Knowledge and Information Systems (2022) 64:2457–2480
https://doi.org/10.1007/s10115-022-01709-1

REGULAR PAPER

Graph attention-based collaborative filtering for user-specific recommender system using knowledge graph and deep neural networks

Ehsan Elahi[1] · Zahid Halim[1]

Received: 7 January 2022 / Revised: 18 June 2022 / Accepted: 26 June 2022 /
Published online: 23 August 2022

Abstract
Collaborative filtering suffers from the issues of data sparsity and cold start. Due to which recommendation models that only rely on the user–item interaction graph are insufficient to model the latent relationship between complex interaction of users and items. Existing methods utilizing knowledge graphs for recommendation explicitly model the multi-hop neighbors of an entity while ignoring the relation-specific as well as user-specific information. Moreover, a collaborative signal is also crucial to be modeled explicitly besides knowledge graph information. In this work, a novel end-to-end recommendation scenario is presented which jointly learns the collaborative signal and knowledge graph context. The knowledge graph is utilized to provide supplementary information in the recommendation scenario. To have personalized recommendation for each user, user-specific attention mechanism is also utilized. The user and item triple sets are constructed which are then propagated in the knowledge graph to enrich their representation. Extensive experiments are carried out on three benchmark datasets to show the effectiveness of the proposed framework. Empirical results show that the proposed model performs better than the state-of-the-art KG-based recommendation models.

Keywords Recommender system · Knowledge graph · Deep neural networks · Data mining

1 Introduction

Due to the massive information overload on e-commerce websites and social media platforms, it becomes an essential requirement to display only personalized content to the users. For this purpose, many e-commerce websites such as Amazon, eBay, and Costco, to name a few and other platforms (like, YouTube, Spotify, and Netflix) are deploying their recommendation systems to expand the business and attract more users. This way, the information overload problem can be alleviated as users will only see what they are interested in, from

✉ Zahid Halim
ehsan.elahi@giki.edu.pk; zahid.halim@giki.edu.pk

1 Faculty of Computer Science and Engineering, The Machine Intelligence Research Group (MInG), Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan

the vast sea of products, such as movies, news, books, or restaurants, to name a few, thus, filtering out the irrelevant items from the user display. In the past few years, several recommendation techniques, for example, the collaborative filtering-based method [1] and Graph Neural Network (GNN) [2], have emerged. They consider the user–item interaction information for making personalized recommendations for the clients. Recently, deep learning models are being used for solving recommendation system problems due to their capability to effectively model the latent nonlinear relationship between a user and an item. GNN is widely used in the recommendation scenario as it exhibits outstanding capability for learning graph data [3]. Since the recommendation problem is mapped as a bipartite graph with users and items represented as nodes, while the observed interaction between them is represented as edges, the GNN is employed in the literature for recommendation problems. Moreover, GNN has the benefit of providing effective and systematic methods for examining multi-hop interactions, which has shown improved performance of the recommender system. Many studies have shown that GNN-based models outperform previous techniques and produce new state-of-the-art results on publicly available datasets [4–6].

Since the user–item interactions may reach up to millions or even larger in real-world scenarios, it is important to extract only the relevant information for each user/item. Therefore, the concept of high order interaction is introduced which is an obvious way to incorporate the collaborative signal and enrich the representation of both users and items. The information about directly connected neighbors can be gathered with the help of one layer, i.e., first-hop neighbors of a given node can be accessed by one layer. In the same way, the information about L-hop away neighbors can be gathered by stacking L-layers in the architecture. It is important to note here that multiple layers enrich the target node representation; however, they also cause the over-smoothing problem which causes losing the uniqueness of each node and every node contains information of every other node. In general, the two main components of Collaborative Filtering (CF) models are embedding and interaction modeling. To elaborate, embedding is the vectorized representation of users and items (which are essentially the nodes in a bipartite graph). In interaction modeling, the embedded form of users and items is used to rebuild the user–item historical interaction. Matrix Factorization (MF) [7] is one such example that maps the user and item IDs into vectorized form (the embedding component). Once the vectorized form is obtained, historical interaction is rebuilt by taking an inner product of user/item embeddings (the interaction modeling component). Some other examples include neural collaborative filtering models which, instead of using the inner product for modeling user–item historical interaction, make use of nonlinear neural networks [8]. Some others, e.g., translation-based CF models [9] make use of Euclidean distance metric for modeling user–item historical interaction. Although these methods are quite effective, they do not consider the modeling of collaborative effect in the embedding process, in an explicit manner. Thus, it makes the representation of user/item less expressive in depicting the behavior similarity between users/items. Specifically, some methods make use of descriptive features only such as ID in the embedding function, while ignoring the collaborative signal which is latent in the user's historical interaction. Thus, the embedding generated in this way is insufficient; therefore, the only way for modeling user–item interaction is through the usage of interaction function.

Although a lot of research has been conducted on recommendation systems, still the problem of data sparsity and cold start problems are the key challenges that hinder the performance of a recommender system. Data sparsity is the issue where there is lack of sufficient users, as the active users have given ratings to only a few items. Cold start means that it is difficult to generate recommendations for cold (not active) users, since they have little item-interaction information. To alleviate this problem, some sort of auxiliary information

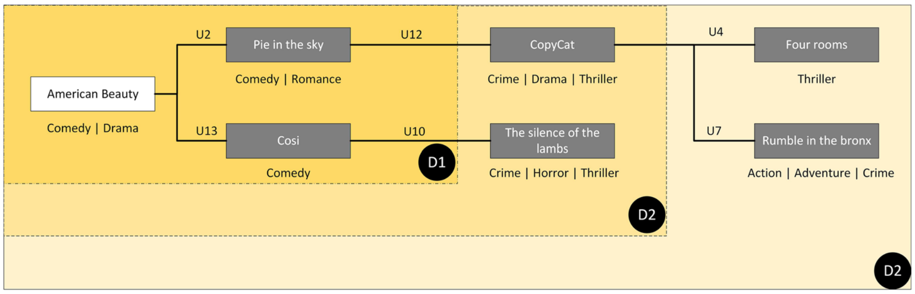


Fig. 1 Illustration of noise being introduced in the MovieLens dataset by aggregating information from high order neighbors

is needed to enrich the user and item representation. Knowledge Graph (KG) is one such example of providing auxiliary information. KG has different types of rich relations between items and hence, proven to be effective for making precise recommendations [10]. KG is a heterogeneous graph having entities and relations. The entities correspond to the nodes in the graph, whereas the relation between entities corresponds to the edge between them. Various KGs have been used effectively in different circumstances such as in KG completion tasks (Microsoft Satori, DBpedia) [11], and in question answering undertakings [12]. From the success of KG in the aforementioned applications, researchers have also utilized them in the recommendation scenario for improving the performance of the recommendation system as in [13] and [14].

Propagation-based methods have gained much attention due to their success in KG-based recommendation models. However, there is a problem with them as long-term connectivity along links of KG may introduce the noise and can ignore the semantic relation between the entities. The user interacts with a wide range of items, the original representation of the item may be different with the high order propagation, and thus, more irrelevant entities may be injected into the original item representation [15]. For example, as shown in Fig. 1, the movie *American Beauty* has a genre as *Comedy* | *Drama* but after propagating to three order interaction, it reaches to completely irrelevant movie *Four rooms* having the label *Thriller*. This is due to the higher-order propagation along with the links of KG which may introduce noise into the representation of items and entities.

In this work, we address the aforementioned limitations of KG-based methods in the literature. Specifically, we develop a KG-based model called Graph Attention-based Collaborative Filtering (GACF) for user-specific recommender system using knowledge graph which explicitly integrates the user–item interaction information and KG propagation is employed by explicit consideration of dependency between the target entity (an entity whose representation is to be learnt to enrich its information) and its multi-hop neighbors. The graphical representation of user–item interaction graph along with KG is shown in Fig. 2 where a toy example is illustrated for the understanding purpose. In GACF, the information is viewed differently in user–item interaction as well as in KG, as their weights are different in diverse views. Moreover, the proposed model is user-specific which produces personalized recommendations for each user. The user-specific component is introduced in the knowledge-aware attention mechanism which explains the varying importance of the neighboring entities. To sum up, we have made the following contributions in this work:

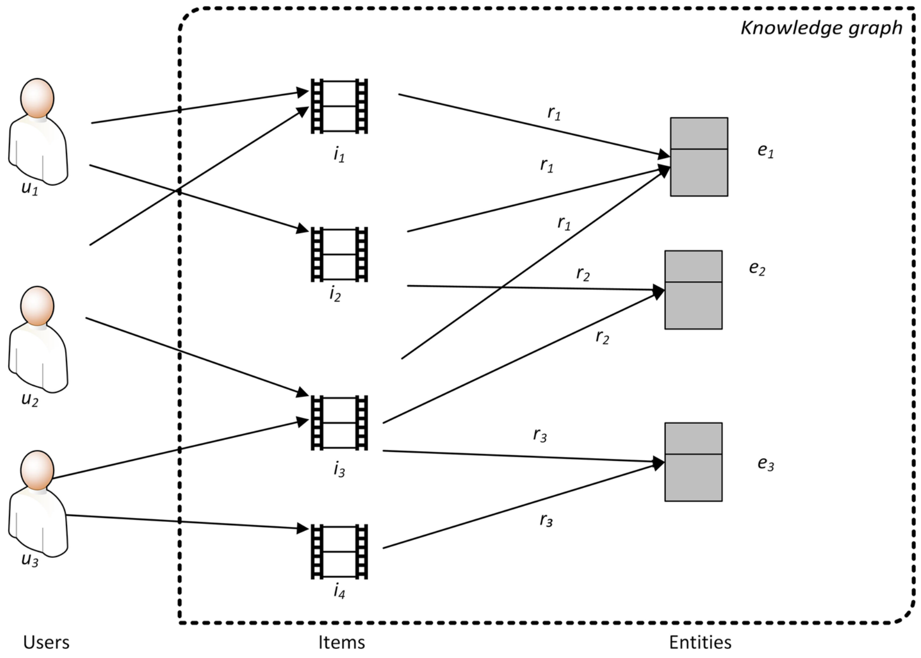


Fig. 2 Example of user–item interaction with enhanced knowledge graph information

- We have proposed the recommendation model known as GACF which is an end-to-end framework that incorporates the collaborative effect and knowledge association, to provide side information, in an explicit manner.
- We highlight the importance of having a semantic rich contextual representation of an entity, which can assign different importance levels to its neighboring entities.
- A user-specific attention network is deployed which, for each user u, assigns different weights to neighboring entities in KG for better-quality recommendation.
- Extensive experiments on three publicly available benchmark datasets have been done to validate the effectiveness of our model. Experimental results show that our model outperforms the current state-of-the-art baselines.

The rest of this paper is organized as follows; in Sect. 2, some recent and relevant studies have been reviewed. In Sect. 3, the problem formulation is reported, which is studied in this work. The proposed recommendation model is discussed in Sect. 4, whereas the experimental results and discussions are summarized in Sect. 5. Finally, this work is concluded by discussing the future directions and potential research work in Sect. 6.

2 Related Work

This section presents the recent work done in the field of recommendation system utilizing KG and GNN. The section first lists the contributions based on KGs followed by the recent works on graph neural networks for the recommendation systems.

2.1 Recommendation with knowledge graph

In the literature, there exist different KG-based recommendation methods having main focus to improve the recommendation performance by integrating the KG. Broadly speaking, a KG-based recommendation method can fall into three categories: regularization-based, path-based, and propagation-based approach.

In regularization-based methods, the regularization term is considered in the loss function, which captures the KG structure and learns entity embeddings. In [16], the authors proposed the collaborative knowledge base embedding method in which item KG is used to derive the semantic entity representations. These enhanced representations are useful in improving the quality of recommendations. Since KG is often incomplete, some authors [17] have used transfer learning approach in which item recommendation as well as KG completion tasks are jointly trained by sharing item and entity embeddings. However, the problem with this approach is that relations among the entities are ignored while transferring the knowledge from KG. In a KG, relations among entities play a vital role in a user-specific personalized recommendation. So, the problem of ignoring relations in transfer learning is addressed in [14] by aligning the items with the relations in KG, and items are aligned with corresponding entities. Furthermore, a cross and compress unit is introduced [18] which model multi-hops feature collaborative signal between items and entities, thus aligning the items with the corresponding entities in KG. Although the regularization-based methods are adaptable, there is also one drawback in them. The explicit modeling of semantic relations is missing which essentially means that they do not consider the relations in KG for making the recommendation.

In path-based methods, various types of connectivity among entities in KG are presented in the past to provide additional information for the recommendation. There are two approaches to cope with the large number of paths that occur between several entities. One of them makes use of selection technique which picks out the most significant paths [10, 19]. The other approach is to use meta-path patterns to limit the paths [20]. The main problem with path-based algorithms is that path selection is difficult to optimize in practice. Moreover, domain expertise is necessary for constructing significant meta-paths, which is time and labor-intensive to be designed by hand, especially in the case of highly complex KG having multiple relations and entities. To improve the recommendation quality, authors in [21] have developed an attribute-rich Heterogeneous Information Network (HIN). Using a Bayesian ranking model, estimation and decomposition of the meta path-based preference matrix between user and item in KG are done in [22]. To learn about users' personalized preferences, the Bayesian ranking model is extended in [23] by the formation of users' clusters which are based on the interest of users. The weighted graph and weighted meta-path models are introduced [24] to more delicately highlight object relations by varying link data points for precisely capturing the semantic relations among users. The authors in [25] have built the heterogeneous neighborhood of a given node by employing the meta path-based random walk method, and the node embedding is learned by employing a heterogeneous skip-gram model. Moreover, to create a node sequence for HIN embedding, [26] uses a meta path-based sampling method. Since the path-based methods depend on the manually selected paths/graphs, domain knowledge is a key resource involved in them. This is the main drawback of path-based methods. Since the proposed model falls in the category of propagation-based methods, more attention is given to that category.

In propagation-based methods, propagation is accomplished in an iterative manner across the whole KG to find supplementary knowledge for the recommendation. This approach has recently gained a lot of attention. For example, RippleNet [13], which has demonstrated success in KG-based recommendation, investigates users' potential preferences through KG links, even though the importance of connections in the framework is poorly defined. KGNN-LS [3] and KGCN [27] utilize the Graph Convolutional Network (GCN) [28] for acquiring item embeddings from their neighbors to enrich their representations. This way of acquiring items from neighborhood demonstrates the importance of neighboring nodes in the recommendation task. However, both these approaches have not considered explicit collaboration signals, leading to inadequate item embeddings. To enhance entity embeddings, KGAT [29] introduces a Collaborative Knowledge Graph (CKG), which combines User–Item bipartite Graph (UIG) and Knowledge Graph (KG) and conducts recursive propagation across CKG through GCN. One deficiency with KGAT is that it treats users as nodes in the same way as CKG treats entities, making it difficult to deal with new users. Once a new interacting user is identified, KGAT must reconstruct the CKG and retrain the entire model to provide predictions, which is expensive in terms of computational resources. Furthermore, KGAT considers that the interacted items in UIG and the related entities in KG are homogenous nodes, despite the fact that they are in distinct latent spaces. Additionally, because the user may interact with different types of items which are generally diverse, high-order interactions in UIG might have totally different interpretations than the latent semantic representation of the actual item, thus injecting noise into embeddings. CKAN [30] introduces the concept of heterogeneous propagation, which integrates collaboration propagation and knowledge graph propagation. It considers interaction and knowledge as information in separate spaces and integrates them in a natural manner such that they contribute to the embeddings with different weights. Moreover, CKAN utilizes knowledge-aware attentive embedding that learn different weights of neighboring nodes, thus assuming that different neighbors contribute different in enriching the target node in KG under diverse situations. One thing to note is that in CKAN, attention mechanism only considers head and relation information which lacks the contextual knowledge for the given target node.

2.2 Recommendation with graph neural network

In recent years, GNN [31] is proposed which are specific to graph structure data, where the concept is based on neural networks. GCN is the extension of a Convolutional Neural Network (CNN) which operate by aggregating neighborhood information iteratively to enhance the representation of the target node. This operation is performed for each node in the graph; thus, each node has information about its directly connected neighbors and their neighbors (high order connections). In the literature, several GCN methods have been introduced that are of two broad types: spectral-based methods and spatial-based methods.

In spectral-based methods, convolution is not done directly on the graph data rather convolutional operation is defined in the Fourier domain. The authors in [32] have done eigen decomposition in the Fourier transform. The first-order approximation of spectral convolutions on graphs may be used to induce a basic convolutional architecture [5]. Since the proposed method falls in the type of spatial-based methods, more focus is given on spatial methods.

In spatial-based methods, operations are carried out directly on graphs which means that information of a node is propagated via edges to the directly connected nodes (neighbors of the node). The first introduced spatial-based method works by summing the neighboring nodes information for the given target node [33]. Afterward, the residual connection at each layer is applied so that the information from the previous layers is memorized. Since each node in the graph has a different number of neighbors, the sampling approach is also utilized to select the fixed number of neighbors for a given node followed by the aggregator to aggregate the features of the neighboring nodes [4]. GAT [34] is introduced which argues that each neighboring node has a different contribution (i.e., importance) to a given target node. In [35], graph knowledge is effectively described by considering the dual GCN having two graph convolutional layers which deal with local and global consistency. RetaGNN [36] is sequential-based recommendation where for each user, its interacted items are given and the task is to recommend the next item to each user. LightGCN [37] is another GNN-based model which is proposed to improve the GCN by making it light model. For this purpose, they have argued that feature transformation and nonlinearity adds the complexity in the model and even worsens the performance of the model. IMP-GCN [38] aims to alleviate the over-smoothing problem which is introduced in the aggregation of high-order neighbor's information. One shortcoming of LightGCN and IMP-GCN is that they treat each neighbor uniformly without discriminating them with attention mechanism.

3 Preliminaries

Before going into the details of the proposed framework, we first introduce the concept of KG and KG-based recommendation, also the set of notations used in this work.

3.1 User–item interaction information

In a typical recommendation system, we assume that there exists a set of users $\mathcal{U}$ and a set of items I. Based on the user–item interaction (e.g., views or purchases), we have an adjacent matrix Y such that $y_{ui} = 1$ if user $u \in \mathcal{U}$ have interacted with item $i \in I$ (also called as observed interaction) else $y_{ui} = 0$. One important thing to note here is that $y_{ui} = 0$ does not mean that user u is uninterested in the item i rather there is a possibility that user u has skips that accidentally, or maybe the user is unaware about item i. $u \in \mathcal{U}, i \in T$.

3.2 Knowledge graph information

In addition, we have utilized KG to incorporate the side information as they provide a way to fuse heterogeneous information into the vectorized representation of items and users. Formally, $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ is the KG where each triple (h, r, t) represents that there exists some relation r between the head $h \in \mathcal{E}$ and tail $t \in \mathcal{E}$ entities. $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and relations, respectively, in the KG. To understand the knowledge triple, consider the triple example, (*Leonardo*, *ActorOf*, *Logan*) represents the fact that Leonardo is an actor in the movie Logan. One thing to note here is that the $\mathcal{R}$ contains the relations in both canonical directions, i.e., (*ActorOf*) and its inverse relation is (*ActedBy*). Moreover, we utilize a set $\mathcal{A} = \{(i, e) \mid i \in I, e \in \mathcal{E}\}$, which essentially means that item i can be aligned with entity e in the KG.

Once we have interaction matrix Y and supplementary knowledge in the form of the KG $\mathcal{G}$, our goal is to predict the probability that some user u would be interested in the item i with which he/she has not interacted before (unobserved interaction). Formally, the task of recommendation is to predict the score function $\widehat{y}_{ui} = \mathcal{F}(u, i; \mathbf{\Theta}, Y, \mathcal{G})$, which predicts the probability that a given user u would like to interact with the item i or not, where $\mathbf{\Theta}$ is model parameters of the score function $\mathcal{F}$. All the notations used in this paper are presented in Table 1 along with their brief description.

Table 1 Notations and their explanations

Notation	Description
$\mathcal{U}$	Set of all users
I	Set of all items
$\mathcal{E}$	Set of entities
$\mathcal{R}$	Set of relations
$\mathcal{G}$	Knowledge graph
$\mathcal{A}$	Alignment set, to facilitates the integration of user–item interaction into $\mathcal{G}$
(h, r, t)	(head, relation, tail) a triplet of knowledge graph
Y	Interaction matrix of users and items
$\widehat{y}_{ui}$	Prediction score or probability
e_u^*	Aggregated representation of a user, after L layers
e_i^*	Aggregated representation of the item, after L layers
$\boldsymbol{\Theta}$	All the model's parameters
$\mathcal{E}_u^{l+1}$	User entity set at $l + 1$ propagation layer
$\mathcal{E}_i^{l+1}$	Item entity set at $l + 1$ propagation layer
J_u^{l+1}	User triple set after $l + 1$ propagation layer
J_i^{l+1}	Item triple set after $l + 1$ propagation layer
$\mathcal{L}_{\mathrm{CE}}$	The loss function of the model

4 Proposed framework

This section explains the proposed recommendation framework along with the flow of information. The framework of the proposed recommendation model is visually presented in Fig. 3. In the proposed model, there are four main components: (1) exploiting user–item interaction information; (2) exploiting KG information; (3) user-specific attention network, and (4) model prediction. The detail of these components is given in the following sections. The proposed framework is an end-to-end recommendation model which integrate the user–item interaction graph and knowledge graph information. Knowledge-aware attention mechanism is utilized which explicitly integrates those neighbors for a given entity which have similar relation. In this way, noisy entities are not introduced in the aggregation step, thus improving the recommendation performance. Moreover, user-specific attention mechanism is utilized which generate personalized recommendation for each user.

4.1 Exploiting user–item interaction information

In the recommendation scenario, historical items of the users are used to represent the user preference. These interacted items also enrich the user's representation thus providing more information about the user. We argue that these interacted items represent the user preference to some extent and some more information may also be needed for a better-quality recommendation. Hence, the set of items the user has interacted with is used to construct the initial entity set. This initial entity set is then propagated into KG by the links between the items and entities. This alignment between the items and entities in KG forms the alignment set $\mathcal{A}$

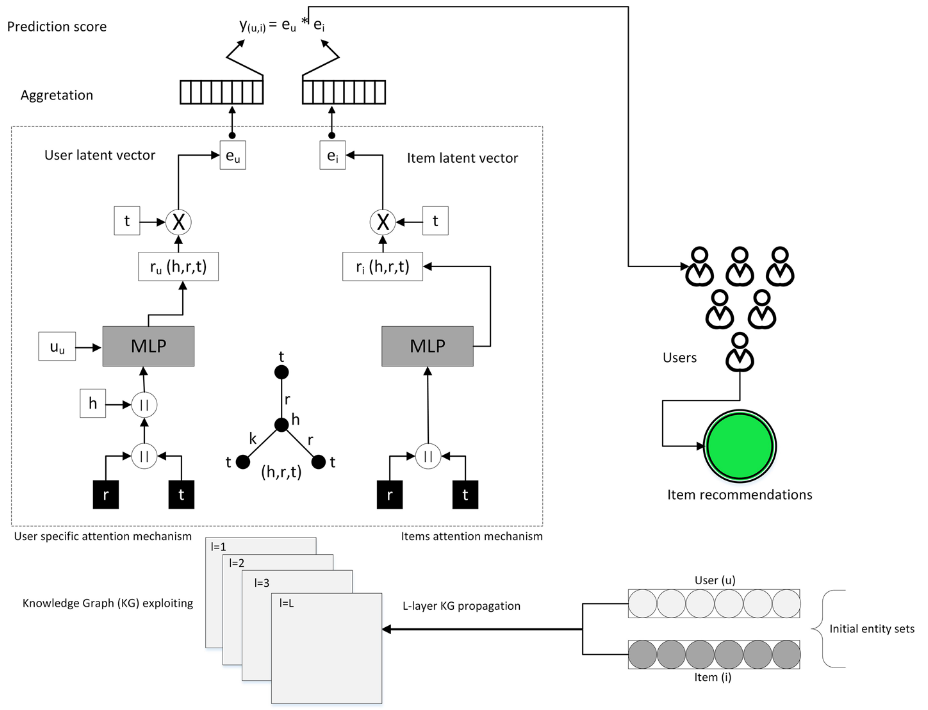


Fig. 3 The proposed recommendation model

which facilitates the integration of user–item interaction into the KG. For a user u, the initial entity set is given as;

$$\mathcal{E}_u = \{\text{e} \mid (i, \text{e}) \in \mathcal{A} \text{ and } i \in \{i \mid y_{ui} = 1\}\} \tag{1}$$

Similarly, we enrich the item representation by the users who have consumed those items. Thus, an item representation is contributed by the neighboring users who interact with it. This collaborative effect of items, which have been used by the user u, contributes toward the collaborative set of items which is defined as;

$$I_i = \left\{i_u \mid u \in \{u \mid y_{ui} = 1\} \text{ and } y_{ui_u} = 1\right\} \tag{2}$$

Once we have a collaborative set of items and alignment set, we can construct the initial entity set of items i which is given as;

$$\mathcal{E}_i = \{\text{e} \mid (i_u, \text{e}) \in \mathcal{A} \text{ and } i_u \in I_i\} \tag{3}$$

4.2 Exploiting KG information

KG is a weighted graph where nodes represent head and tail entities and the edges are relations between those entities. The triple set is in the structure of head, relation, and tail (h, r, t) which is useful to be utilized for knowledge propagation to enrich each entity representation [39]. The proposed model is a propagation-based framework in which local (one-hop) and non-local (neighbors of neighbors) entities are aggregated by propagating along with KG links. The advantage of the propagation-based method is that no domain-specific knowledge is required in designing paths for long-range entities, as in the case of path-based methods. Moreover, propagation-based methods do not require manual feature engineering rather those features are learned by GNN layers.

In entity representation learning, the two main steps involved are propagation and aggregation. In the propagation step, each entity propagates the contextual information to its neighboring entities iteratively. Aggregation is the cumulative effect of neighboring entities on the target entity. The single GCN layer propagates and aggregates contextual information of its directly attached neighbors which essentially means that L layers are required to encode contextual information up to L hops neighbors. This process is done iteratively, and mathematically it can be formulated for user and item entity as given;

$$\mathcal{E}_u^{l+1} = \left\{t \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^l\right\} \tag{4}$$

$$\mathcal{E}_i^{l+1} = \left\{t \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_i^l\right\} \tag{5}$$

where, $\mathcal{E}_u^{l+1}$ is a user entity set at $l+1$ layer, $l = 1, 2, \ldots, L$. t represents the neighboring entity of the target entity h. Once user and item entity sets are constructed, they are utilized in the formation of user and item triple set given as,

$$J_u^{l+1} = \left\{(h, r, t) \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_u^l\right\} \tag{6}$$

$$J_i^{l+1} = \left\{(h, r, t) \mid (h, r, t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_i^l\right\} \tag{7}$$

J_u^{l+1} and J_i^{l+1} are the user and item triple set, respectively, obtained after $l+1$ layer propagation. In this way, KG is exploited in the recommendation scenario to provide auxiliary information as well as to enhance the model's capability by extended entity sets and triple sets.

4.3 User-specific attention network

In the KG, each entity is surrounded by different entities having varying relations and importance to the target entity. Therefore, it would not be appropriate to directly apply GNN and aggregate the neighboring entities' feature information without taking diverse relations and importance into consideration. Moreover, user personalized preference is necessary to be taken into account in the recommendation system. For this reason, the proposed model exploited the user-specific attention mechanism which appropriately combines this information. GACF calculates different attention scores for each user to ensure that user-specific information is captured, and personalized recommendations would be presented to each user.

For a given entity, its relation r with other entities has a different semantic as compared to other neighboring entities with different relations. In the proposed model, this is incorporated by a linear transformation of relation embedding e_j^r and neighborhood entity embedding e_j^t as given by,

$$k_{rt} = \left(\mathrm{e}_j^r \| \mathrm{e}_j^t\right) W \tag{8}$$

where $W \in \mathbb{R}^{2d \times d}$ is the trainable weight matrix, the concatenation operation is denoted as $\|$, and k_{rt} represents the integrated form of embeddings of relation and neighborhood entities.

The head entity's embedding is integrated with k_{rt} to form the enhanced representation of the target entity e_i^h, followed by nonlinear transformation as given by;

$$\pi_u(h, r, t) = \tanh\left[\left(\mathrm{e}_j^h \| k_{rt}\right) W_1 + b\right] * u_u \tag{9}$$

where $W_1 \in \mathbb{R}^{2d \times d}$ and $b \in \mathbb{R}^{1 \times d}$ are trainable weight matrix and bias vector, respectively, u_u is user-specific component that assures that different attention score is being calculated for each user u and is given as,

$$u_u = \mathrm{ReLU}[p_u W_2 + \mathrm{b}] \tag{10}$$

The utilization of p_u in Eq. 10 makes it user-specific, and it has ability to reflect the personalized preferences for each user. Moreover, p_u is the users' embedding which is obtained from the embedding table containing user indices of each user. Henceforth, the SoftMax function [40] is applied for normalizing the coefficients throughout the triples present in the user and item triple set.

$$\pi(h, r, t) = \frac{\exp[\pi_u(h, r, t)]}{\sum_{(h,\tilde{r},\tilde{t}) \in J^l} \exp\left[\pi_u(h, \tilde{r}, \tilde{t})\right]} \tag{11}$$

Now, we have the normalized coefficient of each neighborhood entity for each user, the attention score is then calculated for the tail entity as,

$$e_u = \sum_{J^l} \pi(h, r, t) \mathrm{e}_j^t \tag{12}$$

Here, e_j^t represents the neighboring entity (tail entity) for which the attention score is to be calculated. As the neighboring entities may have different importance to the head entity, so for each neighboring entity, attention score will be calculated. Similarly, for the item, we have the following formula,

$$e_i = \sum_{J^l} \pi(h, r, t) \mathrm{e}_j^t \tag{13}$$

where, $\pi(h, r, t)$ in Eq. 12 determines the attention weight for each head entity in the j^{th} triple of l^{th} layer triple set. Moreover, $\pi(h, r, t)$ operation is accomplished by the usage of single feed-forward network layer, and it can be extended by stacking multiple neural networks layers to aggregate information from non-local neighbors and to enrich target node representation. Thus, at the l^{th} layer, the representation set R of user u and item i formed would be given by,

$$\begin{aligned} R_u &= \left\{\mathrm{e}_u^{(0)}, \mathrm{e}_u^{(1)}, \mathrm{e}_u^{(2)} \ldots, \mathrm{e}_u^{(L)}\right\} \\ R_i &= \left\{\mathrm{e}_i^{(0)}, \mathrm{e}_i^{(1)}, \mathrm{e}_i^{(2)} \ldots, \mathrm{e}_i^{(L)}\right\} \end{aligned} \tag{14}$$

4.4 Model prediction

Since each layer outputs different representations for a user and/or item, so there is a need to aggregate all these multiple representations from L layers and formulate the final representation for both user and item as a single vector. For this purpose, we have employed three types of aggregators which aggregates the user and item representations. In the recent literature, these aggregators are being extensively used in the recommendation scenario [29] and [30]. This is the reason we have chosen these aggregators in our work.

GraphSage Aggregator: This aggregator concatenates all the representations in the representation set R [4]. After aggregating representation, it applies the nonlinear transformation to filter the required or necessary propagation information.

$$f^u_{\text{GraphSage}} = LeakyReLU\Big(\mathrm{W}\cdot\Big(\mathrm{e}_u^{(i_1)}\|\mathrm{e}_u^{(i_2)}\|\ldots\|\mathrm{e}_u^{(i_n)}\Big)+\mathrm{b}\Big) \tag{15}$$

$$f^i_{\text{GraphSage}} = LeakyReLU\Big(\mathrm{W}\cdot\Big(\mathrm{e}_i^{(i_1)}\|\mathrm{e}_i^{(i_2)}\|\ldots\|\mathrm{e}_i^{(i_n)}\Big)+\mathrm{b}\Big) \tag{16}$$

where, LeakyReLU is the activation function, $\mathrm{e}_u \in R_u, \mathrm{e}_i \in R_i$. W and b are learnable weight matrix and bias vector, respectively.

GCN Aggregator: In this aggregator, the representations of users and items in the latent representation set are added to have the final representation [5]. Afterward, nonlinear transformations are applied to it.

$$f^u_{\text{GCN}} = LeakyReLU\left(\mathrm{W}\cdot\sum_{\mathrm{e}_u\in R_u}\mathrm{e}_u+\mathrm{b}\right) \tag{17}$$

$$f^i_{\text{GCN}} = LeakyReLU\left(\mathrm{W}\cdot\sum_{\mathrm{e}_i\in R_i}\mathrm{e}_i+\mathrm{b}\right) \tag{18}$$

Pooling Aggregator: In this aggregator, the maximum value from the representation set is taken and then nonlinear transformation is applied to it.

$$f^u_{\text{Pooling}} = LeakyReLU\big(\mathrm{W}\cdot \mathrm{pool}_{max}(R_u)+\mathrm{b}\big) \tag{19}$$

$$f^i_{\text{Pooling}} = LeakyReLU\big(\mathrm{W}\cdot \mathrm{pool}_{max}(R_i)+\mathrm{b}\big) \tag{20}$$

The aggregated representations for a user are represented as e_u^* and for an item, it is e_i^*. Once we have aggregated representations of user and item, we perform inner product to predict a probability score of the user's preference for the item as;

$$\widehat{y}(u,i) = \mathrm{e}_u^{*\top}\mathrm{e}_i^* \tag{21}$$

4.5 Model training

Before we go into the details of training and loss, firstly, it is necessary to understand the concept of positive and negative interactions. Positive interactions of a user with the items are the actual interactions that indicate that the user either liked, consumed, or watched an item. In the same way, negative interactions of a user are those which are unobserved or un-interacted, these are generated by negative random sampling of items for each user. Thus, for each user, we have generated negative samples from his/her un-interacted items. The number of these negative samples is kept equal to the size of positive samples for each user, to have a balanced ratio of both as well as to assure the effect of model training. The loss function of the proposed recommender system is given by:

$$\mathcal{L}_{\text{CE}} = \sum_{u\in\mathcal{U}}\left(\sum_{(u,i)\in R^+}\mathcal{J}\big(y_{(u,i)},\widehat{y}_{(u,i)}\big)-\sum_{(u,j)\in R^-}\mathcal{J}\big(y_{(u,j)},\widehat{y}_{(u,j)}\big)\right) \tag{22}$$

where, R^+ represents the positive (interacted) interaction between user u and item i, R^- indicates sampling set which is obtained by random negative sampling, while $\mathcal{J}$ is the cross-entropy loss, whose formula is given as;

$$\mathcal{J} = \frac{1}{N}\sum_{i=1}^{N} -(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)) \tag{23}$$

As we know, predicted probabilities (scores) are within the range (0, 1), whereas log function has negative values in this range (0, 1). To make the loss value positive, it is multiplied with a negative log. Finally, we have the following objective function which learns the parameters of a model and minimizes the loss obtained from Eq. (22).

$$\underset{\Theta}{min}\mathcal{L}_{\text{CE}} + \lambda\|\Theta\|_2^2 \tag{24}$$

Θ indicates all the model parameters, while $\|\Theta\|_2^2$ indicates L2-regularizer which is parameterized by λ. Algorithm 1 explains how the optimization is achieved and prediction score is calculated.

```
Algorithm 1 Model Optimization
Input:
    User-item interaction graph,
    Knowledge graph $\mathcal{G}$

Output:
    An optimized model,
    Predicted score $\hat{y}(u, i)$

1.  Initialize all model's parameters randomly
2.  for epoch = 1 to max_epoch do
3.      Select batch of user triples and item triples from $J$;
4.      Perform forward propagation on $\mathcal{G}$;
5.      Calculate CTR prediction score $\hat{y}(u, i)$;
6.      Calculate gradients from Eq. (23) for user and item triples in the batch;
7.      Update models' parameters $\boldsymbol{\Theta}$ by applying gradient descent with learning rate;
8.  end for
9.  return $\hat{y}(u, i)$
```

5 Experiments

This section presents the conducted experiments and obtained results. Here, the proposed model is evaluated on three publicly available datasets, to answer the following research questions.

RQ-1 How does the proposed model, i.e., GACF perform when compared with state-of-the-art KG-based recommendation models?

RQ-2 How do different modules of GACF (variants of GACF) influence the performance of base GACF?

RQ-3 How effectively does the user-specific component describe the user customized preferences on different benchmarks?

Table 2 Statistics of datasets used

	ML	FM	BC
# Users	138,159	1872	17,860
# Items	16,954	3846	14,967
# Interactions	13,501,622	42,346	139,746
# Avg- interactions	98	23	8
Entities	102,569	9,366	77,903
Relations	32	60	25
Triples	499,474	15,518	151,500

RQ-4 How does a variety of settings of hyper-parameters (e.g., depth of layer, aggregation function, user and item triplet set size, etc.) influence the proposed model GACF?

5.1 Datasets

The effectiveness of the proposed model (GACF) is evaluated by conducting experiments on three public benchmarks named Movielens-20 M,[1] Last-FM,[2] and Book-Crossing[3] (respectively, denoted by ML, FM, and BC). These datasets have different sizes and sparsity levels. Furthermore, these datasets belong to different domains as well, i.e., ML belongs to movie recommendation, FM belongs to an online music system, and BC belongs to the book-crossing community.

Movielens-20 M (ML) In this dataset, there are almost 20 million explicit ratings of more than 138 thousand users. This dataset is widely used in the movie recommendation scenario.

Last-FM (FM) It is an online music listening system where musical tracks are viewed as items.

Book-Crossing (BC) In this dataset, readers of different books gave ratings from 0 to 10 scale. This dataset is collected from the book-crossing community.

In all these datasets, ratings are mentioned as explicit feedback, so there is a need to convert them as implicit feedback where the digit 1 shows that the observed (positive) interactions, whereas 0 indicates the unobserved (negative) interaction. In FM and BC datasets, all the ratings are kept as the observed (positive) implicit feedback, since these datasets are sparse as compared to ML, where we only keep those interactions as implicit feedback which have ratings greater than or equal to 4. Besides user–item interactions, KGs of these datasets are also utilized which are currently publicly available at github.[4] Given the whole KG, sub-KG is constructed for each dataset whose purpose is to retain only those triples in KG having a confidence level above 0.9. In sub-KGs, we search the items and entities by their names and if there is no match, we remove that item and entity. Moreover, we also remove those items and entities which are being matched with multiple entities. The statistics of these datasets are summarized in Table 2.

[1] https://grouplens.org/datasets/movielens/20m/.

[2] https://grouplens.org/datasets/hetrec-2011/.

[3] http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

[4] https://github.com/xiangwang1223.

5.2 Baselines

The proposed model, GACF, is compared with state-of-the-art recommender models to check the effectiveness of our method. The qualitative comparison is recorded in Table Five which shows the relatedness of baselines to the proposed framework. The baselines along with their brief description are listed as follows.

- *RippleNet* [13] It is one of the propagation-based methods, which works by propagating users' preferences of items over the entities via links in KG in such a way that user's representation is enriched.
- *KGNN-LS* [3] It is the state-of-the-art method that computes user-specific item embedding through propagation and aggregation of neighboring entities. Label smoothness regularization is utilized which provides inductive bias.
- *KGAT* [29] It is also a propagation-based model based on spatial GCN methods. It integrates user–item interaction graph and KG to form the unified graph known as collaborative knowledge graph (CKG). Unlike KGCN, KGAT makes use of an attentive network that differentiate the significance of different neighbors of CKG.
- *KGCN* [27] It is state-of-the-art model based on GCN in which neighborhood information is aggregated selectively and biasedly to learn the structural information of KG along with the users' potential interests. KGCN is the extension of non-spectral GCN by incorporating KG into context.
- *K-NCR* [41] This method is the extension of neural collaborative filtering (NCF) that combines KG and NCF by propagating the information from multi-hop neighbors for items. Moreover, the heterogeneous attention-aware network is also there which assign varying priorities to different neighboring nodes in KG.
- *CKAN* [30] It is state-of-the-art model that is a heterogeneous propagation of collaborative signal and knowledge-aware attention associations. The attention mechanism is relation-aware as it considers the entity's relation when aggregation them.

5.3 Experimental settings

For each dataset, the proportion of training, validation and testing set is kept as 6:2:2, respectively. This distribution is chosen as it is widely used in previous studies [41] and [15]. In this work, two recommendation scenarios are considered. These are; (a) Click-Through Rate (CTR) prediction is considered, which evaluates the model using AUC and F1 scores. In CTR prediction, the training set first trains the model and then, using the interaction from the testing set, the prediction probability is determined; 2) Top-K recommendation where the default value of K is set to 20. Recall@K and ndcg@K are considered the performance measure in the top-k recommendation, due to their wide usage in the recommendation scenario. For the optimization, Adam [42] optimizer is used which optimize our model, and the batch size chosen is 1024. For initializing the model parameters, Xavier initializer [43] is adopted.

We implement our model using Pytorch, and the hyper-parameters are selected using the grid search. The embedding size is chosen among $\{4, 8, 16, 32, 64, 128\}$, and the learning rate is chosen among $\left\{10^{-3}, 4 \times 10^{-3}, 10^{-2}, 4\times 10^{-2}\right\}$, whereas the regularization parameter is chosen among $\left\{10^{-5}, 5 \times 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\right\}$. The appropriate size of user triple set and item triple set is chosen from the $\{4, 8, 16, 32, 64\}$. The attention mechanism adopted in this study is found by doing experiments using a different number of layers. The optimal number of layers found is 2 for the attention mechanism. The dimensions of the hidden layer are equal to the dimensions of the embedding. We have used the open source implementation of the baselines, to obtain the results on three benchmark datasets. For all the comparison methods, the hyper-parameters are chosen either empirically or as mentioned in the original papers.

5.4 Performance comparisons (*RQ-1*)

Extensive experiments are performed to assess the effectiveness of the proposed GACF model with state-of-the-art models, and results are recorded in Table 3 (CTR prediction) and Table 4 (top-k recommendation). In Table 3 and Table 4, the best performance is highlighted with boldface, while second best is highlighted with underline. The following explanations are made from the experiments.

- GACF significantly outperforms when compared with all baselines using Wilcoxon signed rank statistical test [44, 45] ($p < 0.5$). This shows the significance of exploiting relation with neighboring entities as well as user-specific component into consideration. More formally, GACF improve the performance when compared with second best baseline by 1.2%, 2.7% and 1.9% in ML, FM and BC dataset, respectively, w.r.t AUC. One thing to note here is that GACF is not as much improved in case of ML dataset due to the huge number of

Table 3 Experimental results w.r.t AUC and F1 score. The best performance is highlighted with boldface, while second best is highlighted with underline

Dataset	Model	AUC	F1 score
ML	Ripple Net	0.971	0.921
	KGCN	0.969	0.924
	KGNN-LS	0.972	<u>0.926</u>
	KGAT	0.974	0.924
	CKAN	<u>0.975</u>	0.925
	K-NCR	0.971	**0.928**
	$\text{GACF}_{/\text{concat}}$	0.973	0.921
	GACF	**0.986**	<u>0.926</u>
FM	Ripple Net	0.769	0.701
	KGCN	0.791	0.704
	KGNN-LS	0.801	0.718
	KGAT	0.821	0.740
	CKAN	<u>0.840</u>	<u>0.758</u>
	K-NCR	0.754	0.697
	$\text{GACF}_{/\text{concat}}$	0.838	0.752
	GACF	**0.863**	**0.785**
BC	Ripple Net	0.714	0.641
	KGCN	0.681	0.632
	KGNN-LS	0.669	0.631
	KGAT	0.721	0.653
	CKAN	<u>0.749</u>	0.669
	K-NCR	0.716	<u>0.678</u>
	$\text{GACF}_{/\text{concat}}$	0.745	0.674
	GACF	**0.763**	**0.681**

Table 4 Experimental results w.r.t. ndcg@k and recall@k (top-k recommendation)

Model	ML		FM		BC	
	Ndcg	recall	ndcg	recall	ndcg	recall
Ripple Net	0.153	0.211	0.124	0.153	0.135	0.061
KGCN	0.157	0.201	0.135	0.220	0.131	0.054
KGNN-LS	0.161	0.213	0.132	0.195	0.139	0.051
KGAT	0.191	0.230	0.142	0.234	0.201	<u>0.068</u>
CKAN	<u>0.194</u>	<u>0.232</u>	0.149	0.221	0.197	0.063
K-NCR	0.183	0.225	<u>0.175</u>	**0.224**	<u>0.205</u>	0.065
GACF	**0.198**	**0.240**	**0.184**	<u>0.241</u>	**0.211**	**0.072**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

average interactions per user, thus making propagation and enriching representations as less useful.

- Among baselines, CKAN usually shows best performance in terms of AUC and this highlights the importance of heterogeneous propagation along with knowledge-aware attention mechanism to differentiate neighboring entities importance. On the other hand, KGCN shows worse performance in ML dataset and second worse in FM and BC dataset. This is because KGCN does not consider explicit encoding of collaborative signal into entity embedding.
- KGAT show best performance after CKAN in terms of AUC, which again verifies the usefulness of incorporating knowledge-aware associations into the embedding process. This lag in performance justifies that item in user–item interaction should not be assumed as same to that of entities in KG. In actual, this homogeneous propagation of nodes in not reasonable as user may interacts with broad category of items which should not be considered equally.
- Among the three public benchmarks, ML shows best performance (above 96%) in all the baselines, followed by FM and then BC. This behavior among these datasets is due to the different number of average interactions per user. For example, ML is large dataset as compared to BC and FM, and ML has large number of average interactions and links per user (around 98), so less information is learnt from latent vector embedding due to the rich set of interaction is already contain.
- In FM, GACF outperforms Ripple Net, KGCN, KGNN-LS, KGAT, CKAN and K-NCR by 12.2%, 9.1%, 7.7%, 5.1%, 2.7% and 14.5%, respectively, in terms of AUC. Furthermore, in BC, GACF outperforms Ripple Net, KGCN, KGNN-LS, KGAT, CKAN and K-NCR by 6.9%, 12.0%, 14.1%, 5.8%, 1.9% and 6.6%, respectively, in terms of AUC. Moreover, in ML dataset, GACF outperforms Ripple Net, KGCN, KGNN-LS, KGAT, CKAN and K-NCR by 1.5%, 1.8%, 1.4%, 1.2%, 1.2% and 1.5%, respectively, in terms of AUC.
- To verify the usefulness of concatenation operation in Eq. (8), experiments are conducted with sum operation, i.e., by replacing concatenation operation with sum operation. $\text{GACF}_{/\text{concat}}$ in Table 3 represents the same thing where the concatenation operation is disabled. From Table 3, it can be seen that $\text{GACF}_{/\text{concat}}$ underperforms the GACF (the proposed one). Since concatenation operation is capable of retaining more information as compared to sum operation, performance (in terms of AUC) of GACF is increased in ML, FM and BC by 1.3%, 3.0%, 2.4%, respectively.

- In terms of ndcg@k, GACF outperforms by 2.1% in ML, 5.1% in FM and 2.9% in BC, when compared with the second-best performance (underlined in the table). This improvement once again highlights the importance of incorporating relation-aware attention mechanism. Among baselines, K-NCR outperforms in FM and BC datasets w.r.t ndcg@k. It is due to the hidden layers which allow it to learn the complex features between user and item interactions.
- KGCN shows worst performance on BC dataset, while second worst performance is on ML and FM datasets w.r.t ndcg@k. This shows that by treating all the neighboring nodes of an entity equally introduces noise in the aggregation step. This highlights the importance of attention mechanism which, when incorporated into the model, adds strength to it by discriminating different neighboring nodes based on their relations and user's interests.

5.5 Ablation study (*RQ-2*)

The purpose of the ablation study is to determine the effectiveness of different modules or components of the model by the addition or removal of various components. In this work, we have introduced two variants of GACF and then, performance is assessed by conducting experiments on different datasets (Table 5).

$GACF_{/deepCF}$ In this variant, deep layer propagation is removed and only the first order or local neighbors of an entity are considered. In this way, we have an understanding of the importance of higher-order or global neighbors of an entity in terms of performance.

$GACF_{/Att}$ This variant disables the attention-aware component in the original GACF model and considers that each neighbor of the target entity is contributing equally. By removing the attention component, we took an average of all the neighbors of an entity to determine how important the attention mechanism is in our model.

In Table 6, the experimental results are recorded which describe the importance of deep propagation information as well as the attention aware mechanism which is there in the original GACF model. Table 6 has the following findings.

- In each dataset, the performance of $\text{GACG}_{/\text{deepCF}}$ is worse than the original GACF which highlights the importance of deep collaboration propagation as it adds more capability in the representation of users and items and is thus able to encode useful information in their latent vectors.

Table 5 A qualitative comparison between baselines and the proposed model

Model	Knowledge graph	Attention mechanism	User-specific	Propagation-based model
Ripple Net	✓			✓
KGCN	✓			✓
KGNN-LS	✓		✓	✓
KGAT	✓	✓		✓
CKAN	✓	✓		✓
K-NCR	✓	✓		✓
Proposed GACF	✓	✓	✓	✓

Table 6 Performance of GACF (w.r.t AUC) w.r.t two variants on benchmark datasets

	GACF/$_{deepCF}$	GACF/$_{Att}$	GACF
ML	0.9837	0.9841	**0.9866**
FM	0.8452	0.8081	**0.8635**
BC	0.7287	0.7116	**0.7634**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

Table 7 Effect of the user-specific component on model performance (w.r.t AUC)

	Without user-specific	With user-specific
ML	0.9821	0.9863
FM	0.8512	0.8635
BC	0.7348	0.7634

- GACF$_{/\mathrm{Att}}$ is performing even worse than that of GACF$_{/\mathrm{deepCF}}$ which highlights the effectiveness of the user-specific attention mechanism in the recommendation model. Since KG is a heterogeneous graph having different relations between entities. These relations should not be ignored when aggregating neighboring entities' information to enrich the target entity.

5.6 Effectiveness of user-specific component (RQ-3)

In personalized recommendation, items are recommended based on the user's history and also her closely related entities rather than suggesting the same item to all users. In the proposed recommendation model, the user-specific component is considered since it gives fruitful results. We have conducted experiments by disabling the user-specific component (removing u_u from Eq. 9) and the performance scores are shown in Table 7. The comparison is made by first disabling the user-specific component and then enabling the user-specific component, for each dataset. It can be seen from Table 7 that recommendation performance is enhanced when the user-specific component is considered which clearly shows its effectiveness in GACF.

5.7 Hyper-parameters study (*RQ-4*)

To check the effect of different hyper-parameters such as depth of layer, type of aggregator, size of the user, and item triple set, on the performance of the model, we have conducted extensive experiments. The results along with the possible explanation are recorded in the following sections.

Effect of different aggregators: The proposed model is evaluated on three different aggregators which aggregate user (and item) representations. The experiments conducted for different aggregators are reported in Table 8. It can be seen from Table 8 that the concatenation aggregator outperforms other aggregators such as pool and sum in all datasets. One possible reason for it is that the concatenation aggregator is capable of retaining more useful information than that of sum and pool aggregators. This benefit of concatenation aggregator comes at the cost of more memory usage to retain each tensor. Moreover, sum aggregator gives us only

Table 8 Effect of different aggregators on performance (w.r.t AUC)

	Pool	Sum	Concatenation
ML	0.9264	0.9836	**0.9866**
FM	0.7916	0.8452	**0.8634**
BC	0.7121	0.7345	**0.7632**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

Table 9 Effect of depth of layers (w.r.t AUC)

No. of layers	1	2	3	4
ML	**0.9866**	0.9861	0.9852	0.9847
FM	0.8612	**0.8634**	0.8586	0.8540
BC	0.7627	**0.7633**	0.7587	0.7532

The bold numbers shoes best performance, whereas the underline shoes second-best performance

one single value after combining tensors from different network layers rather than retaining those tensors.

Effect of depth of layers: To determine the model's performance for different depths of layer, we have conducted experiments and the results are given in Table 9. In the table, values 1, 2, 3, and 4 represent the respective depths of a layer. It can be seen that for ML (movie) dataset, the best performance score is achieved when L is taken as 1, whereas, for FM (music) and BC (book) datasets, the best performance is obtained when L is set to 2. It is also highlighted in Table 7 with boldface. One important thing to note here is that as the depth of the model is increased from 2 to 4, recommendation performance is decreased. One of the possible explanations of this is due to noise which is being introduced by incorporating multi-hop's information in the target node embedding. Besides the decrease in performance, the training time of the model is also increased as we increase the depth of the layer.

Effect of triple set size: In the proposed model, triple set size also plays its role in improving the recommendation performance. This can be noted by the experimental results presented in Tables 10, 11 and 12 for ML, FM, and BC datasets, respectively. In our experiments, we have varied the triple set size of user and item from the range of 8, 16, 32, and 64. The best performance is achieved by setting the size of user triple set and item triple set to 64

Table 10 Effect of triple set size on ML dataset (w.r.t AUC)

	Item			
User	8	16	32	64
8	0.9742	0.9755	0.9816	0.9842
16	0.9756	0.9847	0.9854	0.9859
32	0.9857	0.9861	0.9863	0.9863
64	0.9860	0.9862	0.9863	**0.9864**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

Table 11 Effect of triple set size on FM dataset (w.r.t AUC)

	Item			
User	8	16	32	64
8	0.8420	0.8542	0.8557	0.8580
16	0.8433	0.8572	0.8583	0.8602
32	0.8470	0.8545	0.8590	0.8603
64	0.8468	0.8560	0.8601	**0.8635**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

Table 12 Effect of triple set size on BC dataset (w.r.t AUC)

	Item			
User	8	16	32	64
8	0.7274	0.7432	0.7524	0.7589
16	0.7252	0.7374	0.7545	0.7587
32	0.7328	0.7451	0.7563	0.7582
64	0.7309	0.7515	0.7594	**0.7633**

The bold numbers shoes best performance, whereas the underline shoes second-best performance

for all datasets. It is highlighted in the table in boldface. One possible explanation of this performance is when we increase the triple set size, the number of entities also increases, which enriches the representation of user and item.

6 Conclusion and future work

This work presented an end-to-end recommendation model named as Graph Attention-based Collaborative Filtering (GACF). It falls in the category of propagation-based methods, which explicitly encodes the collaborative signal that is latent in user–item interaction information. User and item triple sets were constructed which propagated in KG to enrich their representation as well as to integrate information of the multi-hop neighboring entities. Knowledge-aware attention mechanism was utilized in the model, which explicitly integrated those neighbors for a given entity having similar relation. In this way, noisy entities were not introduced in the aggregation step, thus improving the recommendation performance. Moreover, the attention mechanism assigned different importance to the neighboring entities which was useful in the personalized recommendation. Furthermore, user-specific attention mechanism was also utilized which generated personalized recommendations for each user. To aggregate representations of users and items obtained from multi-layer neural network, three different aggregation functions were employed and their performance was compared. Among these three aggregators, the concatenation aggregator outperforms other aggregators due to its ability to retain more information. The parameters involved in the proposed model were jointly learned, which could capture the latent relationship between user–item interactions. The superiority of GACF was validated by conducting extensive experiments on three real-world datasets of different scenarios.

This work can be extended in multiple ways in the future. For future work, we plan to discover more effective methods for finding the important high order neighbors of an entity to reduce the noise which may introduce in the large receptive field. Moreover, we are also interested in designing an aggregation function that can combine context information of entities and thus, recommendation accuracy can be improved. Furthermore, one can also extend the proposed GACF model on other KG-based recommendation scenarios. Another potential research direction is to explore the attention mechanism in which more emphasize can be given to the integration of head and tail entities to have better recommendation performance.

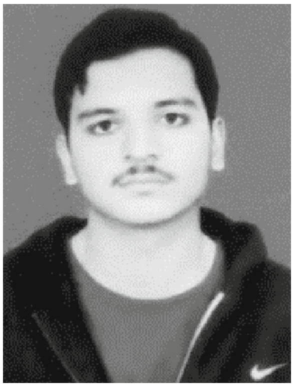
Acknowledgements The authors are indebted to the editor and anonymous reviewers for their helpful comments and suggestions during multiple revisions. The authors would like to thank GIK Institute for providing research facilities. This work was supported by the GIK Institute graduate program research fund under the GA-F scheme.

References

1. Yin C, Wang J, Park JH (2017) An improved recommendation algorithm for big data cloud service based on the trust in sociology. Neurocomputing 256:49–55
2. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv (CSUR) 52(1):1–38
3. Wang H, Zhang F, Zhang M, Leskovec J, Zhao M, Li W, Wang Z (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, pp 968–977
4. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp 1025–1035
5. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks ArXiv e-prints
6. Zhang C, Song D, Huang C, Swami A, Chawla NV (2019) Heterogeneous graph neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 793–803
7. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Comput 42(8):30–37
8. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering in: Proceedings of the 26th international conference on World Wide Web pp 173–182
9. Tay, Y., Tuan, L.A., Hui, S.C.: Latent relational metric learning via memory-based attention for collaborative ranking, in: Proceedings of the 2018 World Wide Web Conference, 2018. pp. 729–739
10. Wang X, Wang D, Xu C, He X, Cao Y, Chua TS (2019) Explainable reasoning over knowledge graphs for recommendation. Proc AAAI Conf Artif Intell 33(1):5329–5336
11. Lin Y, Liu Z, Sun, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion, in: Twenty-ninth AAAI conference on artificial intelligence pp 2181–2187
12. Huang X, Zhang J, Li D, Li P (2019) Knowledge graph embedding based question answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining pp 105–113
13. Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplenet: propagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management pp 417–426
14. Cao Y, Wang X, He X, Hu Z, Chua TS (2019) Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences, in: World Wide Web Conference pp 151–161
15. Yin R, Li K, Zhang G, Lu J (2019) A deeper graph neural network for recommender systems. Knowl Based Syst 185:105020

16. Zhang F, Yuan NJ, Lian D, Xie X, Ma WY (2016) Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining pp 353–362
17. Piao G, Breslin JG (2018) Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model, European Semantic Web Conference, Springer pp 496–511
18. Wang H, Zhang F, Zhao M, Li W, Xie X, Guo M (2019) Multi-task feature learning for knowledge graph enhanced recommendation, in: World Wide Web Conference pp 2000–2010
19. Sun Z, Yang J, Zhang J, Bozzon A, Huang LK, Xu C (2018) Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems pp 297–305
20. Hu B, Shi C, Zhao WX, Yu PS (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 pp 1531–1540
21. Yu X, Ren X, Sun Y, Sturt B, Khandelwal U, Gu Q, Norick B, Han J (2013) Recommendation in heterogeneous information networks with implicit user feedback, in: Proceedings of the 7th ACM conference on Recommender systems pp 347–350
22. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) BPR: Bayesian personalized ranking from implicit feedback, ArXiv e-prints
23. Yu X, Ren X, Sun Y, Gu Q, Sturt B, Khandelwal U,. Norick B, Han J (2014) Personalized entity recommendation: a heterogeneous information network approach, in: Proceedings of the 7th ACM international conference on Web search and data mining pp. 283–292
24. Shi C, Zhang Z, Luo P, Yu PS, Yue Y, Wu B (2015) Semantic path based personalized recommendation on weighted heterogeneous information networks, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management pp 453–462
25. Dong Y, Chawla NC, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining pp 135–144.
26. Shi C, Hu B, Zhao WX, Philip SY (2018) Heterogeneous information network embedding for recommendation. IEEE Trans Knowl Data Eng 31(2):357–370
27. Wang H, Zhao M, Xie X, Li W, Guo M (2019) Knowledge graph convolutional networks for recommender systems, in: Proceedings of the World Wide Web Conference pp 3307–3313
28. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining pp 974–983
29. Wang X, He X, Cao Y, Liu M, Chua TS (2019) Kgat: knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining pp 950–958
30. Wang Z, Lin G, Tan H, Chen Q, Liu X (2020) CKAN: collaborative knowledge-aware attentive network for recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 219–228
31. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: a review of methods and applications. AI Open 1:57–81
32. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs, ArXiv e-prints
33. Micheli A (2009) Neural network for graphs: a contextual constructive approach. IEEE Trans Neural Networks 20(3):498–511
34. Velickovic, P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks, ArXiv e-prints
35. Zhuang C, Ma Q (2018) Dual graph convolutional networks for graph-based semi-supervised classification, in: Proceedings of the 2018 World Wide Web Conference pp 499–508
36. Hsu C, Li CT (2021) RetaGNN: relational temporal attentive graph neural networks for holistic sequential recommendation, in: Proceedings of the Web Conference pp 2968–2979
37. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020) Lightgcn: simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval pp 639–648
38. Liu F, Cheng Z, Zhu L, Gao Z, Nie L (2021) Interest-aware message-passing gcn for recommendation, in: Proceedings of the Web Conference, pp 1296–1305
39. Guo Q, Zhuang F, Qin C, Zhu H, Xie X, Xiong H, He Q (2020) A survey on knowledge graph-based recommender systems IEEE Transactions Knowl Data Eng

40. Memisevic R, Zach C, Pollefeys M, Hinton GE (2010) Gated softmax classification. Adv Neural Information Process Syst 23(2010):1603–1611
41. Sang L, Xu M, Qian S, Wu X (2021) Knowledge graph enhanced neural collaborative recommendation. Expert Syst Appl 164:113992
42. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization, ArXiv e-prints
43. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings 9:249-256
44. Shani G, Gunawardana A (2011) Evaluating recommendation systems, In: Ricci F, Rokach L, Shapira B, Kantor P (eds) Recommender Systems Handbook. Springer, Boston, MA, pp 257-297
45. Halim Z, Rehan M (2020) On identification of driving-induced stress using electroencephalogram signals: A framework based on wearable safety-critical scheme and machine learning. Inf Fusion 53:66–79



Ehsan Elahi received the BS (Hons) degree in Software Engineering from National University of Science and Technology, in 2016 and the MS degree in Software Engineering from COMSATS University in 2019. He is currently working towards his PhD from the Faculty of Computer Science and Engineering, Ghulam Ishaq Khan (GIK) Institute of Engineering Sciences and Technology, Pakistan. Elahi's research interests include: data mining, computer vision, human—computer interaction, recommender systems, and computational intelligence.

Zahid Halim received the BS degree in computer science from the University of Peshawar, Pakistan, in 2004, MS degree in computer science from the National University of Computer and Emerging Sciences, Pakistan, in 2007, and also the PhD degree in Computer Science from the National University of Computer and Emerging Sciences, Pakistan, in 2010. He was with the National University of Computer and Emerging Sciences, Islamabad, Pakistan, as a Faculty Member (Lecturer and then Assistant Professor) from 2007 to 2010. Currently he is a Professor with Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan. His current research interests include machine learning and data mining, probabilistic/uncertain data mining, and human factors in computing. Dr. Halim is a member of the IEEE Computational Intelligence Society.