GInRec: A Gated Architecture for Inductive Recommendation using Knowledge Graphs

Theis E. Jendal[1], Matteo Lissandrini[1], Peter Dolog[1] and Katja Hose[1,2]

[1]{tjendal,matteo,dolog,khose}@cs.aau.dk – Aalborg University (AAU), Denmark
[2]katja.hose@tuwien.ac.at – Technische Universität Wien (TU Wien), Austria

Abstract
We have witnessed increasing interest in exploiting KGs to integrate contextual knowledge in recommender systems in addition to user-item interactions, e.g., ratings. Yet, most methods are transductive, i.e., they represent instances seen during training as low-dimensionality vectors but cannot do so for unseen instances. Hence, they require heavy retraining every time new items or users are added. Conversely, inductive methods promise to solve these issues. KGs enhance inductive recommendation by offering information on item-entity relationships, whereas existing inductive methods rely purely on interactions, which makes recommendations for users with few interactions sub-optimal and even impossible for new items. In this work, we investigate the actual ability of inductive methods exploiting both the structure and the data represented by KGs. Hence, we propose GInRec, a state-of-the-art method that uses a graph neural network with relation-specific gates and a KG to provide better recommendations for new users and items than related inductive methods. As a result, we re-evaluate state-of-the-art methods, identify better evaluation protocols, highlight unwarranted conclusions from previous proposals, and showcase a novel, stronger architecture for this task. The source code is available at: https://github.com/theisjendal/kars2023-recommendation-framework.

1. Introduction

In Recommender Systems (RSs), an item is recommended to a user based on their preferences. Usually, these preferences are extracted from a user's historic interactions with items, such as clicks or purchases. A RS can either recommend based on user-item interactions, based on descriptive features of the items, or both. In the first case, for example in the movie domain, the system would assume that users that watch the same movies are likely to do so also in the future; this approach is commonly referred to as Collaborative Filtering (CF) [1, 2, 3, 4, 5]. In the second case, instead, the system would assume that the user is likely to watch movies of similar genres and plots of movies they watched in the past, i.e. a content-based method. Challenges arise for the former approach when, for a given user, only very few interactions are known; a similar challenge arises when information describing items is scarce. *The idea is then to combine both kinds of information.* In this regard, recently, RSs have been proposed to model knowledge about items derived from a KG [6, 7, 8, 9, 10, 11, 12]. A KG represents entities and their attributes as nodes and edges within a graph model, e.g., taxonomies, item descriptions, or categories attached to items. These models further integrate user-item interactions into the graph, obtaining in this way a Collaborative KG (CKG), as in Figure 1.A. Allowing for recommendations to users that have only a few ratings or with newly added items that have none at all.

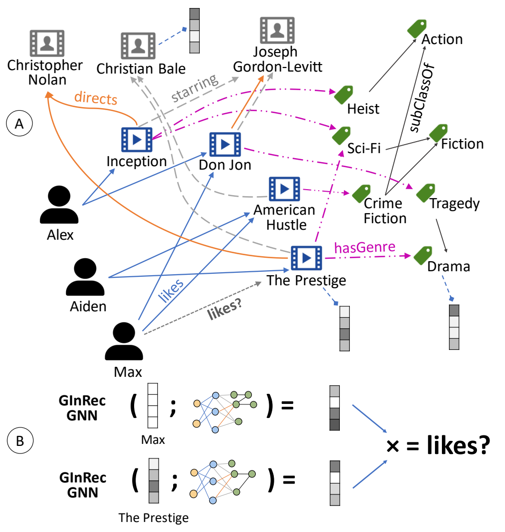

Figure 1: Item recommendation over a CKG: Part A shows a CKG in the movie domain, with users, movies, and connected entities; Part B illustrates the recommendation task.

For example, in Figure 1.B, we are making predictions for a user for whom we do not have any information (empty the embedding vector) except for a few rated items. The KG connects directors, genres, and actors to the rated movies, where some of these entities are described by textual information, e.g., bios and synopses. We can use the connections and data to infer user preferences beyond the collaborative signals.

Many existing methods only work in a *transductive* setting; that is, it is assumed that all users and items have

Fifth Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop @ RecSys 2023, September 18–22 2023, Singapore.

been seen during training [13, 14], meaning transductive models require retraining whenever new users or items are introduced. Instead, some models try to offer *inductive* capabilities [13, 15, 14]. *In an inductive setting, users and items exist that are not in the training set*; therefore, they extract information from the data to incorporate local structures to obtain an inductive bias. Nonetheless, *existing methods usually model only user-item interactions, ignoring the KG* [16, 13, 14, 17]. Our analysis of existing works (see Section 3) identified four important limitations: (i) the reliance on user metadata [17, 18], (ii) the tendency to rely exclusively on collaborative information and to bias preference over popular items [13, 15, 14], (iii) the poor scalability of methods that create user-item subgraphs for every rating [19, 15], and (iv) the missed opportunity to exploit item metadata and KG structure.

Hence, we first propose a new architecture for inductive recommendation using KGs. In our design, we strive for simplicity by adopting the efficiency and expressivity of Graph Neural Networks (GNNs) to aggregate structural information of each node's neighborhood, but going beyond trivial extensions of the GraphSAGE architecture as well as any other existing inductive method [16, 19, 20, 17, 15, 14, 21] due to its gated architecture that more effectively extrapolates inductive biases from the semantic and structural information encoded in the CKG. Furthermore, by reviewing the experimental evaluation of existing works, we have identified problematic methodologies on which we report here together with our results. Thus, we propose the Gated Inductive Recommendation (GInRec), *a new architecture that fully exploits the semantic information of real-world KGs* for inductive predictions in a scalable way.

2. Problem Formulation

Formally, we define a KG as a directed labeled multigraph identified by the triple $\langle \mathcal{N}, \mathcal{R}, \mathcal{L} \rangle$, where $\mathcal{N}$ is the set of entities (nodes) in the graph, $\mathcal{L}$, $\mathcal{L} \cap \mathcal{N} = \emptyset$, is the set of labels for the relations, and the edges between entities are represented as $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{L} \times \mathcal{N}$. Consider, for instance, the top-right portion of the example in Figure 1. Here, nodes represent movies, actors, directors, and a taxonomy of genres, while edges represent how nodes are connected, e.g., in the triple (Inception, hasGenre, Heist). As common in the literature [22], we split entities into two sets: the set of recommendable entities $\mathcal{I} \subset \mathcal{N}$, being the entities that the system can recommend to a user (e.g. movies); and the set of descriptive entities $\mathcal{E}_{desc} \subset \mathcal{N}$ (e.g., actors, genres, classes), such that $\mathcal{N} = \mathcal{I} \cup \mathcal{E}_{desc}$.

Furthermore, we adopt the concept of a CKG [6], i.e. a KG augmented with users' interactions with items, also shown in Figure 1. Formally, given the set of users $\mathcal{U}$, the interaction matrix $\mathbf{I} \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$ is a matrix of size $|\mathcal{U}| \times |\mathcal{I}|$, having $\mathbf{I}_{u,i} = 1$ if user $u \in \mathcal{U}$ has liked the item $i \in \mathcal{I}$ and $\mathbf{I}_{u,i} = 0$ if we do not have any information about the specific pair, e.g., if the user has never interacted with the item. Then, given the matrix $\mathbf{I}$ and the KG $\mathcal{G}$, the CKG $\mathcal{G}_\mathbf{c} : \langle \mathcal{N}_\mathbf{c}, \mathcal{R}_\mathbf{c}, \mathcal{L}_\mathbf{c} \rangle$, is an extension of $\mathcal{G}$, having $\mathcal{N}_\mathbf{c} = \mathcal{N} \cup \mathcal{U}$, $\mathcal{L}_\mathbf{c} = \mathcal{L} \cup \{\mathsf{Likes}\}$, and $\mathcal{R}_\mathbf{c} = \mathcal{R} \cup \{(u, \mathsf{Likes}, i) \mid \forall u \in \mathcal{U}, i \in \mathcal{I} \text{ s.t. } \mathbf{I}_{u,i} = 1\}$.

Finally, every node $n \in \mathcal{N}_\mathbf{c}$ is associated with a set of node features, assuming a function $\mathcal{X} : \mathcal{N}_\mathbf{c} \mapsto \mathbb{R}^d$ exists, called the *feature function*, assigning to each node a feature vector of dimension d. Typically, this vector provides a d-dimensional encoding of the node's contents, e.g., in this and other works [13] the word embedding of the textual description obtained by literal values attached to the nodes are used. However, since we do not want to use, or have, any user information, with the exception of their ratings, then we have $\forall u \in \mathcal{U} \,.\, \mathcal{X}(u) = \vec{\mathbf{0}}$. Therefore, given an interaction matrix $\mathbf{I}$, a KG $\mathcal{G}$ with feature function $\mathcal{X}$, a user u, and an item i such that $\mathbf{I}_{u,i} = 0$, we model the recommendation problem as the problem of predicting the likelihood of $\mathbf{I}_{u,i} = 1$ if we present the item i to the user u. In practice, we model our task as a *top-k recommendation problem.* Thus, we aim at learning a model Θ to parametrize a transformation function $\mathcal{F} : \mathcal{U} \times \mathcal{I} \mapsto [0, 1]$, such that $\mathcal{F}_\Theta(u, i) \geq \mathcal{F}_\Theta(u, j)$, imposes a partial order on $\mathcal{I}$ for every user in $\mathcal{U}$, if it is more likely that the user u would like i over j than vice versa.

Finally, in the recommendation setting, we define two types of users: those for which preferences across some items in $\mathcal{I}$ were known when learning Θ, i.e., at training time, and those for which no item rating was known during training, but for which some rating is known at inference time. We refer to the former as to the *warm-start* users $\mathcal{U}_w$ and to the latter as *cold-start* users $\mathcal{U}_c$, with $\mathcal{U} = \mathcal{U}_c \cup \mathcal{U}_w$. Transductive methods can only recommend for users in the warm-start set, while inductive methods can recommend for users in both sets. Typically, when a new user joins a platform, it is common practice to present them with an initial set of items to be rated. Thus, we consider cold-start users for which, at inference time, we have some ratings, even though those ratings are usually few and sparse [18, 23, 20].

3. Related Work

Most existing recommendation methods either only use bipartite graphs of user interactions with items [3, 5, 16] or are transductive [6, 7]. Instead, inductive learning models generate predictions for unseen nodes by directly reasoning over the features that describe them, but existing methods do not exploit KG data. Here, we provide an overview of inductive methods, detailing their limitations compared to our proposal (as summarized in Table 1) and describe the advantages of relational gates.

Inductiveness. GraphSAGE [13] was the first inductive GNN capable of efficiently generating embeddings for unseen nodes by leveraging pretrained node features for node classification. It was later expanded to a scalable item-item recommendation method, meaning no explicit modeling of user-item ratings [16].

Other methods have been proposed for inductive matrix completion by extracting subgraphs around each user-item pair to obtain the necessary representations [24, 25, 19, 15]. These approaches are designed for the single rating prediction task and not for the ranking task. *Generating these sub-graphs is prohibitively space- and time-consuming*. Thus, they cannot efficiently produce user-item rankings, since a subgraph is generated for *all* user-item pairs [20]. Furthermore, these methods do not use KG information; thus, they cannot provide predictions for new items with no interactions. Therefore, instead of constructing subgraphs, GInRec employs subsampling of neighboring nodes to obtain a scalable prediction mechanism [16, 13] and uses KGs to gain information about items with few user interactions. Several methods exploit user metadata, e.g., gender and age information [17, 18]. Yet, this information is rarely available, making it impossible to use these methods in practice. Hence, in our method, we assume no user metadata, learning instead *how* to aggregate information. Some methods are made for sequential recommendations [26], or cannot recommend for new users or for new items [14, 27], and in general cannot capture high-order connectivities between users, making them less relevant for our study.

Additionally, some methods are *quasi-inductive* since they consist of two parts: (1) a transductive part to obtain some initial embeddings and (2) an inductive part where the method learns to generate embeddings for new users or items [21, 20]. GInRec is *fully inductive* since it uses the extracted node textual features instead and thus does not need to learn the initial embeddings. Finally, many methods target the prediction of a user rating, which underperforms in the ranking task, even compared to non-personalized methods [2]. Thus, existing works (Table 1) either: (i) create subgraphs, which do not scale in the ranking task; (ii) use personal user data, which is almost never available; or (iii) solve a rating-prediction task that offers sub-optimal performances in practice. Therefore, we select GraphSAGE [13] and PinSAGE [16] as the only inductive recommenders fitting our recommendation setting and select IDCF [20] as a representative baseline for quasi-inductive methods.

Gates. Gates were originally used in Recurrent Neural Networks (RNNs) to learn long-term dependencies in time series [29, 30]. A gate limits the amount of information passed by learning a scalar in $[0, 1]$, for each dimension in a vector. On the contrary, the attention mechanism, which is often used in GNN aggregators, learns a single scalar for the entire vector [6, 8, 16]. Hence, the gates allow for differentiation at the dimension level

Table 1
Related methods, whether they use User Metadata, whether they handle Relational information (i.e., KG), the Task they support among (C) Node Classification, (R) Ranking, and (P) Rating Prediction, and whether the method constructs a Subgraph from user-item pairs.

	Inductive		User			
Model	**User**	**Item**	**Metadata**	**Relational**	**Task**	**Subgraph**
NGCF [5]	✗	✗	✗	✗	R	✗
KGAT [6]	✗	✗	✗	✓	R	✗
KPRN [7]	✗	✗	✗	✓	R	✗
KGCN-LS [8]	✗	✗	✗	✓	R	✗
MeLU [18]	✗	✗	✓	✗	R	✗
RuleRec [28]	✗	✗	✗	✓	R	✗
MGAT [9]	✗	✗	✗	✓	R	✗
LGCN [3]	✗	✗	✗	✗	R	✗
GraphSAGE [13]	(✓)	✓	✗	✗	C	✗
PinSAGE [16]	(✓)	✓	✗	✗	R	✗
BERT4Rec [26]	✓	✗	✗	✗	R	✗
IGMC [19]	✓	✓	✗	✗	P	✓
IDCF [20]	✓	✗	✗	✗	P	✗
PGD [17]	✓	✓	✓	✗	R	✗
ICP [14]	✗	✓	✗	✗	R	✗
GIMC [15]	✓	✓	✗	✗	P	✓
ReBKC [27]	✓	✗	✗	✓	P	✗
GInRec	✓	✓	✗	✓	R	✗

rather than the vector level.

For KGs, gates have been used to capture long-term path relations [7] and for aggregating neighbors for multi-model graphs [9]. Multi-modal information and relations in KGs differ in both semantic meaning and practical application, requiring different aggregation techniques. Therefore, GInRec adopts new relational-specific gates as an addition to the neighborhood aggregation.

Thus, *GInRec proposes a scalable inductive method for user-personalized recommendation* that learns to extract knowledge from a KG using relational gates without requiring any user metadata.

4. Methodology

We now present our model *Gated Inductive Recommendation (GInRec)*. The model consists of three components (as shown in Figure 2): (i) embedding layer, where we compress node feature information to create node embeddings (Figure 2 A); (ii) gated propagation layer, which chooses which information to propagate from the embeddings of neighboring nodes in a CKG to produce a high-order representation of each node and its neighbors (Figure 2 C and E); and (iii) prediction layer, creating a user and an item embedding given all propagation layers, outputting a ranking score (Figure 2 D). Hence, our architecture learns to recommend for users based on their interactions alone, introducing a gating mechanism to adaptively select information during aggregation along with an autoencoder regularization measure.

4.1. Embedding Layer

GInRec is designed as an inductive relational graph neural network. Therefore, given a target node $n \in \mathcal{N}_{\mathbf{c}}$, and

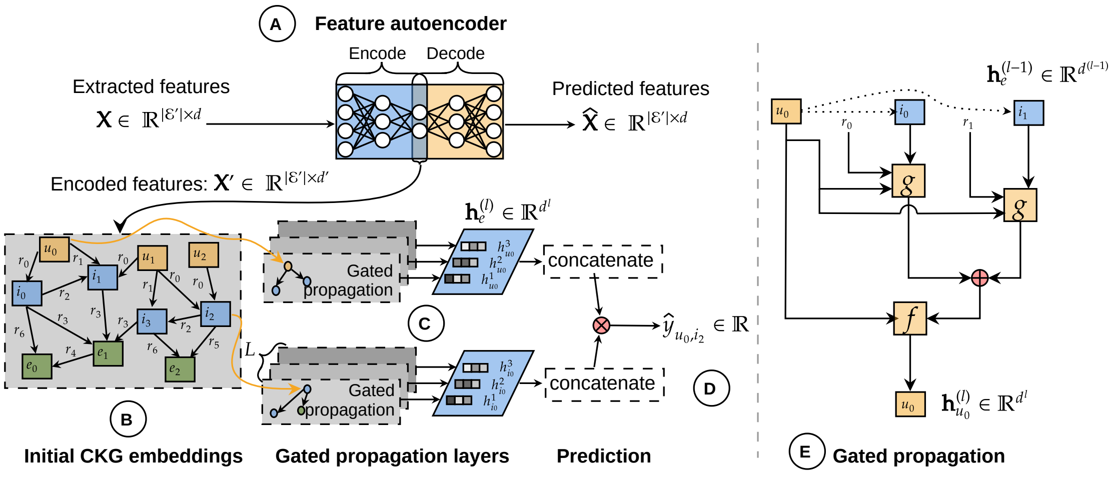
A Feature autoencoder
Encode
Decode
Extracted features
$\mathbf{X} \in \mathbb{R}^{|\mathcal{E}'| \times d}$
Predicted features
$\hat{\mathbf{X}} \in \mathbb{R}^{|\mathcal{E}'| \times d}$
Encoded features: $\mathbf{X}' \in \mathbb{R}^{|\mathcal{E}'| \times d'}$
$\mathbf{h}_e^{(l)} \in \mathbb{R}^{d^l}$
Gated propagation
concatenate
$\hat{y}_{u_0,i_2} \in \mathbb{R}$
L
B
C
D
Initial CKG embeddings
Gated propagation layers
Prediction
$\mathbf{h}_e^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$
g
f
$\mathbf{h}_{u_0}^{(l)} \in \mathbb{R}^{d^l}$
E Gated propagation

Figure 2: Illustration of GInRec, where the left side shows the model framework and the left is the gated propagation layer.

given a fixed number of steps L, we define $\mathcal{N}'$ as the set of all nodes reachable with an undirected path of length at most L from n, including n itself. Then, our initial input is the output of the feature function $\mathcal{X}$ for each node in $\mathcal{N}'$. The embeddings $\mathcal{X}$ is a combination of the textual descriptions of entities with their structural data (e.g., node degree). Specifically, we assume that each node in the KG $\mathcal{G}$ has a small set of descriptive sentences, e.g., movie plot or biography of the actor. Similar to seminal works [31], we use Sentence-BERT [32] to process the textual description of each entity and produce sentence embeddings such that sentences of similar semantic meaning are close to each other in a vector space. For multi-sentence descriptions, the average sentence embedding is used. When textual descriptions for descriptive entities are not available, we use ComplEX [33] to train entity embeddings for the descriptive entity in the KG, since these are static or very slowly changing. The initial vector is a concatenation of the textual embedding and the structural data, e.g., node degree. We standardize the features by removing the mean and scaling to unit variance as in other works [13]. Our approach can be extended to include additional features, such as item pictures for multi-modal descriptions, but we leave their study as future work. Thus, defining the initial matrix as $\mathbf{X} \in \mathbb{R}^{|\mathcal{N}'| \times d}$, where the i'th entity has the embedding of the i'th row and *users are initialized as zero vectors* in the embedding. As such, we only require a few interactions to represent users and textual descriptions for items.

The size of the initial feature vectors are usually large (in our model, we have $d > 756$), making subsequent computations infeasible. Therefore, we introduce an AutoEncoder (AE) layer to reduce the dimensionality [34] (shown in Figure 2 A). The loss of the AE is defined as:

$$\mathbf{L}_{AE} = \text{MSE}\Big(\mathbf{X}, \text{AE}_{\text{de}}\big(\text{AE}_{\text{en}}(\mathbf{X})\big)\Big) \tag{1}$$

where $\text{AE}_{\text{en}}:\mathcal{R}^{|\mathcal{N}'| \times d} \mapsto \mathcal{R}^{|\mathcal{N}'| \times d'}$, with $d' \ll d$, is the *encoding* function mapping the initial feature vector for each node to a set of lower dimensionality vectors through multiple fully connected layers with the Leaky ReLU activation [35]. Analogously, $\text{AE}_{\text{de}}:\mathcal{R}^{|\mathcal{N}'| \times d'} \mapsto \mathcal{R}^{|\mathcal{N}'| \times d}$ is a *decoding* function mapping the lower dimension embeddings back to the original vectors. Therefore, we produce a matrix $\mathbf{X}' = \text{AE}_{\text{en}}(\mathbf{X})$ of low dimensionality embedding for the initial nodes in $\mathcal{N}'$. Moreover, in our architecture, AE is jointly learned with the final ranking loss (as described later in subsection 4.4). Thus, $\mathbf{X}'$ provides a fine-tuned compressed representation of the extracted features. Since the initial embedding has no range limits, no activation is used for the final decode layer.

4.2. Gated Propagation Layer

The core of GNNs is the ability to aggregate information from its neighborhood. Relation types could allow the model to differentiate between the relation interactions and aggregate information dependent upon the combinations of edges in the CKG. Thus, we explore the effect of gates in our GNN's architecture and extend these with relation-specific weights [9, 36]. In the following, we first describe the individual parts for a single step, i.e., relation-specific gating, information propagation, and aggregation, and then how to generalize the process to high-order propagations.

Relation-specific gates: We design two relation-specific gates that control the information flow during message passing: (i) Inner Product and (ii) Concatenation. The Inner Product gate uses the inner product of the h and t entities as the gate, making the gate dependent on the affinity between the two. We take into account different relations (similar to TransR [37]) by first transforming the entities' embeddings into a relation-specific vector space before finding the affinities: $g_i(h, r, t) =$

$\sigma\big((\mathbf{W}_r\mathbf{e}_h)^\top\mathbf{W}_r\mathbf{e}_t\big)$. where σ is the sigmoid activation function, $\mathbf{W}_r$ is a relation-specific transformation matrix, and $(h,r,t)\in\mathcal{G}$. The Concatenation gate works as the original reset and update gate mechanisms used by GRU [30]. Here, we utilize a relation-specific linear transformation, which learns which parts of the tail entity's embedding are important in the aggregation step given both the head and the tail with $\|$ being concatenation as: $g_c(h,r,t)=\sigma\big(\mathbf{W}_r(\mathbf{e}_h\|\mathbf{e}_t)\big)$.

Information Propagation: Given the direct neighbors of entity h as $\mathcal{N}_h=\{(h,r,t)|(h,r,t)\in\mathcal{G}\}$, also called its ego-network [38], we can define the neighborhood aggregation vector of h as:

$$\mathbf{e}_{\mathcal{N}_h}=\frac{1}{|\mathcal{N}_h|}\sum_{(h,r,t)\in\mathcal{N}_h} g(h,r,t)\mathbf{e}_t. \tag{2}$$

In contrast to other gated networks [36, 9], our model's gates are relation-specific, allowing it to propagate different information from different parts of an entity's embedding based on the relation to it. This fine-grained information propagation is vital for GInRec's performance, especially when not relying on the initial user features.

Aggregation: The final part combines an entity's current embedding $\mathbf{e}_h$ with the aggregated ego embedding $\mathbf{e}_{\mathcal{N}_h}$, formally defined as $\mathbf{e}'_h=f(\mathbf{e}_h,\mathbf{e}_{\mathcal{N}_h})$, where f is an aggregator function. We identify four common aggregators used in other architectures, namely: *Bi-interaction aggregator* [6], *GCN aggregator* [39], *GraphSAGE aggregator* [13], and *LightGCN aggregator* [3], finding the LightGCN aggregator to be the best performing through hyperparameter tuning. The LightGCN aggregator can be defined as $f_{LGCN}=\mathbf{e}_{\mathcal{N}_h}$, not having any transformations or non-linear activations.

High-order propagation: To propagate information from n-hop neighbors and utilize high-order connectivity information, we stack the model in layers [6, 9, 13]. As illustrated by the arrow from A to B in Figure 2, we use the output of the embedding layer $\mathbf{X'}\in\mathbb{R}^{|\mathcal{N}|\times d'}$ as the initial embedding in the propagation layers. We thus define the next representation layer $l+1$, recursively using the previous layer l and the neighborhood representation as: $\mathbf{e}_h^{(l)}=f(\mathbf{e}_h^{(l-1)},\mathbf{e}_{\mathcal{N}_h}^{(l)})$.

The weight matrices in the aggregator functions are in the space $\mathbb{R}^{d^{(l+1)}\times d^{(l)}}$, defining the embedding dimension for the next layer. We define the information propagated from the ego-network as:

$$\mathbf{e}_{\mathcal{N}_h}^{(l)}=\frac{1}{|\mathcal{N}_h|}\sum_{(h,r,t)\in\mathcal{N}_h} g(h,r,t)^{(l-1)}\mathbf{e}_t^{(l-1)} \tag{3}$$

where the elements of the gates are the entities $\mathbf{e}_h^{(l-1)}$, $\mathbf{e}_t^{(l-1)}$, and the relation matrix is either $\mathbf{W}_r^{(l)}\in\mathbb{R}^{n\times d^{(l-1)}}$ or $\mathbf{W}_r^{(l)}\in\mathbb{R}^{d^{(l-1)}\times 2d^{(l-1)}}$ depending on whether they refer to the g_i or g_c gate, respectively.

4.3. Prediction

At each layer, information from increasingly distant entities is aggregated, and we, therefore, have multiple representations of the entities after L layers of propagation. Figure 2 C shows the information from multiple layers as it is passed to the prediction step. Similar to previous apporaches [39, 6, 9], we concatenate the output after each layer for a user u and item i as:

$$\mathbf{e}_u^*=\mathbf{e}_u^1\|\dots\|\mathbf{e}_u^L,\qquad \mathbf{e}_i^*=\mathbf{e}_i^1\|\dots\|\mathbf{e}_i^L \tag{4}$$

This approach is able to retain information of the different representations at all steps. For the final prediction, learned, non-linear similarity measures are usually outperformed by a simple inner product [40] that also reduces complexity. Hence, our prediction is computed as follows: $\hat{y}_{ui}=\mathbf{e}_u^{*\top}\mathbf{e}_i^*$.

4.4. Optimization

We use Bayesian Personalized Ranking (BPR) as the collaborative loss function, assuming previously interacted items should be ranked higher than others [9, 6] as:

$$\mathbf{L}_{CF}=\sum_{(u,i,j)\in\mathcal{B}}-\ln\sigma(\hat{y}_{ui}-\hat{y}_{uj}) \tag{5}$$

where $\mathcal{B}=\{(u,i,j)\mid \mathbf{I}_{ui}=1 \text{ and } \mathbf{I}_{uj}\neq 1\}$ is a set of training triples with item i being rated higher than j and σ is the sigmoid function. The final loss function is a combination of autoencoder loss in Equation 1 and the BPR loss in Equation 5, so to learn an encoded embedding suitable for recommendation while containing enough information to reconstruct, computed as: $\mathbf{L}=\mathbf{L}_{CF}+\lambda\mathbf{L}_{AE}+\gamma\|\Theta\|_2^2$ with $\Theta=\{\mathbf{W}_r^{(l)},\mathbf{W}^{(l)}|\forall l\in\{1,...,L\}\}\cup\{\mathbf{W}_e^{(l')},\mathbf{W}_d^{(l')}|\forall l'\in\{1,...,L_{ae}\}\}$ is the set of learnable parameters, γ is a parameter for tuning the L_2 regularization, and λ is a parameter to tune the autoencoder loss. The autoencoder loss also works as a regularizer while also being recommender-specific. Generating embeddings for all nodes, with MovieLens Subsampled (ML-S) shown in Table 2 took 0.48s, and ranking items for all users took 0.078s, compared to PinSAGE's 0.446s and 0.966s, with a RTX 2070 Super and Intel i9-9900, averaged over 5 runs.

Training: We use mini-batch training sampled from $\mathcal{B}$ of size 1024, limiting the computation graph by having a fixed size ego-network of 10 and starting construction from the last layer [13]. The entities used in the first layer of the gated propagation are used for the autoencoder loss, such that we learn to represent not only users and items but also entities like genres and actors.

Scalability. In our embedding approach, both the calculation of the aggregation ($\mathbf{e}_h^{l+1}$) and of the prediction ($\hat{y}$) are all bounded by the number of nodes in the graph, while the calculation of the ego-network ($\mathbf{e}_{\mathcal{N}_h}^{(l+1)}$) is bounded by the number of edges. As these steps are applied sequentially and $|\mathcal{V}|\ll|\mathcal{E}|$, we know that the complexity of our method is bounded by the ego-network aggregation

Table 2
Dataset (top) and KG (bottom) statistics.

	ML-20m	ML-S	AB	AB-S
# Users	132,287	12,500	70,679	60,000
# Items	4725	4438	24,841	24,841
# Ratings	11,376,533	1,106,000	847,733	720,111
Density	0.018	0.019	0.00048	0.00048

	Entities	Edges	Labels	Density
MindReader [22]	13,767	201,438	8	1.06e-3
Amazon Book [6]	88,572	2,555,995	39	3.26e-4

complexity, more specifically, the linear transformation of the gate calculation. When naïvely applying the gates over all edges, the complexity is $O(|\mathscr{E}|d)$, where d is the largest dimension utilized during graph convolutions – we note that $|\mathscr{E}|$ is bounded by $O(|\mathscr{V}|^2|\mathscr{R}|)$. Yet, as $\mathbf{W}_r(\mathbf{e}_h \| \mathbf{e}_t)$ is equivalent to $\mathbf{W}_r^1\mathbf{e}_h + \mathbf{W}_r^2\mathbf{e}_t$ we only need to compute the transformation for each unique (h, r) and (r, t) pair instead of each unique (h, r, t) triple. Therefore, we can apply a MapReduce computation [16] to have at most $2|\mathscr{V}||\mathscr{R}|$ calculations, leading to the complexity $O(|\mathscr{V}||\mathscr{R}|d) \ll O(|\mathscr{E}|d)$. Finally, our prediction is a dot product after graph convolutions; hence, our method can predict in $O(|\mathbf{e}_u^*| \cdot |\mathscr{I}|)$ for a single user as the vector dot product complexity is $O(|\mathbf{e}_u^*|)$, which we do $|\mathscr{I}|$ number of times, which is less than existing architectures with comparable approaches, e.g., PinSAGE.

5. Experiments

Inductive approaches are designed to provide recommendations in a cold-start setting, where ratings for new users are only known at inference time. Yet, as we will show, these baselines do not perform in this setting due to poor selection of learning metrics, evaluation methodologies, or other complexities. In the following, we aim at answering the questions: **RQ1)** Which design decisions affect the prediction performance compared to state of the art? **RQ2)** What is the effect of the negative sampling strategy in the evaluation? **RQ3)** How does relational gates affect performance?, and finally **RQ4)** How do the structure and data of the KG affect performance?

Datasets. We adopt two real-world datasets: (i) MovieLens-20m (ML-20m) [41], a dataset with ratings on movies and (ii) Amazon-Book (2014) (AB) [42], a dataset with reviews on books. Neither dataset has an associated KG. We therefore use the MindReader KG [22] for the ML-20m dataset, and for the AB dataset the KG constructed to evaluate KGAT [6]. These two graphs link reviewed items to nodes in popular open-domain KGs such as DBpedia and WikiData. In both cases, we keep only items mapped to the KG leading to the statistics shown in Table 2. We adopt splitting ratios 0.8 : 0.1 : 0.1 for train, validation, and test sets, respectively. We note that different versions of the AB dataset exist, and results cannot necessarily be directly compared between related works and our dataset [20, 6, 3, 5, 43]. In our cold-start experiments, as defined in Section 4, we sampled 12,500 users from ML-20m and 60,000 users from AB for training, named ML-S and Amazon-Book Subsampled (AB-S), respectively. We sample more users for AB-S due to few ratings per user. We then created two cold-start scenarios on ML-S: one adding 10% new users (i.e., 1250); and one where we treat all users not in ML-S as cold-start users, being ~90% of the users in the original ML-20m dataset, allowing us to test the scalability of the inductive methods. For AB-S, we create one scenario adding the remaining users from the original dataset, corresponding to an additional ~15% of the total number of users.

Methods. We compare to five methods: TopPop [2], a non-personalized common baseline [10] that recommends the most popular items; GraphSAGE [13], modified to recommend using cosine similarity between a user's rated items and new items; PinSAGE[16], with a semi-supervised objective, i.e., items co-rated should be similar, analogous to the pin / board setting; IDCF [20], a two-step learning method, using key user embeddings to initialize new users; and BPR-MF [4], which we report as a reference transductive method retrained on the dataset including also the cold-start users, since it is fast to train and is competitive to state-of-the-art methods *without* requiring sequential data and shown to outperform the standard kNN method.

All models are implemented in PyTorch and optimized using the Adam optimizer . We save the best-performing state based on the validation set and stop after 50 successive epochs without improvement. For hyperparameter tuning of all models, we employ Asynchronous Successive Halving (ASHA) [44], all hyperparameter options and ASHA parameters available in our source code. We know that compared to PinSAGE, GInRec has only one extra hyperparameter, in the form of λ, for which we tune, as in subsection 4.4.

Evaluation metrics. Following other evaluation methods [6], for each user in the test set, we rank all items not interacted with in the train and validation sets, only treating ratings in the test set as positive items. We measure NDCG, recall, precision, and coverage at 20 for each user, reporting the average performance over all users. Let $\mathscr{I}_u$ to be the top-k items recommended to a user u, then we can define coverage as: $Cov@k = |\bigcup_{u \in \mathscr{U}} \mathscr{I}_u| / |\mathscr{I}|$. Hence, a naive recommender as TopPop is expected to perform poorly due to recommending the same set of items each time [45]. We further include I-NDCG [20] which is the metric used to evaluate IDCF in the original work, where X negative items (X=5 in IDCF) are sampled per positive item in the test set instead of all possible negative items as for NDCG. For all metrics, we remove items seen by the user during training from the set of negative samples. We use '*' to represent a statistically significant increase in performance using student t-test.

Table 3
Results on the different dataset. '*' represents statistical significant increase over the best baseline. NDCG, Recall and Precision is performed at 20, while I-NDCG is the calculation method used in [20] for the full subsampled list.

	MovieLens Subsampled + 1250 users				MovieLens Subsampled + 90%				Amazon Book Subsampled + 15%			
	NDCG	Recall	Precision	I-NDCG	NDCG	Recall	Precision	I-NDCG	NDCG	Recall	Precision	I-NDCG
TopPop	0.11182	0.14351	0.05256	0.85136	0.10916	0.14426	0.05147	0.85350	0.01552	0.03496	0.00273	0.78715
BPR-MF	0.13221	0.16920	0.06504	0.88250	0.10743	0.13961	0.05120	0.85350	0.01724	0.03630	0.00304	0.78715
GraphSAGE	0.07862	0.10637	0.04336	0.84262	0.07281	0.10356	0.04164	0.84345	0.00667	0.01695	0.00149	0.76845
PinSAGE	**0.14129**	**0.18745**	**0.06396**	**0.88378**	**0.13592**	**0.18622**	**0.06280**	**0.88496**	**0.05398**	**0.11764**	**0.00927**	**0.89937**
IDCF	0.11042	0.14155	0.05244	0.81959	0.10875	0.14394	0.05113	0.85388	0.03033	0.06674	0.00531	0.85204
GInRec	**0.18085***	**0.24618***	**0.08460***	**0.92200***	**0.18411***	**0.24482***	**0.08365***	**0.92208***	**0.06472***	**0.14055***	**0.01144***	**0.91769***

RQ1 As Table 3 shows, GInRec is able to outperform all methods on all metrics with statistical significance. We also see contrasting results w.r.t. the original IDCF evaluation. IDCF was originally evaluated in a ranking setting; however, the learned embeddings of IDCF are learned towards Cross-Entropy, a non-ranking, point-wise learning objective. Such learning methodologies have been shown to perform poorly, with a similar or worse ranking than TopPop [2, 46]. Yet, IDCF uses the Cross-Entropy loss [47]. In the original work, IDCF outperforms PinSAGE by a small margin, *yet we observe the opposite in our evaluation.* PinSAGE's increased performance in our evaluation is due to (i) a more appropriate early stopping based on the evaluation metric instead of the loss function and (ii) our evaluation adopting a better learning objective for PinSAGE.

Figure 3 also shows GInRec outperforms all models in different user splits, we leave out the result on AB-S for brevity, noting we get similar results. We demonstrate that using KG information and relational gates provides superior predictive power in all cases; given the improved performance over all popularity and sparsity groups.

To study the method's ability to make personalized recommendations, we utilize coverage [45]. We note that we are not able to perform statistical significance testing with coverage as we only generate a single score per dataset instead of one score per user. The metric is not useful by itself; a random model would have close to 1 in coverage. Having higher coverage but a far lower ranking indicates more random recommendations while having low coverage but a high ranking score means low personalization and a popularity-biased dataset. In Table 4, we can see a clear improvement over all other methods. Only GraphSAGE gets higher performance, yet, it is unable to make high-quality recommendations. Its performance is therefore more random. IDCF performs poorly on all datasets w.r.t. coverage; this is correlated as to why it performs better on NDCG-I compared to NDCG as we will discuss later. We also test with the Gini Coefficient on the ML-S+1250 dataset, where 0 would be an equal (uniform) distribution, and 1 is unequal. Here GInRec get 0.959, BPR-MF 0.989, PinSAGE 0.991, TopPop 0.993, and random having 0.241. Thus using this metric GInRec gets $\geq$3% more diverse distribution over PinSAGE and BPR-MF. While BPR-MF performs better than TopPop on both ML-S dataset with 1250 new users and the AB-S dataset, its performance decreases when adding a large number of users to the ML-S dataset. We find our method to have similar increase in performance over all k's in set $\{1, 5, 10, 20, 50\}$, but these results are not included here due to space constraints.

RQ2. When evaluating ranking performance, NDCG is the metric commonly adopted, but there are two alternatives on which set of items to rank: either rank all items in the dataset or just a subset. In other evaluations [20, 1, 2], instead of ranking all items, only X negative items are randomly sampled per each positive. While this would aim at making it equally hard to rank positive items for all test users, it has been proven to produce unreliable comparisons of performances across methods [47]. This has also been witnessed when other works re-evaluated BERT4Rec [26], which also utilized negative subsampling, though using popularity-biased sampling instead of uniform sampling. Also in this version, negative sampling leads to unreliable results [48] finding even simple baselines outperforming this state-of-the-art method. *Yet, in the original IDCF evaluation, this faulty method (here labeled I-NDCG) is adopted.* Thus, hard-to-rank items are often missing from the evaluation when subsampling negative items, and thus I-NDCG does not test the actual performance of the method as if all possible negative items are available. This presents an issue when considering the experimental evaluation of previous works. Therefore, here we once more compare the two evaluation techniques: (a) ranking all items, (b) subsampling negative items, and verify once more that *the latter methodology should be avoided since it produces biased results.* In Table 3, we see GraphSAGE outperforms IDCF on I-NDCG in ML-S+1250, though clearly performing worse in the appropriate NDCG@20. Hence, this negative subsampling (I-NDCG) unfairly favors Top-

Table 4
Coverage at 20 for all datasets.

	ML-S + 1250	ML-S + 90%	AB-S + 10%
TopPop	0.01587	0.02222	0.00125
BPR-MF	**0.07534**	0.03217	0.00624
GraphSAGE	0.06307	**0.31471**	**0.14243**
PinSAGE	0.02857	0.04169	0.12439
IDCF	0.01566	0.02201	0.04348
GInRec	**0.18540**	**0.42646**	**0.21460**

Figure 3: Comparison on ML-S + 90% (top) on NDCG@20. Figure **a** illustrates binning users based on number of ratings, **b** are based on percentage of popular ratings, and **c** binning user based on number of ratings with each bin having equal total amount of ratings as in [6]. The grey bars are the number of users in each bin. The legend applies to all plots.

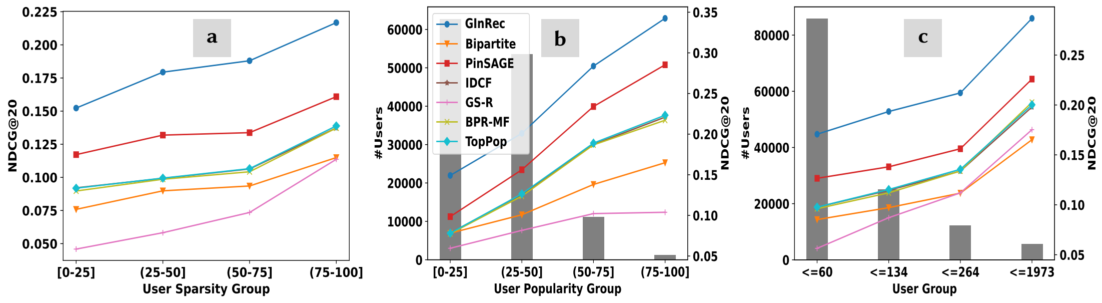

Table 5
Effect of the gating mechanism and KG measured at 20.

	MovieLens Subsampled + 1250 users				MovieLens Subsampled + 90%				Amazon Book Subsampled + 15%			
	NDCG	Recall	Precision	Cov	NDCG	Recall	Precision	Cov	NDCG	Recall	Precision	Cov
Bipartite	**0.17990**	**0.24356**	**0.08392**	**0.18307**	0.09353	0.19029	0.06455	**0.37757**	0.04193	0.09271	0.00770	**0.29254**
Inner Product	0.12798	0.17291	0.05968	0.04254	0.11336	0.15324	0.05143	0.04593	0.03453	0.07328	0.00605	0.06465
W/o relations	0.17720	0.24193	0.08288	0.18243	**0.17799**	**0.23688**	**0.08140**	0.34519	**0.06451**	**0.13395**	**0.01106**	0.18465
W/o gates	0.12482	0.17000	0.05848	0.08148	0.09416	0.13479	0.04331	0.05122	0.03653	0.00634	0.00927	0.07673
GInRec	**0.18085**	**0.24618**	**0.08460**	**0.18540**	**0.18411***	**0.24482***	**0.08365***	**0.42646**	**0.06472**	**0.14055**	**0.01144**	0.21460

Pop even above IDCF and IDCF over other methods. Instead, when appropriately considering all items (NDCG) as recommended [47], then GInRec, and other methods, perform up to 3x times better than TopPop.

RQ3 & RQ4. The method's results with different gating mechanisms can be seen in Table 5. In the table, 'w/o relation' is the gating mechanism without relation type, i.e., effectively ignoring edge types, and 'w/o gates' is the method without the gating mechanism. Overall, the *gating mechanism improves performance*, as it adaptively selects information from neighboring nodes, and it outperforms the two other models in all metrics. Disregarding relation types leads to worse performance on all datasets, and completely removing the gates leads to dramatically lower performance. Thus, *it is vital to design models that can exploit the semantic information modeled by KGs.* The Inner Product gate scales its neighbors' embeddings instead of selecting different parts as the Concatenate gate and thus limits the flow from certain nodes. Yet, it is not able to select which part of the neighbor's embeddings to propagate, thereby achieving a similar performance to the 'w/o gates' method. Having the ability to limit flow for each dimension of the neighbor's embedding is, therefore crucial for our method. GInRec without gates is worse than PinSAGE, though still better than IDCF. Hence, only using the user's interactions without a gating mechanism is still better than the reconstruction used in IDCF. Even without relation types, we see a large and statistically significant increase in performance. When looking at Figure 3, we see in all cases that GInRec performs better than the bipartite version. In the first bin of all plots (i.e., the bins with fewer or less popular ratings), *we see a large performance increase when using the semantic information carried by KG, both over the bipartite model, but also related works.*

Summarised, our gated aggregators can exploit the relational information, as either removing the KG or the relational information lead to a decrease in performance. When adding many users, we even see a large decrease in performance for the bipartite method, illustrating the scalability of our method and gated aggregation.

6. Conclusion and future work

In this work, we devise a scalable gated GNN architecture to perform inductive recommendation, with the ability to utilize high-order connectivities in CKGs. We show that our method outperforms existing approaches. Further, we showcase methodological limitations in previous evaluations. We conclude that this kind of architecture deserves further study, especially given its ability to: (1) scale to large graphs and large numbers of users (easily extensible to distributed frameworks), and (2) maintain good prediction with new users and items despite its lightweight inference methodology.

Acknowledgments

Matteo Lissandrini is supported by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement no. 838216. Katja Hose and Theis Jendal are supported by the Poul Due Jensen Foundation and the Independent Research Fund Denmark (DFF) under grant agreement no. DFF-8048- 00051B.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: TheWebConf'17, 2017.

[2] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: RecSys'10, 2010.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: SIGIR'20, 2020.

[4] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: UAI'09, 2009.

[5] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: SIGIR'19, 2019.

[6] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: SIGKDD'19, 2019.

[7] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.-S. Chua, Explainable reasoning over knowledge graphs for recommendation, in: AAAI'19, 2019.

[8] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: SIGKDD'19, 2019.

[9] Z. Tao, Y. Wei, X. Wang, X. He, X. Huang, T.-S. Chua, Mgat: Multimodal graph attention network for recommendation, Information Processing & Management (2020).

[10] E. Palumbo, D. Monti, G. Rizzo, R. Troncy, E. Baralis, entity2rec: Property-specific knowledge graph embeddings for item recommendation, Expert Systems with Applications (2020).

[11] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Ripplenet: Propagating user preferences on the knowledge graph for recommender systems, in: CIKM'18, 2018.

[12] Z. Yang, S. Dong, Hagerec: hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation, Knowledge-Based Systems (2020).

[13] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NeurIPS'17, 2017.

[14] C. Zhang, H. Yao, L. Yu, C. Huang, D. Song, H. Chen, M. Jiang, N. V. Chawla, Inductive contextual relation learning for personalization, ACM Transactions on Information Systems (2021).

[15] C. Zhang, H. Chen, S. Zhang, G. Xu, J. Gao, Geometric inductive matrix completion: A hyperbolic approach with unified message passing, in: WSDM'22, 2022.

[16] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: SIGKDD'18, 2018.

[17] S. Wang, K. Zhang, L. Wu, H. Ma, R. Hong, M. Wang, Privileged graph distillation for cold start recommendation, in: SIGIR'21, 2021.

[18] H. Lee, J. Im, S. Jang, H. Cho, S. Chung, Melu: Meta-learned user preference estimator for cold-start recommendation, in: SIGKDD'19, 2019.

[19] M. Zhang, Y. Chen, Inductive matrix completion based on graph neural networks, in: ICLR'19, 2019.

[20] Q. Wu, H. Zhang, X. Gao, J. Yan, H. Zha, Towards open-world recommendation: An inductive model-based collaborative filtering approach, in: ICML'21, 2021.

[21] Y. Wu, Q. Cao, H. Shen, S. Tao, X. Cheng, INMO: A model-agnostic and scalable module for inductive collaborative filtering, in: SIGIR'22, 2022.

[22] A. H. Brams, A. L. Jakobsen, T. E. Jendal, M. Lissandrini, P. Dolog, K. Hose, Mindreader: Recommendation over knowledge graph entities with explicit user ratings, in: CIKM'20, 2020.

[23] K. Zhou, S.-H. Yang, H. Zha, Functional matrix factorizations for cold-start recommendation, in: SIGIR'11, 2011.

[24] M. Xu, R. Jin, Z.-H. Zhou, Speedup matrix completion with side information: Application to multi-label learning, in: NeurIPS'13, 2013.

[25] P. Jain, I. S. Dhillon, Provable inductive matrix completion, arXiv preprint arXiv:1306.0626 (2013).

[26] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: CIKM'19, 2019.

[27] B. Hui, L. Zhang, X. Zhou, X. Wen, Y. Nian, Personalized recommendation system based on knowledge embedding and historical behavior, Appl. Intell. (2022).

[28] W. Ma, M. Zhang, Y. Cao, W. Jin, C. Wang, Y. Liu, S. Ma, X. Ren, Jointly learning explainable rules for recommendation with knowledge graph, in: TheWebConf'19, 2019.

[29] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation (1997).

[30] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: EMNLP'14, 2014.

[31] S. Liu, I. Ounis, C. Macdonald, Z. Meng, A heterogeneous graph neural model for cold-start recommendation, in: SIGIR'20, 2020.

[32] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: EMNLP-IJCNLP'19, 2019.

[33] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: ICML'16, 2016.
[34] M. A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE journal (1991).
[35] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, Nature 405 (2000).
[36] Y. Li, D. Tarlow, M. Brockschmidt, R. S. Zemel, Gated graph sequence neural networks, in: ICLR'16, 2016.
[37] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: AAAI'15, 2015.
[38] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, J. Tang, Deepinf: Social influence prediction with deep learning, in: SIGKDD'18, 2018.
[39] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks (2017).
[40] S. Rendle, W. Krichene, L. Zhang, J. Anderson, Neural collaborative filtering vs. matrix factorization revisited, in: RecSys'20, 2020.
[41] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) (2015).
[42] J. Ni, J. Li, J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in: EMNLP-IJCNLP'19, 2019.
[43] J. Zhu, Q. Dai, L. Su, R. Ma, J. Liu, G. Cai, X. Xiao, R. Zhang, BARS: towards open benchmarking for recommender systems, in: SIGIR'22, 2022.
[44] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, A. Talwalkar, Massively parallel hyperparameter tuning (2018).
[45] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Trans. Knowl. Data Eng. (2012).
[46] J. Wu, X. Wang, X. Gao, J. Chen, H. Fu, T. Qiu, X. He, On the effectiveness of sampled softmax loss for item recommendation, arXiv preprint arXiv:2201.02327 (2022).
[47] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: SIGKDD'20, 2020.
[48] S. Latifi, D. Jannach, A. Ferraro, Sequential recommendation: A study on transformers, nearest neighbors and sampled metrics, Inf. Sci. 609 (2022) 660–678.