# Knowledge Graph Attention for Sequential Recommendations

MEHRNAZ AMJADI*, NVIDIA, USA
SEYED DANIAL MOHSENI TAHERI* and THEJA TULABANDHULA, University of Illinois at Chicago, USA

Sequential recommendation systems model dynamic preferences of users based on their historical interactions with platforms. Despite recent progress, modeling short-term and long-term behavior of users in such systems is nontrivial and challenging. To address this, we present a solution enhanced by a knowledge graph called KATRec (Knowledge Aware aTtentive sequential Recommendations). KATRec learns the short and long-term interests of users by modeling their sequence of interacted items and leveraging pre-existing side information through a knowledge graph attention network. Our novel knowledge graph-enhanced sequential recommender contains item multi-relations at the entity-level and users' dynamic sequences at the item-level. KATRec improves item representation learning by considering higher-order connections and incorporating them in user preference representation while recommending the next item. Experiments on three public datasets show that KATRec outperforms state-of-the-art recommendation models and demonstrates the importance of modeling both temporal and side information to achieve high-quality recommendations.

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

Additional Key Words and Phrases: sequential recommendations, knowledge graphs, deep neural networks, attention mechanism

**ACM Reference Format:**
Mehrnaz Amjadi, Seyed Danial Mohseni Taheri, and Theja Tulabandhula. 2021. Knowledge Graph Attention for Sequential Recommendations. 1, 1 (September 2021), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

With the exploding growth of online platforms in recent years, recommendation systems have become an essential component in elevating user engagement levels and thus have taken a central role in business success. Many services leverage historical data of users and their interactions with their service (e.g., an app or a website) to personalize recommendations. Such recommendation systems are increasingly popular in various domains including: e-commerce, social media, search engines, content portals, and online publishing platforms.

In this work, we focus on exploiting the sequential behavior of users in order to predict their upcoming interactions. Existing sequential recommender designs, e.g., Markov chains, recurrent neural networks (RNN), graph convolutional neural networks (GCN), and self-attention based models (to name a few), primarily focus on various ways to model such historical data. However, these sequential models tend to disregard the relationship between items. In particular, platforms usually have access to two types of information that can be valuable for recommendations: (i) interactions of users and the service, which may evolve over time, and (ii) side information about users, items, and other auxiliary

*Both authors contributed equally to this research.

Authors' addresses: Mehrnaz Amjadi, mamjadi@nvidia.com, NVIDIA, 2701 San Tomas Expwy, Santa Clara, California, USA, 95050; Seyed Danial Mohseni Taheri, smohse3@uic.edu; Theja Tulabandhula, theja@uic.edu, University of Illinois at Chicago, 1200 W Harrison St, Chicago, Illinois, USA, 60607.

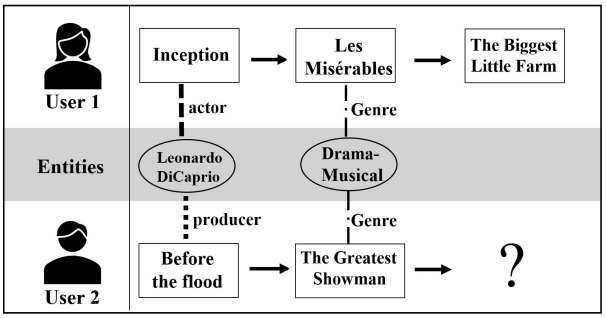



Fig. 1. Sequential interactions of two users with the systems. While users interact with different items, their collaborative signal can be detected via shared entities.

components. Examples of such side information for a movie, as an item, are genre, director, actors, etc. We build the knowledge graph by including these two types of information in the graph. Users, items, and other entities are the nodes, and the interactions and other relations are the edges of the knowledge graph. Recommender systems can generate more relevant content by taking advantage of item relations that are hard to elicit from interaction sequences of users. Such side information about the items can be based on higher-order item-entity connections and co-occurrence patterns, which can provide implicit information about related items (for instance, a mouse and a laptop).

Efficient exploitation of both temporal data of users and side information is the primary gap this paper attempts to fill. While there are many well-performing solutions in the literature, they do not effectively capture both types of information. For instance, models such as BERT4Rec, SASRec, and GRU4Rec [9, 14, 24] heavily focus on the temporal aspect by encoding user behavior sequences. Another stream of literature focuses on graph structure to capture item relations and side information, for example: TransE [1], TransH [31], and TransR [18]. In this work, we build on both these prior works and provide a novel way to integrate them. We follow this up by systematically exploring the importance of capturing both types of information on recommendation quality.

To capture the short-term preferences, long-term interests, and item-item relations, we propose the Knowledge Aware aTtentive Sequential Recommendations (KATRec) system. Our recommendation system consists of two modules: (i) a bidirectional transformer, which captures sequential interests by considering the inter-dependencies among items at any temporal distance, and (ii) a knowledge graph attention network that models higher-order user-item and item-item relations. The user-item relations are based on the interactions of users with items, e.g., click, purchase, view, etc., and capture *collaborative information*. The item-item relations capture the *semantic relatedness* among items based on their shared entities (e.g., movies with the same actor or genre). The importance of capturing such information while making sequential recommendations has been previously discussed in Ji et al. [13], Ma et al. [19], Zhang et al. [34] to name a few. In addition to first-order relations, sequential recommendations generated by KATRec can use higher-order connections and relations among items (e.g., an individual can be an actor in a movie and a producer in another). Figure 1 illustrates two sequences of user interactions. A traditional sequential recommendation system models these two users differently, as they interact with different movies, although these users show similar interests to movies with shared entities (genre, actor, producer, etc.). Therefore, incorporating information using a knowledge graph can enhance their representations, and consequently improve recommendation performance Wang et al. [29], Zhao et al. [35].

This work proposes a novel deep neural network architecture incorporating both sequential behavior of users and side information about items. Our proposed structure captures the temporal information using a sequential attention

mechanism and spatial information via a knowledge graph attention mechanism. To summarize, the key contributions of this work are:

- KATRec builds on a knowledge graph neural network that captures multi-relationships between items by tying them together using the underlying entities. This allows for better representations of items and enhances the recommendation performance.
- KATRec models the short-term and long-term user preferences by adaptively aggregating dynamic interactions and item-item multi-relations through a gating mechanism. This mechanism can significantly alleviate the sparsity in both user sequences and item relationships.
- KATRec captures co-occurrence information and collaborative signals by leveraging attention mechanisms in the knowledge graph, which ultimately impact the attention weights in the bidirectional transformer module and the overall recommendations.
- We conduct experiments on GPU to optimize training and evaluate the impact of different components on the performance of KATRec. Our experiments show that our model outperforms state-of-the-art baselines on three public datasets.

The paper is organized as follows: in Section 2, we review the relevant prior literature. Section 3 defines the problem and presents the new architecture KATRec. We conduct a detailed comparison of the performance of KATRec with multiple competitive baselines in Section 4. Section 5 concludes with some pointers to future directions.

## 2 LITERATURE REVIEW

### 2.1 General Recommendations

Early work on recommendation systems uses collaborative filtering (CF) to make recommendations based on common interests between users [16], which can be modeled using either explicit or implicit ratings [11]. CF based models usually suffer from the sparsity of the user-item interactions, which makes them susceptible to the cold start problem. To address this issue, newer variants of matrix factorization, including point-wise and pair-wise methods, have been proposed [5, 21].

Recent advances in deep neural networks (DNNs) have resulted in the development of recommendation models that have higher flexibility in learning user and item representations [33]. Among several recent works in this area, [32] and [7] use conventional matrix factorization with DNNs to predict ratings using autoencoders, and also predict user preferences using certain multi-layer perceptron architectures.

### 2.2 Sequential Recommendations

Given a user's temporally structured historical interaction logs, sequential recommendations predict the next item that a user is likely to adopt. Early works incorporated Markov chains to capture the underlying temporal behavior. For example, [5, 22] look at the most recently interacted item(s) to predict the next item to recommend. Newer methods have incorporated higher-order Markov chains to capture a better representation of user behavior. For instance, the Convolutional Sequence Model (Caser) in [25] learns the sequential patterns of a user via an embedding matrix of a fixed size that captures the previous $L$ items, using it as an image input and leveraging convolutional filters. A collection of works has similarly adopted RNNs to encode temporal user preferences into a context vector [3, 8, 9]. The attention mechanism has recently been used as a key modeling component for time series data in natural language processing, including in architectures such as the transformer [26] and BERT [2]. Inspired by these initial works, [14] proposes a

two-layer transformer decoder, namely SASRec, to capture sequential user behavior. [24] encodes user preferences with a bidirectional model using the Cloze task setup. While all these models encode sequential preferences of users, they have limited power in capturing the relationships between items themselves and between items and users in a parsimonious way. Our proposed approach aggregates higher-order and multi-relation item connections and dynamic preferences of users to predict the next item while retaining modular interpretability.

### 2.3 Knowledge-aware Recommendations

There have been multiple prior works that focus on enhancing recommendation system performance by explicitly considering multi-relations between items [4, 20, 28]. These models usually leverage the structure of an associated knowledge graph and a static user-item interaction graph to exploit such higher-order relationships, typically through path selection algorithms [17, 30] and meta-paths heuristics [10, 12]. However, these approaches can be sub-optimal for the recommendation objective, and may requires a reasonably high degree of domain knowledge, making them less suitable for generating highly relevant recommendations. A recent work focuses on regularizing the recommendation objective by adding a loss that captures the knowledge graph structure [27, 29]. Due to certain modeling choices, it is not fully clear if higher-order connectivity is being captured effectively, while retaining interpretability. In contrast to these methods, some of which are focused on generic graphs, we incorporate a knowledge graph while taking the higher-order connectivity between entities and their neighbors into account in a parsimonious way, and fuse these signals into an attention-based sequential model that captures dynamic preferences of a user.

## 3 METHOD

Our goal is to provide a personalized next item recommendation for users based on their history and higher-order relations between items. In this section, we first state a formal definition of the problem, and then we elaborate on different parts of our proposed solution.

### 3.1 Problem Definition and Solution Overview

Given a set of users $\mathcal{U}$ and items $\mathcal{I}$, we have a sequence of items $\mathcal{S}^u = \{S_1^u, \cdots, S_T^u\}$ that user $u$ has interacted with over $T$ time steps ($T = |\mathcal{S}^u|$). Also, we have access to side information related to items (e.g., actors, directors, and genre as shown in Figure 1). Based on the historical interaction sequence $\mathcal{S}^u$, our goal is to predict the item that user $u$ is most likely interested in at the next time step $T + 1$.

In KATRec, we build a knowledge-aware attentive sequential recommendation system to facilitate modeling of dynamic behavior of users while capturing the multi-relations between items. Specifically, our model contains two modules as shown in Figure 2, namely: (a) a graph neural network $\mathcal{G}$ which captures item level multi-relations, and (b) a bidirectional transformer module that incorporates item embeddings from the knowledge graph network into the representations of dynamic preferences exhibited by the users. In the following, we discuss the details of each.

### 3.2 Knowledge Graph Module with Attention

This module encodes items' metadata as a unified graph to exploit the higher-order connectivity between items. The graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{I} \subset \mathcal{E}$, incorporates entities as nodes and relationships as the edges. For instance, entities in a movie data set include the items and their side information e.g., genres, producer, actor, etc. We use entities as building blocks that capture connections between items, and we focus on paths that start and end with items. Formally, the $K$-order connectivity between items is a path that captures a higher-order relationship between items $(i, j)$ as:

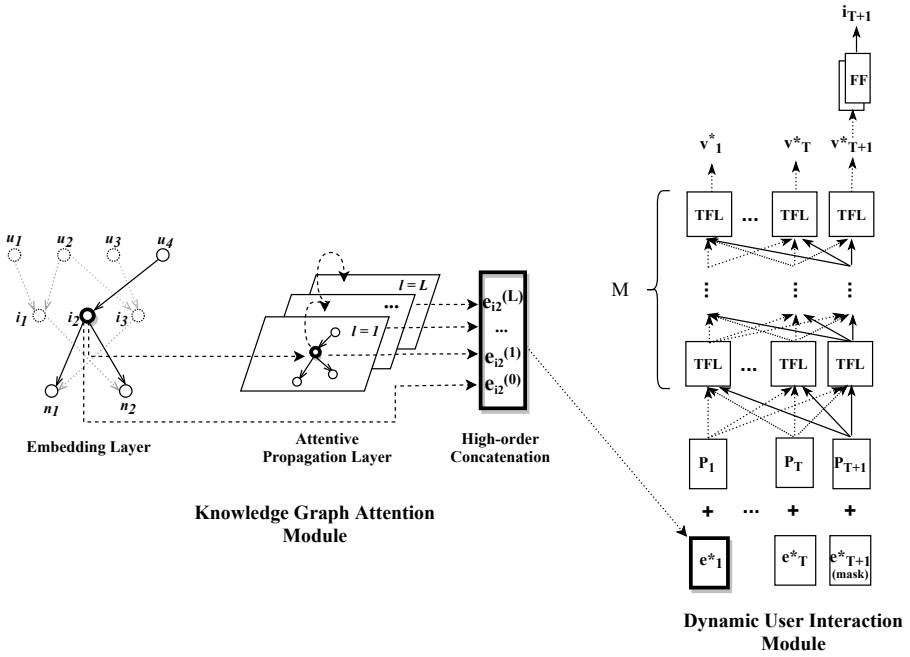



Fig. 2. Illustration of different components of the knowledge graph attention module (left) and the dynamic user interaction module (right). First, we learn the initial item embeddings using the knowledge graph attention module. Then, we feed these item embeddings and the user's interaction sequence $\mathcal{S}^u$ through $M$ layers of the bidirectional transformer (TFL) to obtain the final item embeddings that are then used for next item recommendation.

$i \xrightarrow{r_1} n_1 \xrightarrow{r_2} n_2 \xrightarrow{r_3} \cdots \xrightarrow{r_K} j$, where $(i, j) \in \mathcal{I}$, entity $n_k \in \mathcal{E}$, and relation $r_k \in \mathcal{R}$ for $k \in \{1, 2, \cdots, K\}$. In addition to item and entities in the paths, we also model the joint occurrence of commonly related items using a collaborative knowledge graph by adding users as nodes in the graph. In particular, we integrate the item-user relations $\mathcal{R}_{\mathcal{U}}$ into the knowledge graph, so in the resulting graph, the interactions of users with items are also captured. We can represent the graph by a pair of nodes and their relation as $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' \subset \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' \subset \mathcal{R} \cup \mathcal{R}_{\mathcal{U}}$. Figure 3 illustrates a collaborative knowledge graph where movie $i$ and movie $j$ are second order neighbors related by entity $n_1$ in the (undirected) path $i \xrightarrow{r_4} n_1 \xrightarrow{r_5} j$ or related by user $u_2$ in the path $i \xrightarrow{r_3} u_2 \xrightarrow{r_2} j$. In order to encode these relations as item embeddings, we use the TransR method [18]. TransR learns the embedding of each node and relation via the translation principle: $W^r e_h + e_r \approx W^r e_t$ ($e_h, e_t \in \mathbb{R}^d$, $e_r \in \mathbb{R}^k$, and $W^r \in \mathbb{R}^{k \times d}$), if triplet $(h, r, t)$ exists in the knowledge graph. For each triplet $(h, r, t)$ the dissimilarity score is computed using:

$$s(h, r, t) = \|W^r e_h + e_r - W^r e_t\|_2^2,$$

The lower the score of $s(h, r, t)$ the more likely is the triplet in the KG. Following [29], we capture each relation's importance by generating attentive weights between a node and its higher-order neighbors. So, for each node $h$, we initially consider all nodes that have the first-order relation with it, i.e.,

$$\mathcal{N}_h = \{(h', r, t)|(h', r, t) \in \mathcal{G} \text{ with } h' = h\}.$$

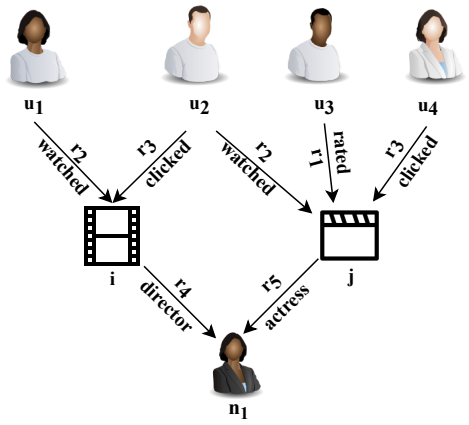



Fig. 3. Collaborative knowledge graph relates items $i$ and $j$ through user $u_2$ and entity $n_1$ with different types of relations.

The first order connectivity embedding of head node $h$ is defined as the linear combination of the embeddings of its ego network:

$$e_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)e_t,$$

where attention factor $\pi(h,r,t)$ controls how much information from different tails can be propagated to head $h$ based on specific relations, and can be computed as follows:

$$\pi(h,r,t) = (W^r e_t)^\top \tanh(W^r e_h + e_r).$$

This attention mechanism will propagate more information from closer entities in the relationship space. Then, we normalize the coefficients across all $h$'s first-order relations using a softmax function:

$$\pi'(h,r,t) = \frac{\exp(\pi(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h}\exp(\pi(h,r',t'))}.$$

We update the node's representation by aggregating its representation and its ego network/connectivity representation using the relation: $e_h^{(1)} = f_{\mathcal{G}}^{(1)}(e_h, e_{\mathcal{N}_h})$, where aggregator $f_{\mathcal{G}}^{(1)}$ is defined as follows:

$$f_{\mathcal{G}}^{(1)} = \sigma\Big(W_1^{(1)}(e_h + e_{\mathcal{N}_h})\Big) + \sigma\Big(W_2^{(1)}(e_h \odot e_{\mathcal{N}_h})\Big).$$

Here $\sigma$ is the Leaky ReLU activation function and $W_1^{(1)}, W_2^{(1)} \in \mathbb{R}^{d^{(1)}} \times \mathbb{R}^d$ are trainable weight matrices. We follow a similar intuition for residual connections by aggregating information through the sum of two representations $e_h$ and $e_{\mathcal{N}_h}$ and retain a copy of input by using the identity transformation. Note that $\odot$ is the element-wise product that captures feature interaction between $e_h$ and $e_{\mathcal{N}_h}$, and ensures richer propagation of information from similar nodes.

For higher-order propagation, we stack more propagation layers to cascade information from higher-order neighbors. The $l$-th step node representation can be formulated as:

$$e_h^{(l)} = f_{\mathcal{G}}^{(l)}(e_h^{(l-1)}, e_{\mathcal{N}_h}^{(l-1)}),$$

where the information cascaded from $l-1$-th ego network is defined as:

$$e_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi'(h,r,t) e_t^{(l-1)}.$$

Using this embedding propagation mechanism to stack $L$ layers, the higher-order connectivities can be captured in the node representation. Finally, we concatenate these representations into one vector to get the final representation of the node.

$$e_h^* = [e_h^{(0)} \parallel \cdots \parallel e_h^{(L)}] \in \mathbb{R}^q,$$

where $e_h^{(0)} := e_h$ and $q = d + d^{(1)} + \cdots + d^{(L)}$. The different layers of knowledge graph attention module described above are shown in Figure 2 (left).

### 3.3 Dynamic User Interaction Module

To capture the sequential patterns among successive items that a user has interacted with, we use the Bidirectional Encoder Representations from Transformers (BERT) architecture [2, 26]. In our context, BERT uses the historical item sequence $\mathcal{S}^u$ corresponding to user $u$ and aims to predict the item that the user is interested in the next time step $T+1$. BERT models $M$ bidirectional transformer layers and revises each item's representation at each layer by exchanging information across all positions at the previous layer. Our key contribution is that we embed the higher-order connectivity of the item relations discussed in subsection 3.2 into the BERT module to capture user preferences under a more informative context. Below, we briefly discuss the self-attention structure used in the BERT module of KATRec.

*3.3.1 Embedding Layer.* Positional Embedding: Since the self-attention mechanism doesn't include any recurrent or convolutional blocks, it cannot be aware of items' position embeddings. So, we incorporate a pre-determined positional embedding $P \in \mathbb{R}^{T\times q}$ into the input embedding:

$$v_i^{(0)} = e_i^* + p_i,$$

where $v_i^{(0)}$ computes the input representation of items at each position index $i = 1, \cdots, T$. Concatenating embedding of $T$ items in a user sequence, $v_i^{(0)} \in \mathbb{R}^q$, results in $V^{(0)} \in \mathbb{R}^{T\times q}$. Next, this positional embedding matrix is provided as an input to the first transformer layer. The transformer layer contains two sublayers, a multi-head self-attention sublayer, and a position-wise feed-forward network. In the following, we describe each of these sublayers briefly.

*3.3.2 Transformer Layer.* Multi-Head Self-Attention: The attention mechanism helps the model capture dependencies between each pair of items at any distance in the input sequence, across multiple subspace representations simultaneously. To learn information in different representation subspaces, we use multi-head attention [2]. First, we linearly project $V^{(0)}$ into $k$ subspaces using different learnable projections and then apply attention function on each in parallel to create $k$ heads $H_1, \cdots, H_k$ as follows:

$$H_j = \text{Attention}(V^{(0)}W_j^{\mathcal{Q}}, V^{(0)}W_j^{\mathcal{K}}, V^{(0)}W_j^{\mathcal{V}}),$$

where $W_j^{\mathcal{Q}}$, $W_j^{\mathcal{K}}$ and $W_j^{\mathcal{V}}$ are all $\mathbb{R}^{q\times q/k}$ learnable projection matrices corresponding to head index $j = 1, \cdots, k$. The attention function is a scaled dot-product defined as $\text{Attention}(\mathcal{Q},\mathcal{K},\mathcal{V}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^\top}{\sqrt{q/k}}\right)\mathcal{V}$. To compute the multi-head attention output, we concatenate these $k$ heads and then project it as follows:

$$\text{MH}(V^{(0)}) = [H_1 \parallel H_2 \parallel \cdots \parallel H_k]W^O,$$

where the learnable projection matrix $W^O \in \mathbb{R}^{q \times q}$. Next, we input $\text{MH}(V^{(0)})$ to a feed-forward sublayer described below.

Position-wise Feed-Forward Network: Since the self-attention sub-layer mainly leverages linear transformations, we need another sub-layer to implement non-linearity and also capture interactions between different dimensions. We use a position-wise feed-forward network on the outputs of the multi-head self-attention sublayer at each position $i = 1, \cdots, T$.

$$\text{PFF}(V^{(0)}) = [FF(v_1^{(0)})^\top \parallel \cdots \parallel FF(v_T^{(0)})^\top]$$

where $FF(x) = \text{GELU}(xW^1 + b^1)W^2 + b^2$ uses the GELU non-linearity, and $W^1, W^{2\top} \in \mathbb{R}^{q \times 4q}$, $b^1 \in \mathbb{R}^{4q}$ and $b^2 \in \mathbb{R}^q$ are learnable and shared at different positions.

Furthermore, we apply dropout to the output of both multi-head self-attention and position-wise feed-forward sub-layers. Due to the architecture depth, we also add residual connections to capture item-item interactions better. Finally, we apply layer normalization (LN) to stabilize and accelerate training. These operations can be summarized as $\text{LN}(x + \text{Dropout}(\text{sublayer}(x)))$. In order to learn a more complex representation of items in the sequence, we stack $M$ Transformer layers as illustrated in Figure 2. Again, we note that the above choices are informed by the original BERT architecture for language modeling.

*3.3.3 Output Layer.* Item Co-occurrence Modeling: Capturing pairwise item relations plays an important role in the effectiveness of the recommendation systems and also allows some degree of interpretability. To include item-item relations in item embeddings, we concatenate item embeddings learned by the sequential module and the knowledge graph.

$$\hat{E} = \sigma((V^* \parallel E^*)\hat{W} + \hat{b}),$$

where $\hat{E} \in \mathbb{R}^{|\mathcal{I}| \times q}$ is the set of final KATRec embeddings for item set $\mathcal{I}$ and $\hat{W} \in \mathbb{R}^{2q \times q}$ and $\hat{b} \in \mathbb{R}^q$ are learnable parameters. $V^*$ is the embedding matrix of items learned by sequential module, and $E^*$ is the items embedding table learned by the knowledge graph. Finally, the next item at time $T + 1$ is predicted by:

$$P(\mathcal{I}) = \text{softmax}(\text{GELU}(v_{T+1}^{(M)} W^P + b^P)\hat{E}^\top + b^O),$$

where $W^P, b^P$, and $b^O$ are learnable parameters, $\hat{E} \in \mathbb{R}^{|\mathcal{I}| \times q}$ is the embedding matrix for item set $\mathcal{I}$, and $P(\cdot)$ is the KATRec model's predicted distribution over the target items. $v_{T+1}^{(M)}$ is the hidden state of position $T + 1$ after $M$ transformer layers, denoted by $v_{T+1}^*$ in figure 2.

Explicit User Modeling: Existing approaches provide personalized recommendation by modeling the user embedding either explicitly [16, 22, 25] based on users' previous actions, or implicitly [8, 14, 24] based on embeddings of the sequence of visited items by a user. KATRec belongs to the latter category as we predict the next item at time step $T + 1$ by considering the hidden state embedding $v_{T+1}^{(M)}$.

As an aside, we also considered explicit user behavior by incorporating user embeddings learned from the knowledge graph into the user's hidden state via concatenation: $[e_u^* \parallel v_{T+1}^{(M)}]$, where $e_u^*$ is the embedding of user $u$ in the knowledge graph. Although this concatenation seems promising, we empirically did not observe improvement in the model's performance. This could be potentially because the model learned users' embedding very well by considering the sequence of interacted items.

### 3.4 Optimization

The loss function of KATRec contains the TransR objective along with a regularizer. In particular, we use the TransR to train the entity embeddings. The objective function can be minimized by discriminating between valid and invalid triplets in the collaborative knowledge graph:

$$\mathcal{L}_{\mathcal{G}} = \sum_{(h,r,t,t')} -\ln \sigma\Big(s(h,r,t') - s(h,r,t)\Big) + \lambda \|\Theta\|_2^2, \tag{1}$$

where the sum is over all valid $(h,r,t) \in \mathcal{G}$ and invalid $(h,r,t') \notin \mathcal{G}$ triplets in the knowledge graph $\mathcal{G}$, $\sigma(\cdot)$ is the sigmoid function, and $\lambda\|\Theta\|_2^2$ represents $\ell_2$ regularization. Similar to [2], we implement the *Cloze task* training approach in addition to pairwise ranking loss in equation (1). This allows us to learn the parameters of the encoders in the transformer layers. In this approach, we randomly mask a portion of items in the input sequence $\mathcal{S}_u$ and try to predict them. The loss for each masked input $S'_u$ is given by:

$$\mathcal{L}_{\mathcal{S}} = \frac{1}{|\mathcal{S}_u^m|} \sum_{i_m \in \mathcal{S}_u^m} -\log P(i_m = i_m^{\star} | \mathcal{S}'_u),$$

where $\mathcal{S}_u^m$ is the set of randomly masked items, $i_m^{\star}$ is the true item corresponding to the masked item $i_m$, and $P(.)$ is the predicted probability mass function over the target item. These two losses are jointly minimized over their respective parameters using standard first-order approaches (see the next section for details).

## 4 EXPERIMENTS

We evaluate our model on three real-world datasets, which are different in domains and have varying levels of sparsity[1]. We aim to answer the following questions in this section:

Q1: How does KATRec perform compared to the current state-of-the-art sequential recommendation methods?

Q2: How do different components of the model (viz., knowledge graph based attention mechanism, information aggregation, and pre-training) affect the performance of KATRec?

Q3: How KATRec's performance change in different settings and datasets?

Following prior work [14, 24], we convert all numeric ratings to positive interaction with a value of 1, which indicates that the user has interacted with the item. Then, we sort each user's interactions by timestamp to build her interaction sequence. Similar to prior works, we split the user sequences into three parts. The test dataset includes the most recent item each user has interacted with ($S^u_{T+1}$), the validation dataset consists of the second most recent item interacted by each user ($S^u_T$), and the remaining items in the sequence belong to the training data. To construct knowledge graph aware attention, we first build the item knowledge graph for each dataset. We follow [29, 35] to capture knowledge graph triplets by mapping items into *freebase* entities. We include triplets with one-hop and two-hop neighbor entities and filter out entities with less than ten occurrences, and relations less than fifty occurrences. For $\mathcal{L}_{\mathcal{G}}$ in equation (1), we pair each observed triplet with a broken (unobserved) triplet.

### 4.1 Datasets Description

We consider three datasets with different levels of sparsity. For most of the experiments, we only include users and items with at least ten interactions to ensure data quality [29]. The statistics of the datasets described below are presented in Table 1 for ease of reference:

[1]Code is available at https://github.com/DanialTaheri/KATRec

Table 1. Statistics of datasets

| **Datsets** | Users | Items | Interactions | Entities | Relations | Triplets | Density |
|---|---|---|---|---|---|---|---|
| Amazon-book | 70679 | 24915 | 846434 | 88572 | 39 | 2557746 | 0.048% |
| LastFM | 23566 | 48123 | 8057269 | 58266 | 9 | 464567 | 0.7105% |
| Yelp2018 | 45919 | 45538 | 1185068 | 90961 | 42 | 1853704 | 0.057% |

Amazon-book: Amazon review data is one of the popular datasets in the recommendation systems literature [6]. The data has been categorized based on different product categories, and in this paper, we focus on the book category.
LastFM: This is a dataset about music listening patterns collected from the Last.fm online music platform [23]. In this dataset, tracks are viewed as items, and we consider a subset of the dataset from January 2014 to August 2014.
Yelp2018: This dataset is adopted from the Yelp 2018 recommendation system challenge. In this dataset, local restaurants and bars are represented as items.
We calculate each dataset's density based on the number of interactions, users, and items as $\frac{|Interactions|}{|Users|\cdot|Items|}$. Therefore, larger values in the density column represent datasets with more interactions (per user and item). Table 1 lists specific properties and the density of each dataset. It also highlights that LastFM has a substantially lesser number of relations and is significantly denser than others.

### 4.2 Experimental Settings

*4.2.1 Evaluation Metrics.* We use two common top-K metrics to evaluate the performance of our model. Hit@K and NDCG@K count the fraction of the time the ground truth item is among the top $K$ recommendations, without and with defining a position-aware weight respectively. Mean Average Precision (MAP) is also a ranked precision metric over all users that emphasizes correct predictions at the top of the list with a position-aware weight. Given a list of top $K$ predicted items $\hat{S}^u_{1:K}$ for a representative user $u$, and the ground truth last item $S^u_{T+1}$, Hit@K, NDCG@K, and MAP are computed as:

$$Hit@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}} |\hat{S}^u_{1:K} \bigcap S^u_{T+1}|,$$

$$NDCG@K = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{i=1}^{K}\frac{rel_i}{\log_2(i+1)},$$

$$MAP = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\sum_{i=1}^{|\hat{S}^u|} Prec@i \times rel_i,$$

where $rel_i = 1$ if the $i^{th}$ item in $\hat{S}^u_{1:K}$ equals $S^u_{T+1}$, and $Prec@i$ is the precision (fraction of recommendations that end up being correct) up till position $i$. Following the works of [14, 24], we randomly sample 100 negative items for each user besides the ground-truth item. We report the average of the metrics Hit@K, NDCG@K over all users.

*4.2.2 Baselines.* To compare the performance of our model with others, we consider the following competitive baselines:

- GRU4Rec [9]: It implements a session-based recommendation model based on RNNs. We consider each user's sequence as a session.
- GRU4Rec$^{++}$ [8]: It modifies the way GRU4Rec is optimized by implementing a new loss function and a new sampling approach.

- SASRec [14]: It uses a self-attention mechanism with a left-to-right Transformer to improve the capturing of useful patterns in user sequences.
- BERT4Rec [24]: This is a recent state-of-the-art sequential recommendation model that adapts the bidirectional Transformers language model architecture to learn the temporal behavior of users.

*4.2.3 Parameters for Models.* We implement KATRec with Tensorflow (version 2.2.0). All parameters are initialized using a truncated normal distribution in the interval $[-0.02, 0.02]$. We use the Adam optimiser [15] with learning rate $10^{-4}$ that decays linearly, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. We fix the maximum sequence length proportional to the average sequence length in the dataset, i.e., 50, 50, and 200 for Amazon-book, Yelp, and LastFM respectively. We set the dimension of the hidden fully connected layers of KATRec to be 128. We propagate neighbors' information up to three levels into each entity's embedding with hidden dimensions 32, 16, and 16. Finally, we set the embedding of entities in the knowledge graph to be 64.

We search for hyperparameters to select the best parameters for different baselines. These include changing embedding size from $\{8, 16, 32, 64, 128\}$ and the regularization hyperparameter $\lambda$ across $\{0.1, 0.05, 0.01, 0.005, 0.001, 0.0001\}$. We use the optimization schemes and parameters suggested by the authors whenever possible. The models are trained on a single GeForce GTX 1080Ti GPU [2].

*4.2.4 Performance Comparison.* Table 2 shows the recommendation performance of KATRec and baselines. We do not include Hit@1 since Hit@1 and NDCG@1 are equivalent. Since we have a single ground-truth, Hit@K is equivalent to Recall@K, and it is proportional to Precision@K. *We observe that KATRec provides improved relative recommendation performance over all alternatives by 6.29% and 3.82% in Hit and 7.15% and 4.64% in NDCG on average, respectively on the Amazon-book and Yelp2018 datasets.*

In particular, KATRec consistently outperforms BERT4Rec on Amazon-book and Yelp2018 datasets, which shows the importance of modeling item-item and user-item relations. KATRec achieves a considerable performance improvement in Amazon-book, while the improvement in Yelp is relatively small. This observation can be attributed to the difference in the sparsity of these two datasets. The importance of the impact of data density and number of relations on KATRec's performance is highlighted explicitly with the LastFM dataset, which we discuss in further detail in the ablation study that follows.

*4.2.5 Ablation Study.* In this section, we analyze variants of KATRec to understand the impact of different components on model performance. The variations are as follows: (1) *No Attention*: we remove the attention mechanism in the knowledge graph and allocate equal weights to each entity's neighbors. (2) *Level-1*: we decrease the level of information that can propagate from the neighbors to a node, and study the impact of only using immediate neighbors to improve node embeddings. (3) *Connection*: While there exists a connection between two encoder modules in our model, we consider the setting where both modules train independently using learnable embeddings. (4) *No Pretraining*: We forgo the pre-trained embeddings of entities in the knowledge graph. (5) *Concat*: We remove a part of our model that deals with item co-occurrence, and only consider the item embedding vector that results from the sequential module.

Results are shown in Table 3. We observe that incorporating the side information in the bidirectional encoder module during the training results in better parameter learning as shown in column *Connection*. Furthermore, the results of making the *NoPretrain* choice in the knowledge graph show that incorporating pre-trained embeddings

[2]GeForce GTX 1080Ti GPU substantially accelerated the training process compared to CPU. To further speed up the process, A100 GPU and V100 GPU are ideal choices.

Table 2. KATRec versus baselines over three datasets. The improvement percentage compares KATRec versus the next-best alternative.

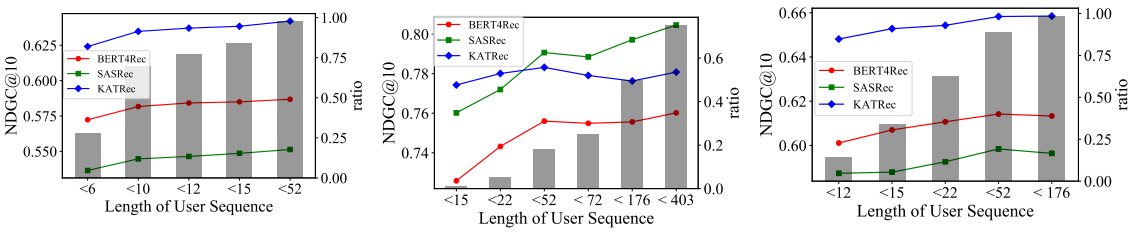
| Datasets | Metrics | GRU | $GRU^{++}$ | SASRec | BERT | KATRec | Improv. |
|---|---|---|---|---|---|---|---|
| Amazon | NDCG@1 | 0.3485 | 0.3464 | 0.3749 | 0.4344 | **0.4706** | 8.33% |
| | NDCG@5 | 0.4404 | 0.4358 | 0.5267 | 0.5715 | **0.6110** | 6.91% |
| | NDCG@10 | 0.4598 | 0.4574 | 0.5600 | 0.6022 | **0.6401** | 6.2% |
| | Hit@5 | 0.5202 | 0.5148 | 0.6594 | 0.6910 | **0.7321** | 5.94% |
| | Hit@10 | 0.58 | 0.5814 | 0.7621 | 0.7856 | **0.8217** | 4.6% |
| | MAP | 0.42 | 0.4259 | 0.5065 | 0.5539 | **0.5907** | 6.64% |
| LastFM | NDCG@1 | 0.3646 | 0.3523 | 0.6771 | 0.6339 | **0.6931** | 2.36% |
| | NDCG@5 | 0.4648 | 0.4448 | **0.7765** | 0.7606 | 0.7725 | -0.51% |
| | NDCG@10 | 0.4881 | 0.4674 | **0.7930** | 0.7786 | 0.7911 | -0.24% |
| | Hit@5 | 0.5531 | 0.5263 | **0.8600** | 0.8281 | 0.8426 | -2.06% |
| | Hit@10 | 0.6249 | 0.5958 | **0.9105** | 0.8836 | 0.9001 | -1.15% |
| | MAP | 0.4577 | 0.4357 | 0.7598 | 0.7509 | **0.7618** | 0.26% |
| Yelp2018 | NDCG@1 | 0.3946 | 0.4148 | 0.3723 | 0.4149 | **0.4405** | 6.17% |
| | NDCG@5 | 0.5041 | 0.5143 | 0.5703 | 0.6039 | **0.629** | 4.15% |
| | NDCG@10 | 0.5278 | 0.5395 | 0.6068 | 0.6400 | **0.663** | 3.6% |
| | Hit@5 | 0.5991 | 0.6021 | 0.7434 | 0.7690 | **0.7927** | 3.08% |
| | Hit@10 | 0.6721 | 0.68 | 0.8551 | 0.8796 | **0.899** | 2.2% |
| | MAP | 0.49 | 0.515 | 0.5351 | 0.5706 | **0.5946** | 4.21% |

Table 3. Ablation study of design choices in KATRec using three datasets.

| Datasets | Metrics | KATRec | NoAtten. | Level-1 | Connect. | NoPretrain | Concat. |
|---|---|---|---|---|---|---|---|
| Amazon-book | NDCG@10 | 0.6401 | 0.6371 | 0.6386 | 0.621 | 0.6306 | 0.6318 |
| | Hit@10 | 0.8217 | 0.8178 | 0.8195 | 0.801 | 0.8092 | 0.8142 |
| LastFM | NDCG@10 | 0.7911 | 0.7836 | 0.7853 | 0.763 | 0.7587 | 0.7855 |
| | HIT@10 | 0.9001 | 0.8967 | 0.8957 | 0.8796 | 0.8752 | 0.8908 |
| Yelp2018 | NDCG@10 | 0.663 | 0.6567 | 0.6546 | 0.6458 | 0.6359 | 0.6515 |
| | HIT@10 | 0.899 | 0.8954 | 0.8929 | 0.885 | 0.8696 | 0.8881 |

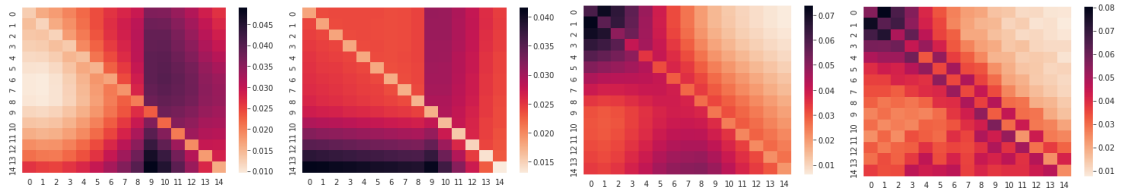
increases the performance of the sequential recommendation model. The attention mechanism between neighbors in the knowledge graph increases the recommendation's performance. However, this increase is not substantial. We also observe that multiple layer information propagation in an item's embedding plays an important role in making next item recommendations. Also, as expected, propagating the first layer's information in the knowledge graph has a higher impact, and this impact decreases when we incorporate higher level of connections. Finally, the results of the *Concat* choice shows that incorporating non-linearity in the final layer is beneficial for learning item embeddings, especially by combining features learned through the knowledge graph and the bidirectional encoder.

We also study the recommendation performance of KATRec for users with different sequence lengths and compare it with competitive baselines. The intuition here is that the use of side information can compensate for potentially sparse user item interactions. In Figure 4, we illustrate the percentage of users with varying sizes of item sequences associated with them in each dataset, and report each method's performance for each of the resulting user groups. KATRec outperforms two other competitive baselines for all user groups for Yelp2018 and Amazon-book datasets. Results for LastFM show that KATRec provides a reasonably robust recommendation performance across user groups,




(a) Amazon-book (b) LastFM (c) Yelp

Fig. 4. Performance comparison of models across users with different sequence lengths on the Amazon-book, LastFM and Yelp2018 datasets.



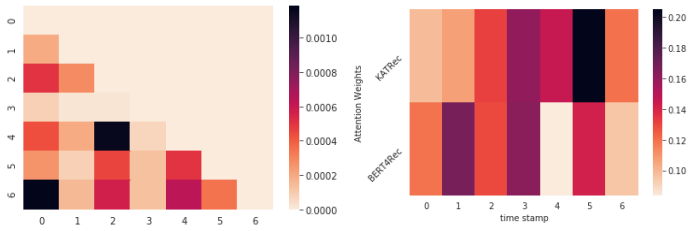
(a) KATRec, Layer 1, head 1 (b) KATRec, Layer 1, head 1, w/o PE (c) KATRec, Layer 2, head 2 (d) BERT4Rec, Layer 2, head 2

Fig. 5. Average attention weights on positions (x-axis) at different time (y-axis) on Yelp

while the performance of the baselines is more sensitive to the user sequence length. This robustness of KATRec can be attributed to the item-item and user-item relationships that are explicitly learned. Our model performs relatively better for users with small sequence lengths. However, SASRec marginally outperforms KATRec for users with long sequence lengths. This can be attributed to the high number of interactions and low number of relations in the LastFM dataset, which indicates that the impact of side information on dense datasets may not be significant enough.

*4.2.6 Visualizing attention weights.* This section visualizes the attention weights related to items and positions to find a meaningful pattern and discuss their differences with the BERT model's attention weights. Figure 5 shows the heatmaps of the average attention weights on the last 15 items of the sequences in the test dataset of Yelp2018. In order to calculate the accurate average weights, we do not incorporate weights of padded items in sequences shorter than 15 items. Comparing heatmaps $(a)$ and $(b)$ shows the impact of positional embeddings (PE). In particular, heatmap $(a)$ illustrates how positional embeddings results in items attending more on recent items. Heatmaps $(a)$ and $(c)$ points out how items in various heads and layers focus on different parts of the sequence at both the right and left sides. To compare attentions in BERT4Rec and KATRec, we analyze weights in the final layer as it is directly connected to the output layer and plays an important role in the prediction (heatmaps $(d)$ and $(c)$). The comparison indicates that while BERT4Rec inclines to focus more on the recent items due to the sparsity of the dataset, KATRec tends to attend on less recent items due to incorporating side information through the knowledge graph. This behavior is similar to the attention weights of self-attention blocks in dense datasets in [14].

*4.2.7 Attention Weight Case Study.* Figure 6a illustrates the co-occurrence ratio between six items in user sequences, computed by the average number of times that a pair of items appeared simultaneously in users' sequence. Figure




(a) Co-occurence information (b) Attention weights

Fig. 6. Attention heatmap comparison of a random user in Yelp2018 dataset. 6a shows the co-occurrence ratio, which is the co-occurrence frequency of each pair of six items among all users' sequences. 6b compares these items' attention weight at the last position in KATRec and BERT4Rec.

6b compares weights of the final attention layer in KATRec and Bert4Rec between the last item (item 6) and the rest of the items in a user sequence. Figure 6a shows that item 6 has a high co-occurrence value with items 0, 2, 4, and 5. Figure 6b confirms that KATRec places a higher attention values for items 4 and 5 and lower values for items with low co-occurrence value, i.e., time 1 and 3. However, we observe that BERT4Rec considers a higher weight between item 6 and 0 which is more aligned to their co-occurrence value.

## 5 CONCLUSION AND FUTURE WORK

Designing robust deep neural network architectures that produce quality recommendations is challenging for several reasons. A couple of these challenges were addressed in this work, namely leveraging of side information and getting around data sparsity. In particular, we proposed incorporating item side information to alleviate both these shortcomings while making recommendations. This information is readily available in many real-world applications.

Our work introduces a novel neural network structure that leverages collaborative knowledge graphs to improve the representations of items in a sequential recommendation system setup. Empirical results are provided to illustrate the benefit via multiple evaluation metrics: the proposed solution is compared against multiple state-of-the-art sequential recommendation systems on three different datasets. Similar to the way we included item metadata in building a more performant recommendation system, further research in incorporating user metadata can be undertaken.

## REFERENCES

[1] 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.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 152–160.

[4] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

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

[6] 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.

[7] 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*. 173–182.

[8] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations. (2018), 843–852. https://doi.org/10.1145/3269206.3271761

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

[10] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.

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

[12] Xiaowen Huang, Shengsheng Qian, Quan Fang, Jitao Sang, and Changsheng Xu. 2020. Meta-path Augmented Sequential Recommendation with Contextual Co-attention Network. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 2 (2020), 1–24.

[13] Mingi Ji, Weonyoung Joo, Kyungwoo Song, Yoon-Yeong Kim, and Il-Chul Moon. 2019. Sequential Recommendation with Relation-Aware Kernelized Self-Attention. *arXiv preprint arXiv:1911.06478* (2019).

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

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] 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.

[17] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive Path Reasoning on Graph for Conversational Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2073–2083.

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

[19] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2019. Memory Augmented Graph Neural Networks for Sequential Recommendation. *arXiv preprint arXiv:1912.11730* (2019).

[20] Chuan Qin, Hengshu Zhu, Fuzhen Zhuang, Qingyu Guo, Qi Zhang, Le Zhang, Chao Wang, Enhong Chen, and Hui Xiong. 2020. A survey on knowledge graph-based recommender systems. *SCIENTIA SINICA Informationis* 50, 7 (2020), 937–956.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[22] 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*. 811–820.

[23] Markus Schedl. 2016. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. 103–110.

[24] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[25] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[26] 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.

[27] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.

[28] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z Sheng, Mehmet A Orgun, Longbing Cao, Francesco Ricci, and Philip S Yu. 2021. Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339* (2021).

[29] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. 950–958.

[30] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

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

[32] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 153–162.

[33] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[34] Yin Zhang, Yun He, Jianling Wang, and James Caverlee. 2020. Adaptive Hierarchical Translation-based Sequential Recommendation. In *Proceedings of The Web Conference 2020*. 2984–2990.

[35] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2019. KB4Rec: A Data Set for Linking Knowledge Bases with Recommender Systems. *Data Intelligence* 1, 2 (2019), 121–136.