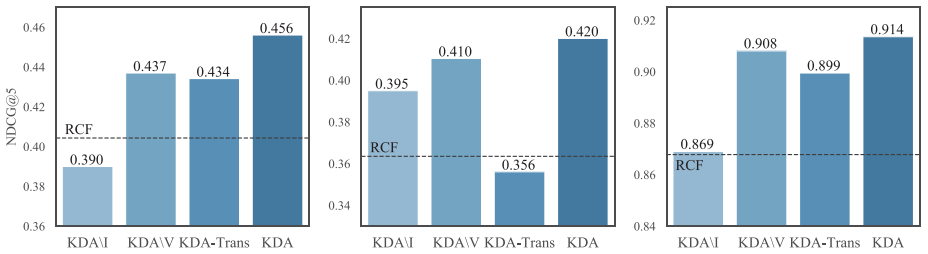
Figure 6 shows NDCG@5 of the KDA model and these variants in all three datasets. We mainly have the following observations:

1) Temporal evolution of relational effects is indeed helpful to understand dynamic user intention. KDA\T performs the worst among these variants in all three datasets. The consistent performance drop demonstrates the usefulness of taking temporal evolutionary effects into consideration. This verifies the observations in Section 4 that the effects of previous relational interactions are actually sensitive to time intervals. Comparatively, the impact of temporal information is the largest in MovieLens-100k, which is consistent with previous studies in this dataset [34].

2) The Fourier-based estimation of temporal decay functions is effective. Although Weibull distribution is flexible compared to other common distributions, the performances of KDA-fit are still not stable across different datasets. It performs similarly with our KDA model in Amazon Electronics, since the time interval distribution in this dataset is comparatively simpler and suitable for Weibull distribution. While in the other two datasets, KDA-fit yields worse results, especially in MovieLens-100k. This indicates that estimating temporal decay functions with predefined functional form suffers from the scalability problem. Differently, our Fourier-based method achieves stable and promising results in all three datasets.

3) The frequency-domain embedding can serve as a general strategy to model temporal evolution. It is noteworthy that KDA-rand yields similar results with KDA, which demonstrates that the model can jointly learn temporal evolution in data without prior knowledge. In this way, other neural models can easily incorporate temporal evolution with a simple embedding, which is domain-free and flexible. Besides, notice that KDA-g results in consistent performance drop, which shows the importance of relation-specific frequency embeddings. As discussed in Section 5.2, it is also easy to include the influence of other factors such as user characteristics with specific embeddings. However, KDA still outperforms KDA-rand in MovieLens-100k and RecSys2017, indicating the usefulness of the information conveyed by time interval distributions in data, which should be taken into consideration if possible.

Furthermore, Figure 7 visualizes the temporal decay functions learned by different variants of KDA in Amazon Electronics and RecSys2017. The function values for all relation types in the



(a) MovieLens-100k (b) Amazon Electronics (c) RecSys2017

Fig. 8. Impacts of item relation modeling. KDA\I means there is no relational intensity, and historical interactions are directly divided by their relations to the target item like RCF; KDA\V discards relation values in the relation representation; In KDA-Trans, TransE is used as the score function instead of DistMult.

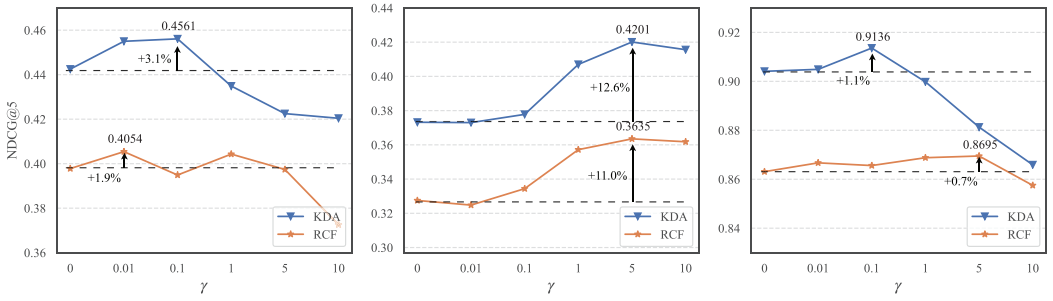
dataset are averaged to observe the global trend, and the maximum value is normalized to 1 for each function. The global time interval distribution is also shown above each figure. It can be seen that the temporal decay functions of KDA are the most consistent with data, since the frequency embeddings are initialized by the DFT results of time interval distributions. But there are still some differences after parameter learning. For example, the short-term effect is enhanced in RecSys2017 compared to the original distribution. More interestingly, frequency embeddings with randomly initialized values (KDA-rand) yield similar functional forms with KDA, which demonstrates the scalability and effectiveness of our Fourier-based method. In this way, we can directly define frequency embeddings without concern about the real distributions in data, and the performance loss is slight in the meantime. With regard to KDA-fit, although the Weibull distribution can be learned to have roughly similar trends with actual distributions, it can hardly capture multi-modal effects in different datasets. For instance, restricted by the functional form, it cannot reflect the long-term effects in RecSys2017. We also find KDA-fit is quite sensitive to the initial values of distribution parameters. Improper parameter initialization will lead to bad performance, which makes it hard to be adopted in real-world applications. Differently, our Fourier-based frequency embedding method does not have this problem and is a more flexible choice.

6.4 Impacts of Item Relation Modeling (RQ3)

Here we further explore the influence of different item relation modeling methods. For one thing, we want to investigate whether relational intensity is useful. For another, we aim to compare different methods to represent and model item relations. Thus, we disign the following variants of KDA:

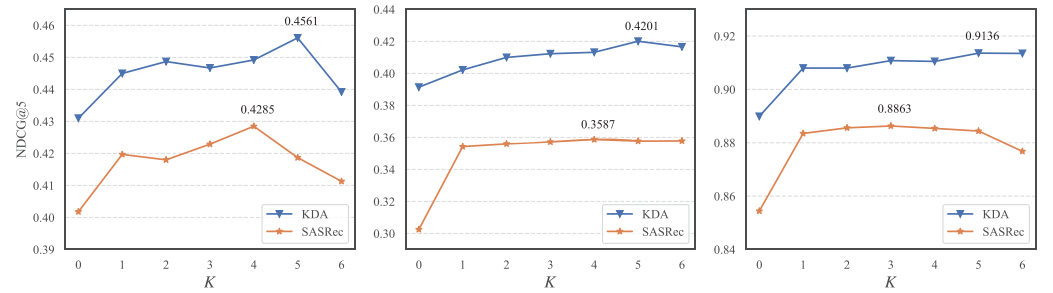
- **KDA\I.** This model divides historical interactions according to their relations to the target item, and only aggregates corresponding relational items when deriving relational dynamic history embeddings, which ignores relational intensities as RCF.
- **KDA\V.** This model removes relation values and only retain relation types (i.e., $\mathbf{r} = \mathbf{x}_\tau$). Item pairs with different attribute values will be seen to have the same relation.
- **KDA-Trans.** In this model, TransE is utilized as the score function instead of DistMult in KDA (i.e., $\text{Rel}_r(i, j) = -||\mathbf{q}_i + \mathbf{r} - \mathbf{q}_j||_2$).

Figure 8 shows NDCG@5 of KDA and these variants, as well as the results of RCF. It can be concluded that our item relation modeling methods in KDA (relational intensity, two-level hierarchy representation, and DistMult score function) together contribute to the best performance in all three datasets.



(a) MovieLens-100k (b) Amazon Electronics (c) RecSys2017

Fig. 9. Performance of KDA and RCF with regard to the coefficient γ of the knowledge graph embedding task in the objective function. The best-performing result for each method is annotated.



(a) MovieLens-100k (b) Amazon Electronics (c) RecSys2017

Fig. 10. Performance of KDA and SASRec with regard to the number of self-attention layers K. The best-performing result for each method is annotated.

First, except for Amazon Electronics, KDA\I results in the largest decline of performance generally, indicating the importance of softly aggregating historical interactions with relational intensity. KDA\I directly divides the history sequence according to their relations to the target item, which neglects some items that are probably relational in the history sequence. Differently, relational intensity can fully exploit the knowledge learned in the KG. With the shared score function, it can handle missing relations and determine the aggregation weight according to relational intensities, leading to tight integration of recommendation and item relation modeling.

Second, KDA\V leads to consistent performance loss in all three datasets. This demonstrates the two-level hierarchy representation of relations can enlarge the model capacity, which helps our KDA model to capture fine-grained characteristics about item relations. The observation about the benefit of introducing relation values is consistent with previous work [60].

Third, KDA-Trans generally performs worse than KDA with DistMult, especially in Amazon Electronics. This may be caused by the fact that most item relations in our datasets are undirected as discussed in Section 5.3. TransE can hardly handle such symmetric relations, while DistMult is simple but more suitable in this case. Amazon Electronics has the largest number of relation triplets, which may account for the bad performance of KDA-Trans in this dataset.

6.5 Hyper-parameter Analyses (RQ4)

In this section, we conduct several experiments to investigate the impacts of some main hyper-parameters in our KDA model, including the coefficient of the knowledge graph embedding loss γ, the number of self-attention layers K, and the point of DFT N.

Table 5. Performance of KDA Model When the Point of DFT Takes Different Values

	Amazon Electronics			
N	NDCG@5	HR@5	NDCG@10	HR@10
32	0.4186	0.5523	0.4594	0.6782
64	0.4201	0.5544	0.4603	0.6795
128	0.4207	0.5537	0.4622	0.6819
256	0.4190	0.5523	0.4602	0.6793

6.5.1 Multi-task Learning Coefficient. The coefficient γ in Equation (27) controls the weight of the knowledge graph embedding task in the total loss, which reflects the importance of item relation modeling. Figure 9 shows the NDCG@5 of KDA and RCF with respect to the multi-task learning coefficient γ. We annotate the result of the best-performing setting for each method in the dataset. From the figure, we can make the following observations.

First, item relation modeling can actually benefit the main recommendation task. Compared to the situations when $\gamma = 0$, both KDA and RCF achieve better performance under proper settings of γ in all three datasets. This indicates the importance of modeling the semantics of item relations. Besides, the overall trends generally increase first and then decrease. The best setting varies across datasets, which may rely on the scale and quality of relational data. In MovieLens-100k and RecSys2017, the coefficients are better to be at a lower level. While for Amazon Electronics, the relational data are not only large scale but also high quality, and hence γ can be set to larger values.

Second, the knowledge graph embedding task is more useful to our KDA model. The figure also shows the relative improvements of KDA and RCF compared to corresponding cases when $\gamma = 0$. We can see our KDA model achieves larger relative improvements in all three datasets. This verifies the statement that KDA can integrate the knowledge graph embedding task more tightly with the shared score function. In practice, the specific value of the multi-task learning coefficient can be determined by cross-validation.

6.5.2 Number of Self-attention Layers. Here we conduct experiments to test the impact of self-attention layer number K. Figure 10 presents the NDCG@5 of KDA and SASRec with respect to the layer number. We can see the overall trends also increase first and then decrease, which demonstrates the usefulness of self-attention layers in limited ranges. The performance gain is especially obvious when the number of self-attention layers changes from 0 to 1 for both KDA and SASRec. However, with subsequent increase of the layer number, although multiple techniques are applied to prevent over-fitting, the performance of SASRec only increases slightly and then begins to drop, especially in sparse datasets. With regard to our KDA model, the performance grows stably when the number of self-attention layers is within a proper range. Besides, the best performance of KDA is generally achieved with deeper self-attention layers. This demonstrates that applying self-attention over fixed-number relational dynamic history embeddings in our KDA model can better exploit the expressiveness of the self-attention mechanism.

6.5.3 Point of DFT. Finally, we investigate whether the point of DFT N will influence the performance of our KDA model. Table 5 shows the experimental results with different N in Amazon Electronics. From the table, we can see the performances are similar for different DFT points (no significant differences). Larger DFT points in limited ranges achieve a little better performance. Results in other datasets demonstrate similar phenomena. It can be concluded that our KDA model

is not that sensitive to the exact point of DFT. The temporal decay functions can be well estimated with both small or large N (at least X). In practice, a smaller N (i.e., size of frequency-domain embeddings) can be set in consideration of the efficiency problem.

7 DISCUSSION ON LIMITATIONS

Beyond the effectiveness and scalability of our KDA model, we discuss some limitations and future directions in this section.

7.1 Long-term User Preference

In this work, we mainly focus on the modeling of short-term user intention. The long-term user preference is only captured by user embeddings, which is not sufficient nor powerful. We have tested in our experiments that the overall performance will not suffer an obvious loss if there are no user embeddings. In recent years, **Graph Neural Network (GNN)** is shown to be an effective method to model long-term user preference. As a result, a possible future direction is to leverage the output of GNN as our basic embeddings for users and items, so as to better combine long-term user preference and short-term user intention.

7.2 Restriction on Item-item Relation

In this work, we assume relations only appear between items, so that the score function in the KG embedding task can be shared to calculate relational intensities between historical items and the target item. Although applicable in most scenarios, this assumption is a little strict. For attribute-based item relations, a more general knowledge graph will represent them by relations between items and other entities, such as (iPhone, *Apple*, *brand_is*). Here we treat these external entities as relation values to construct item-item relations, but it will lead to numerous triplets, because any two items with the same attribute value will constitute a relation. To solve this problem, instead of storing all the item-item relation triplets, we only construct the needed triplets according to the item metadata before each mini-batch in the implementation. Besides, the relationships between external entities cannot be captured now. In the future, we will try to find a general method to model external knowledge beyond item-item relations.

7.3 Model Efficiency

In our KDA model, each target item will get a different history representation $\mathbf{m}_{u,i}$, which is more time-consuming compared to methods based on inner product. This makes KDA more suitable for fine-grained ranking rather than retrieving candidates from large-scale items. While compared to recent target-aware methods (e.g., RCF, Chorus), the computation complexity of our model is still acceptable as discussed in Section 5.4. As for further improvements on efficiency, the number of relation types M in KDA is an influential factor. The main reason is that we need to aggregate history sequence M times, and the subsequent self-attention is sensitive to M. However, it may not be necessary to aggregate historical interactions under the view of each relation type. In the future, some representative relations can be adaptively chosen to improve model efficiency.

8 CONCLUSION

In this work, we focus on better understanding dynamic user intention in the sequential recommendation. Through the empirical study in three real-world datasets, we reveal the importance of modeling temporal evolutionary effects of historical relational interactions. Based on these observations, we devise KDA to adaptively incorporate item relations and their temporal evolutionary effects. Specifically, we aggregate the history sequence into relational dynamic history embeddings where the attention weight considers both relational intensities and time intervals between

historical items and the target item. For one thing, relational intensity is used to model relation-specific impacts of previous relational items. For another, we innovatively introduce a Fourier-based method to estimate the temporal evolution of different relational effects. The latent and continuous temporal decay functions are encoded to fixed-size embeddings in the frequency domain, which can be optimized together with other model parameters. Besides, the self-attention mechanism is leveraged to capture the mutual influence between relations. As a result, the final representation of the history sequence well reflects dynamic user intention. Extensive experiments are conducted in three datasets with different application scenarios. The experimental results indicate that our KDA model consistently and significantly outperforms state-of-the-art recommendation methods.

It is noteworthy that our designs of relational intensity and frequency-domain embeddings both contribute a lot to the performance improvement. The relational intensity helps our model to softly aggregate the history sequence and fully exploit the knowledge learned in the KG embedding task. More importantly, the Fourier-based estimation of temporal decay functions opens up a new avenue to adaptively introduce temporal dynamics into the model design, which is easy to be adopted in general neural models. In the future, we will extend our settings to include general external knowledge and further explore the integration of long-term user preference.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv:1607.06450*. Retrieved from https://arxiv.org/abs/1607.06450.

[2] AD Baddeley. 1968. Prior recall of newly learned items and the recency effect in free recall.*Can. J. Psychol./Rev. can. psychol.* 22, 3 (1968), 157.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*. Retrieved from https://arxiv.org/abs/1409.0473.

[4] 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.

[5] 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.

[6] Ronald Newbold Bracewell and Ronald N. Bracewell. 1986. *The Fourier Transform and Its Applications*. Vol. 31999. McGraw–Hill, New York, NY.

[7] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of the World Wide Web Conference*. 151–161.

[8] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2019. Social attentional memory network: Modeling aspect- and friend-level differences in recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 177–185.

[9] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 225–234.

[10] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Trans. Inf. Syst.* 38, 2 (2020), 1–28.

[11] 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.

[12] 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:1406.1078*. Retrieved from https://arxiv.org/abs/1406.1078.

[13] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. 485–492.

[14] 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.

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

[16] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep learning-based sequential recommender systems: Concepts, algorithms, and evaluations. In *Proceedings of the International Conference on Web Engineering*. Springer, 574–577.

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

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[19] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2018. Translation-based recommendation: A scalable method for modeling sequential behavior. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'18)*. 5264–5268.

[20] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM'16)*. IEEE, 191–200.

[21] 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*. 507–517.

[22] Xiangnan He and Tat-Seng Chua. 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.

[23] 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.

[24] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[25] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv:1511.06939*. Retrieved from https://arxiv.org/abs/1511.06939.

[26] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 263–272.

[27] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.

[28] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[29] 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*. 659–667.

[30] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM'18)*. IEEE, 197–206.

[31] 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.

[32] 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.

[33] 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*. 426–434.

[34] 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.

[35] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8, 42 (2009), 30–37.

[36] Guokun Lai, Wei Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long- and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 95–104.

[37] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[38] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 322–330.

[39] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

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

[41] 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:1903.03714*. Retrieved from https://arxiv.org/abs/1903.03714.

[42] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52.

[43] 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.

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

[45] 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.

[46] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[47] 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.

[48] 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.

[49] Pablo Sánchez and Alejandro Bellogín. 2020. Time and sequence awareness in similarity metrics for recommendation. *Inf. Process. Manage.* 57, 3 (2020), 102228.

[50] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *J. Mach. Learn. Res.* 6 (September 2005), 1265–1295.

[51] 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.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

[53] 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.

[54] 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.

[55] Chenyang Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2020. Make it a chrous: Knowledge- and time-aware item modeling for sequential recommendation. In *Proceedings of the 43th International ACM SIGIR Conference*. ACM.

[56] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In *Proceedings of the World Wide Web Conference*. ACM, 1977–1987.

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

[58] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*.

[59] Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.* 23, 1 (1996), 69–101.

[60] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon Jose. 2019. Relational collaborative filtering: Modeling multiple item relations for recommendation. *arXiv:1904.12796*. Retrieved from https://arxiv.org/abs/1904.12796.

[61] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*. 2048–2057.

[62] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv:1412.6575*. Retrieved from https://arxiv.org/abs/1412.6575.

[63] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5709–5716.

[64] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 582–590.

[65] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 353–362.

[66] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv:1803.06540*. Retrieved from https://arxiv.org/abs/1803.06540.

Received June 2020; revised September 2020; accepted October 2020