# Enhancing Recommendation Systems with Knowledge Graphs and Dynamic Preferences

Guanfeng Li *, Yuyin Chen, Yunli Wang, and Feizhou Qin

College of Information Engineering, Ningxia University, Yinchuan 750021, China
Email: Ligf@nxu.edu.cn (G.L.); 12023131985@stu.nxu.edu.cn (Y.C.); 634564759@qq.com (Y.W.); qinfz@nxu.edu.cn (F.Q.)
*Corresponding author

***Abstract*—Integrating knowledge graphs into recommendation systems mitigates data sparsity and cold start issues while offering explanations for recommendations. Capturing user preferences accurately is vital for enhancing performance, but user interests often shift due to contextual and mood changes. Traditional methods overlook preference dynamics, which prompts this paper to propose a knowledge graph based approach considering both long-term and short-term preferences. This approach examines user knowledge graph connections, extracts relevant associations, and integrates them into recommendations. To capture short-term preferences accurately, a bidirectional GRU with an attention mechanism is used. Long-term and short-term preferences are then fused and computed, with optimal parameters determined through experimentation. Experimental results demonstrate that the proposed model improves recall rates by 0.18%, 1.87%, and 1.52% on three datasets, respectively, exhibiting superior performance compared to baseline models.**

***Keywords*—knowledge graph, recommendation system, attention mechanism, user preferences**

## I. Introduction

With the swift progression of information technology and the proliferation of the internet, data volumes have surged dramatically. Although this has significantly enhanced people's lives and work, it has concurrently precipitated the issue of information overload. The challenge of swiftly and effectively identifying the information that users require or are interested in amidst vast quantities of data has become imperative. Recommendation systems address information overload by serving as pivotal instruments in identifying the content that users are most likely to need [1]. By leveraging a myriad of accessible data, these systems delve into and ascertain user interests or preferences, thereby suggesting content that resonates with users' inclinations and attaining more precise recommendation objectives [2].

Knowledge graphs are typically comprised of extensive relationships and facts pertaining to various items, manifesting as directed, heterogeneous graphs. Within these graphs, nodes represent entities and edges represent relationships, forming intricate networks structured as triples—consisting of a head entity, a relationship, and a tail entity. Presently, the focal point of research on recommendation systems based on knowledge graphs lies in augmenting the representational information of items or users. This augmentation entails leveraging attribute information from project nodes that engage with users within the knowledge graph and bolstering item representations through methodologies such as neighborhood aggregation. In contrast, dynamic knowledge graphs serve to capture and represent knowledge as it unfolds and evolves over time. Distinct from traditional static knowledge graphs, dynamic knowledge graphs encompass not only entities and their relationships but also integrate a temporal dimension. These graphs employ timeseries analysis techniques alongside graph similarity methods to scrutinize shifts and trends within the graph's structure as they occur over time. Consequently, such graphs furnish a more holistic comprehension of knowledge and hold substantial applicability in domains such as finance, traffic flow prediction, action recognition, and climate forecasting.

Recommendation systems can primarily be classified into three categories: collaborative filtering-based systems [3], content-based systems [4], and hybrid systems. Furthermore, a significant amount of research delves into recommendation methods employing matrix factorization techniques. Collaborative filtering-based recommendation systems have gained widespread use owing to their proficiency in capturing user preferences and their straightforward implementation across diverse scenarios, which obviates the need for feature extraction, as necessitated by content-based systems. Despite enhancing recommendation performance, collaborative filtering methods encounter several challenges. These include data sparsity [5], cold start problems, and difficulties in capturing dynamic user preferences. To mitigate these issues, there has been substantial interest in integrating auxiliary information into recommendation systems. This includes leveraging social networks, knowledge graphs, user or item attributes, images, and contextual information [6].

Given the inherently dynamic nature of user interests, short-term preferences are frequently overlooked. To address this issue, this study introduces the Long-term and

Manuscript received December 16, 2024; revised January 14, 2025; accepted February 26, 2025; published April 27, 2025.

Short-term preferences-based knowledge graph Recommendation (LSRec) method. This approach constitutes a knowledge graph recommendation system that seamlessly integrates both long-term and short-term user preferences. Through the utilization of cross compression units and a bidirectional Gated Recurrent Unit (GRU) network, LSRec meticulously explores the temporal relationships between users and entities. Consequently, it delivers personalized and highly accurate recommendations.

The remainder of this paper is organized as follows. The related work is reviewed in Section II. Subsequently, Section III introduces our recommendation model. In Section IV, the experiments around this paper are conducted. Finally, we present our conclusion and further research work in Section V.

## II. Related Work

In this section, we will provide a brief review of related research on traditional recommendation methods and knowledge graph-based recommendation methods.

### A. Traditional Recommendation Methods

In the early stages of research, content-based recommendation algorithms were the pioneers, being the first to be proposed and implemented. Subsequently, Goldberg *et al.* [7] introduced collaborative filtering-based recommendation algorithms. These algorithms discern users' potential interests by scrutinizing interactions between users and items, thereby suggesting items that might pique their interest. To mitigate certain constraints inherent in traditional recommendation algorithms, hybrid recommendation algorithms emerged. These algorithms blend various traditional methodologies, thereby augmenting the overall recommendation performance.

With the advent and progression of deep learning, neural networks have incrementally been integrated into recommendation algorithms. Covington *et al*. [8] proposed a recommendation algorithm for YouTube videos that leverages deep neural networks. By embedding user and item data within these networks, this method partially alleviated issues such as data sparsity and the cold start problem, resulting in a significant enhancement of recommendation performance. Cheng *et al*. [9] introduced the Wide & Deep recommendation model, which concurrently trains both a linear model and a deep neural network. This dual approach addresses the overfitting issue commonly associated with deep neural networks, equipping the model with both generalization and memorization capabilities.

In recent years, the incorporation of attention mechanisms within neural networks has attracted considerable attention. Zhou *et al*. [10] introduced the Deep Interest Network (DIN) recommendation model. This model harnesses attention mechanisms to execute weighted pooling of user interests, thereby more accurately reflecting user preferences across various interest points and enhancing the model's recommendation performance. Building upon the NGCF [11] model, He *et al.* [12] leveraged the robust message passing capabilities of convolutional networks. By eliminating the inner product operation of adjacent nodes, they optimized the algorithm's time complexity, further improving recommendation performance. Lin *et al*. [13] utilizing a graph neural network framework, proposed a graph collaborative filtering model that incorporates neighborhood enhanced contrastive learning through an improved contrastive learning method.

When compared to traditional collaborative filtering algorithms, recommendation algorithms based on deep learning excel in handling sparse data by learning low dimensional representations. Notably, recommendation algorithms rooted in graph neural networks deliver more precise and personalized recommendations by modeling the intricate relationships between users and items. However, in contrast to knowledge graph-based recommendation algorithms, graph neural network-based approaches primarily focus on capturing similarities and relationships between nodes. They often lack adequate representation of edge information and exhibit relatively weaker interpretability in terms of recommendations.

### B. Knowledge Graph-Based Recommendation Methods

Knowledge graph embedding-based recommendation methods forge relationships among users, items, and attributes by embedding them into a low dimensional vector space, facilitating effective recommendations. Wang *et al*. [14] introduced a deep knowledge-aware recommendation model that leverages knowledge graphs as edge information. This model seamlessly integrates word embeddings from news headlines with embeddings of pertinent entities extracted from the knowledge graph using convolutional neural networks. Cao *et al*. [15] presented the Knowledge aware User Preference (KTUP) model, which interprets user item interactions as relationships between entities within the knowledge graph. By concurrently training the recommendation task and the knowledge graph completion task, the model transfers knowledge learned from the graph to enhance user item interaction predictions. Hu *et al.* [16] devised a neural collaborative attention model grounded in meta-path contexts. This model captures relationships among various item types using meta-path information, accounts for user interests and historical behavior, and applies graph convolutional networks to perform representation learning on the weighted relationship graph. This process generates meta-path context representations. Subsequently, an attention mechanism is employed to learn relationships between users and meta-path contexts, combining these relationships to create user interest representations tailored for recommendations.

Wang *et al*. [17] introduced the Knowledge Graph User Preference Propagation Network (RippleNet) model, which disseminates user preferences throughout the knowledge graph and constructs paths originating from the user. However, RippleNet does not prominently emphasize the importance of relationships. To rectify this, Zhang *et al*. [18] algamated collaborative, structural, textual, and visual knowledge within a cohesive framework. Nevertheless, the knowledge graph embedding module in their Collaborative Knowledge

Embedding (CKE) approach is better suited for in graph applications. Under the Bayesian framework, there is a lack of tight coupling between the collaborative filtering module and the knowledge graph embedding module, leading to less significant supervision of the knowledge graph within the recommendation system. Expanding on the concepts of CKE, Wang *et al*. [19] proposed MKR, a multitask learning approach aimed at augmenting knowledge graphs in recommendation systems. As a comprehensive end-to-end deep recommendation framework, MKR strives to enhance the performance of recommendation systems through knowledge graph embedding tasks.

## III. Methodology

In this section, we focus on the functions of our proposed model. Firstly, we delineate the symbols that will be employed subsequently. Let $U = \{u1, u2, \ldots\}$ denote the set of users, and $V = \{v1, v2, \ldots\}$ denote the set of items. The interaction matrix between users and items, designated as $Y = \{y_{uv} \mid u \in U, v \in V\}$, is defined on the basis of implicit feedback from users, as follows:

$$y_{uv} = \begin{cases} 1, & (u, v) \text{ has interactions or feedback} \\ 0, & (u, v) \text{ has no interactions or feedback} \end{cases} \quad (1)$$

If the value of $y_{uv}$ is 1, it signifies the existence of an implicit interaction between user u and item v, such as a click, view, or watch, indicating some level of interest or preference from the user towards the item. Conversely, when $y_{uv}$ is 0, it indicates the absence of such an interaction between user $u$ and item $v$, implying that the user has not exhibited a clear interest in the item.

### A. LSRec Model

The Long-term and Short-term preferences-based knowledge graph Recommendation method, termed LSRec, seamlessly integrates users' short-term and long-term preferences within the TKGR model framework. Additionally, it incorporates the RippleNet concept to effectively capture the intricate connections between users and knowledge graph entities. Prior to model training, the system meticulously extracts user entity relationship pairs from the knowledge graph, storing them in a structured format: (user, relationship, knowledge graph entity, time). This preprocessing step ensures a wealth of data to support subsequent training processes.

Fig. 1 shows the LSRec model structure. Within the recommendation module, the initial step involves the interaction and integration of user and item features through cross compression units. Following this, multilayer neural networks are employed for deep learning and feature extraction. Lastly, the short-term preference model is amalgamated to analyze and model the user's recent interactions, culminating in the predicted score, denoted as $\hat{y}_{uv}$.

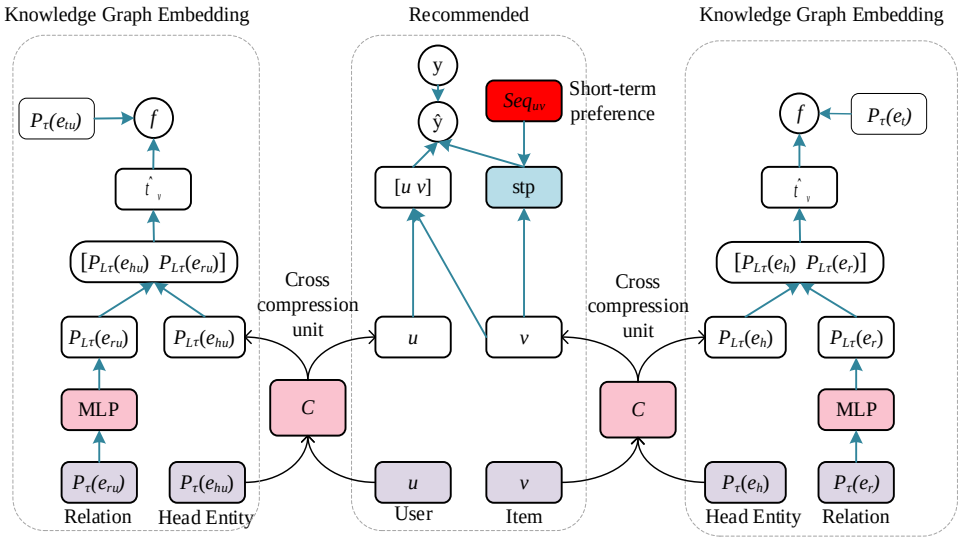


Fig. 1. LSRec model structure.

### B. User Information Embedding

A knowledge graph is a vast network that encompasses a multitude of entities and their intricate relationships. By conducting a comprehensive analysis of these relationships, one can uncover a user's latent interests. For example, when a user watches "Avengers: Endgame" their interest may not be confined to the movie itself but could also stem from an admiration for the lead actor, Robert Downey Jr., a preference for the director, Anthony Russo, or a general affinity for the science fiction genre. A knowledge graph has the capability to unveil these underlying connections, revealing other works that feature Robert Downey Jr., additional films directed by Anthony Russo, or more exceptional science fiction movies. These entities, which are related to "Avengers: Endgame" have the potential to capture the user's interest.

To depict a user's hierarchical preferences with greater precision within the knowledge graph, the RippleNet model can be employed to capture the multilevel set of relevant entities for user $u$, as illustrated in Eq. (2). By simulating the propagation of ripples (as shown in Fig. 2), the model expands outward from the user's initial points

of interest, gathering a set of relevant entities within an $x$-hop range. These entities are not only directly related to the user's interests but may also indirectly reflect their deeper preferences.

$$\varepsilon_u^x = \left\{ t \mid (h,r,t,\tau) \in G, h \in \varepsilon_u^{x-1} \right\}, x = 1,2,3,\cdots,X \quad (2)$$

When $X = 0$, the focus is solely on the items with which the user has previously interacted. These items constitute the seed set, which is denoted by $y_{uv} = 1$. In this context, $x$ represents the number of layers separating the current layer from the seed set. Furthermore, $\varepsilon_u^x$ denotes the set of quadruples within the knowledge graph that establish connections between the head entities of layer $x-1$ and the tail entities of layer $x$.

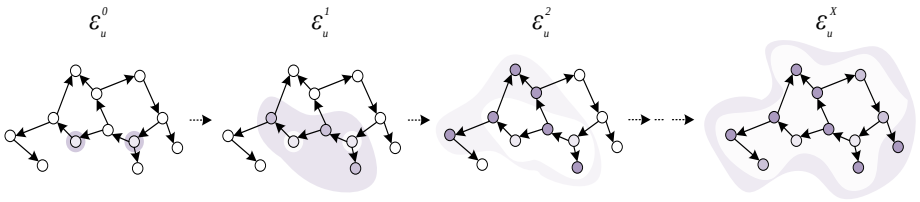


Fig. 2. Illustration of water wave propagation.

After acquiring the user level set of quadruples, the initial step involves performing knowledge graph embedding [20], where entities and relationships are embedded into a temporal space, as depicted in Fig. 3. Within this corresponding space, translation operations are subsequently executed. The detailed procedure is outlined below:

$$P_\tau(\mathbf{e}_{hu}) = \mathbf{e}_{hu} - (\omega_\tau^T \mathbf{e}_{hu})\omega_\tau \quad (3)$$

$$P_\tau(\mathbf{e}_{ru}) = \mathbf{e}_{ru} - (\omega_\tau^T \mathbf{e}_{ru})\omega_\tau \quad (4)$$

$$P_\tau(\mathbf{e}_{tu}) = \mathbf{e}_{tu} - (\omega_\tau^T \mathbf{e}_{tu})\omega_\tau \quad (5)$$

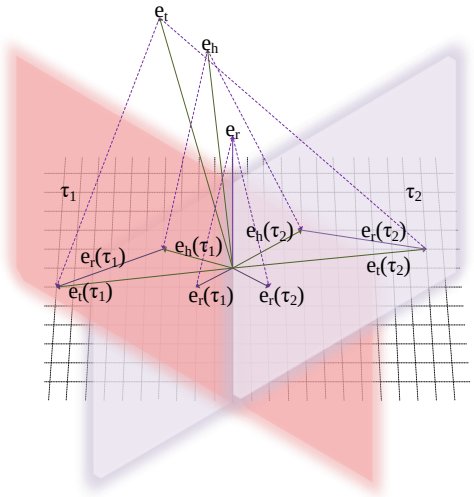


Fig. 3. Internal knowledge graph embedding model.

Here, the vectors $e_{hu}$, $e_{ru}$, and $e_{tu}$ correspond to the triples that are valid at time $\tau$. The constraint $\|\omega_\tau\|^2 = 1$ is applied to ensure that the mapping of the triples valid at time $\tau$ satisfies the condition: $P_\tau(e_{hu}) + P_\tau(e_{ru}) \approx P_\tau(e_{tu})$.

After mapping $P_\tau(e_{hu})$, $P_\tau(e_{tu})$, $P_\tau(e_{ru})$, their latent features are concatenated together. Subsequently, a K-layer Multi-Layer Perceptron (MLP) is employed to predict the tail entity $t$. The formula for this process is as follows:

$$P_{L\tau}(\mathbf{e}_{hu}) = E_{V \sim S(h_u)}[C^L(v, P_\tau(\mathbf{e}_{hu}))[\mathbf{e}]] \quad (6)$$

$$P_{l\tau}(\mathbf{e}_{ru}) = M^L(P_\tau(\mathbf{e}_{ru})) \quad (7)$$

$$\hat{\mathbf{t}} = M^K\left(\begin{pmatrix} P_{L\tau}(\mathbf{e}_{hu}) \\ P_{L\tau}(\mathbf{e}_{ru}) \end{pmatrix}\right) \quad (8)$$

Here, $S(h_u)$ represents the set of items associated with the entity hu, while C denotes the cross compression unit. Additionally, $\hat{\mathbf{t}}$ is the predicted vector for the tail entity $t$.

Finally, the knowledge graph score is computed utilizing the score similarity function $f_{KG}$. The formula for this computation is as follows:

$$f_{KG}(P_\tau(\mathbf{e}_{tu}), \hat{\mathbf{t}}) = \sigma(P_\tau(\mathbf{e}_{tu})^T \hat{\mathbf{t}}) \quad (9)$$

### C. Short-Term Preference Learning

A user's short-term interests are heavily influenced by time and environmental factors. If a recommendation system persists in suggesting the same items, it may deteriorate the user's experience. Therefore, accurately capturing a user's short-term interests and refining their interest features is of utmost importance. This section delves into the methodology of extracting short-term interests from a user's recent interaction data to gain a better understanding of their evolving preferences. A bidirectional Gated Recurrent Unit (GRU) is employed as the core component in this approach, complemented by an attention mechanism to augment its learning capability. The sequence of the most recent k items that the user has interacted with, denoted as $v_{seq}^k$, serves as the input to the network.

The GRU, with its distinctive gating mechanism, adeptly tackles challenges such as long-term memory and gradient vanishing. This enables it to make precise predictions at the current time step based on both input and memory information. When compared to the Long Short-Term Memory (LSTM) network, the GRU exhibits greater sensitivity to sequential data while incurring lower training costs. Consequently, it is more suitable for extracting

short-term interests from a user's recent interaction sequences.

Furthermore, the incorporation of an attention mechanism facilitates the adjustment of weights within the sequence, thereby enabling a more accurate capture of changes in the user's short-term interests. By balancing the significance of different items within the sequence, the attention mechanism enhances the model's capacity to detect subtle shifts in user preferences over time.

The GRU comprises two key gating mechanisms: the reset gate and the update gate. The reset gate governs how new input information is integrated with the previous memory, while the update gate determines the extent to which the previous memory is retained at the current time step. Fig. 4 depicts the operating principle of the GRU. At time step $\tau$, $r_\tau$, and $z_\tau$ represent the values of the reset gate and the update gate, respectively. $x\tau$ is the input vector to the GRU module at time $\tau$, and $y_\tau$ is the candidate output vector at that time. $h\tau-1$ denotes the output from the previous time step $\tau-1$, and $h_\tau$ is the output of the GRU unit at the current time step $\tau$. The short-term representation of the user can be computed as follows:

$$r_\tau = sigmoid(W_r \mathbf{x}_\tau + U_r \mathbf{h}_{\tau-1}) \tag{10}$$

$$z_\tau = sigmoid(W_z \mathbf{x}_\tau + U_z \mathbf{h}_{\tau-1}) \tag{11}$$

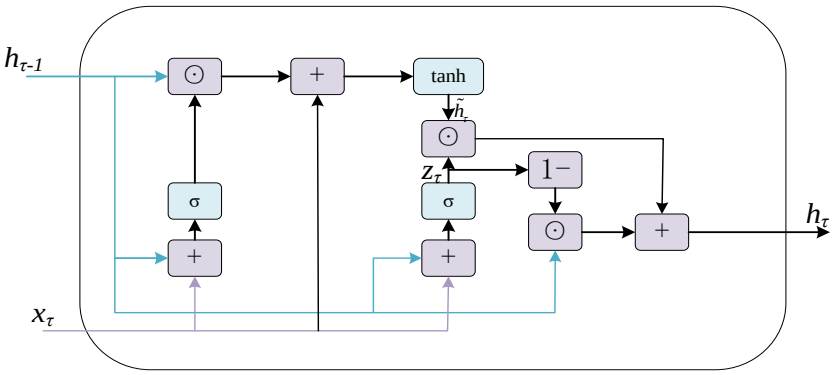


Fig. 4. Structure of the GRU module unit.

Here, the matrices W and U are parameters that govern the impact of the input and the prior state on the current time step, respectively.

Once the gating values have been determined, the data is initially reset utilizing the reset gate. Subsequently, the reset data is concatenated with the user input vector at the present time step. The candidate hidden state is then computed using the tanh activation function, ensuring that its values stay within the bounds of 1 and 1. This results in a candidate output vector that documents the state information at the current time step.

Specifically, the reset gate serves to filter the data, and the filtered data is merged with the current user input. This merged outcome is then processed through the tanh function, producing a candidate hidden state that not only compresses the data within the range of 1 to 1 but also effectively captures the crucial information at the current time point, thereby supplying vital context for the network. This procedure is expressed by the following formula:

$$\tilde{h}_\tau = tanh(W_h x_\tau + U_h(h_{\tau-1} \odot x_\tau)) \tag{12}$$

Here, the Hadamard product ( $\odot$ ) is employed for element-wise multiplication.

At the conclusion of the GRU operation, the mechanisms of the reset gate and the update gate are integrated to facilitate selective filtering and incorporation of user information. To elaborate, the Hadamard product of $(1 - \boldsymbol{z}_\tau)$ and $\boldsymbol{h}_{\tau-1}$ is utilized to selectively filter out past user information, whereas the Hadamard product of $\boldsymbol{z}_\tau$ and $\tilde{\boldsymbol{h}}_\tau$ is used to selectively integrate current information. Consequently, the final hidden state $\boldsymbol{h}_\tau$ consolidates the essential information from the current input and the significant features inherited from the past. This sequence of operations can be summarized by the following formula:

$$h_\tau = (1 - z_\tau) \odot h_{\tau-1} + z_\tau \odot \tilde{h}_\tau \tag{13}$$

To better capture the dynamic short-term preferences of users with precision, an attention mechanism is subsequently utilized. The calculation for this mechanism is as follows:

$$\varphi = softmax(\omega^T \tanh(H_{out})) \tag{14}$$

$$r = H_{out} \varphi^T \tag{15}$$

$$sp = MLP(r) \tag{16}$$

In this workflow, Hout signifies the output of the RNN's hidden layer, $\varphi^T$ denotes a parameter matrix that corresponds to the number of GRU units, and r represents the weighted output derived from the RNN. Subsequently, after undergoing processing through a Multi-Layer Perceptron (MLP), the weighted short-term preferences of the user are obtained.

### *D. Long-Term and Short-Term Preference Fusion Recommendation*

In the construction of the user preference model, the Knowledge Graph and the GRU are employed to capture long-term and short-term user preferences, respectively. The techniques for item embedding and cross compression unit implementation adhere to the effective strategies outlined in Section III.C.

Within the recommendation module, the user feature vector u, the item feature vector v, and the user's recent interaction sequence $v_{seq}^{k}$ serve as inputs. Utilizing these feature vectors, the cross compression unit is initially leveraged to facilitate interaction and fusion between the user and item features. Subsequently, an MLP is utilized for further processing to generate the ultimate recommendation prediction. The detailed procedure is as follows:

$$u_C = E_{e \sim S(u)}[C^L(u,e)[u]] \tag{17}$$

$$u_L = M(M(...M(u_C))) = M^L(u_C) \tag{18}$$

$$v_C = E_{e \sim S(v)}[C^L(v,e)[v]] \tag{19}$$

$$v_L = M(M(...M(v_C))) = M^L(v_C) \tag{20}$$

When users make a choice, their decisions are influenced by both long-term and short-term preferences. However, merely combining these preferences may not fully harness their potential interactions and might not constitute the most effective fusion strategy. To tackle this challenge, a parameter u is introduced to modulate the weights of long-term and short-term preferences during the fusion process. This enables a more adaptable capture of the inherent relationship between long-term and short-term interests.

The integrated recommendation representation, combining both long-term and short-term preferences, can be formulated as follows:

$$\hat{y}_{uv} = \mu \cdot u_L \cdot v_L^T + (1-\mu) \cdot sp \cdot v_L^T \tag{21}$$

## IV. Experiments

To validate the efficacy of the proposed LSRec model, experiments were carried out within the PyTorch framework, assessing its performance on three distinct datasets.

### A. Datasets

The experiments utilized datasets gathered from three renowned online communities [21]. Detailed descriptive statistics for each dataset are provided in Table I.

Douban: This is a widely-used community website where users can post reviews of movies, music, and books. The dataset was compiled by crawling users' identity information, review content, and timestamps from the movie community, encompassing users' social network information as well. Reviews serve as indicators of users' movie consumption.

TABLE I. Statistics of Three Data Sets

| Data Sets | Douban | Delicious | Yelp |
|---|---|---|---|
| Users | 32314 | 1650 | 141804 |
| Items | 14109 | 4282 | 17625 |
| Events | 3493821 | 296705 | 1200503 |
| Timestamp | 298 | 70 | 2068 |
| Start Date | 01/12/2008 | 08/12/2009 | 01/01/2009 |
| End Date | 07/22/2016 | 07/01/2016 | 10/15/2010 |

Delicious: An online bookmarking system that allows users to save, share, and explore web bookmarks, while assigning various semantic tags to them. The dataset comprises the sequence of tags assigned by users to bookmarks, with timestamps for each tagging action.

Yelp: An online review platform enabling users to review local businesses, including restaurants and shops. Akin to Douban, each review in the dataset is regarded as an observation.

### B. Benchmark Model

To assess the performance of the LSRec model, it was benchmarked against the following baseline models:

RippleNet [17]: This is the pioneering knowledge graph-based recommendation model that integrates feature learning with path-based methodologies.

CKE [18]: A model that amalgamates textual, structural, and visual knowledge with collaborative filtering techniques to facilitate embedded recommendations.

MKR [19]: This model introduces a multi-task learning framework, leveraging knowledge graphs to enhance recommendation tasks.

DGRec [22]: A recommendation approach grounded in dynamic graph attention networks, utilizing recurrent neural networks to comprehensively model users' dynamic behaviors.

AEMC: A deep learning-based algorithm that utilizes deep autoencoders to capture non-linear, hidden relationships between multi-criteria ratings, enhancing recommendation accuracy by learning fine-grained user preferences.

KGRec [23]: A recommendation model based on knowledge graph embeddings, utilizing BERT-like attention mechanisms to model user-item interactions and evolving preferences over time. It integrates static knowledge from the graph and dynamic user behavior to provide more accurate, context-aware recommendations.

### C. Evaluation Protocol and Parameter Settings

For the experimental evaluation, recommendation effectiveness is evaluated using Recall [24], Precision, ROC-AUC and Normalized Discounted Cumulative Gain (NDCG@k) [25] were chosen as the evaluation protocol, where *k* represents the number of items in the recommendation list.

Among them, Recall measures the ability of the model to retrieve relevant items from the user's actual interest list; Precision evaluates the accuracy of the recommendations by calculating the proportion of relevant items in the predicted recommendation list; ROC-AUC quantifies the overall ability of the model to distinguish between relevant and irrelevant items. The ROC curve illustrates the relationship between true and false positive rates at different classification thresholds. ROC-AUC is the area under the ROC curve and ranges from 0.5 to 1. The calculation formulas for metrics are as follows:

$$Recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{22}$$

$$Preccision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|} \quad (23)$$

Here, for user u, let $T(u)$ be the list of real interactions, and $R(u)$ be the list of recommendations finally output by the model.

NDCG is a crucial metric for assessing the ranking quality of the recommendation list. It considers not only the sequence of recommended items within the list but also the degree of relevance of these items to the items actually accessed by the user. The computation method is outlined as follows:

$$\mathrm{NDCG@k} = \frac{\mathrm{DCG@k}}{\mathrm{IDCG@k}} \quad (24)$$

In this context, DCG (Discounted Cumulative Gain) computes a ranking score for the generated recommendation list, while IDCG (Ideal Discounted Cumulative Gain) represents the DCG score of the optimal list that the user would ideally interact with, serving as a benchmark. The relevance score, denoted as relv, quantifies the degree of relevance between the recommended items and the user's preferences. These metrics are used collectively to evaluate the performance of the recommendation system.

The LSRec model was implemented within the PyTorch framework. The dimensionality of both the item and user representation vectors was configured to 8, and the length of the sequence of recent interactions was set to 20. The GRU incorporated 16 hidden layers, with a learning rate of 2e−4 for the recommendation model and 2e−5 for the TKGE module. To optimize the model, the Adam optimizer was employed. Fig. 5 shows the trend of loss with epoch during training.

### D. Experimental Results and Analysis

In the LSRec model, the parameter $\mu$ is pivotal for striking a balance between the weights of long-term and short-term preferences in the recommendation outcomes. The magnitude of μ directly dictates the degree to which short-term preferences sway the ultimate recommendation results. To elaborate, a higher $\mu$ value amplifies the influence of long-term preferences, whereas a lower $\mu$ value accentuates the impact of short-term preferences. Fig. 5(b) depict the LSRec model's performance on the Delicious dataset, with recommendation lengths set at 1, 5, and 10, and how its performance, in terms of NDCG metrics, varies with changes in the $\mu$ value. From an examination of Fig. 5, it is evident that the LSRec model attains optimal performance on NDCG metrics when μ is set to 0.6.

Furthermore, the parameter $d$ also holds substantial significance in the model, as it governs the complexity of the user and item representation vectors. Specifically, as the value of d increases, the representation vectors of users and items become more elaborate and refined. To delve into the impact of $d$ on model performance, we have provided Fig. 5(a), which distinctly illustrates the fluctuations in model performance on NDCG metrics across different values of $d$. When observing Fig. 6, it is evident that the LSRec model attains its peak performance on NDCG metrics when the parameter $d$ is set to 8.

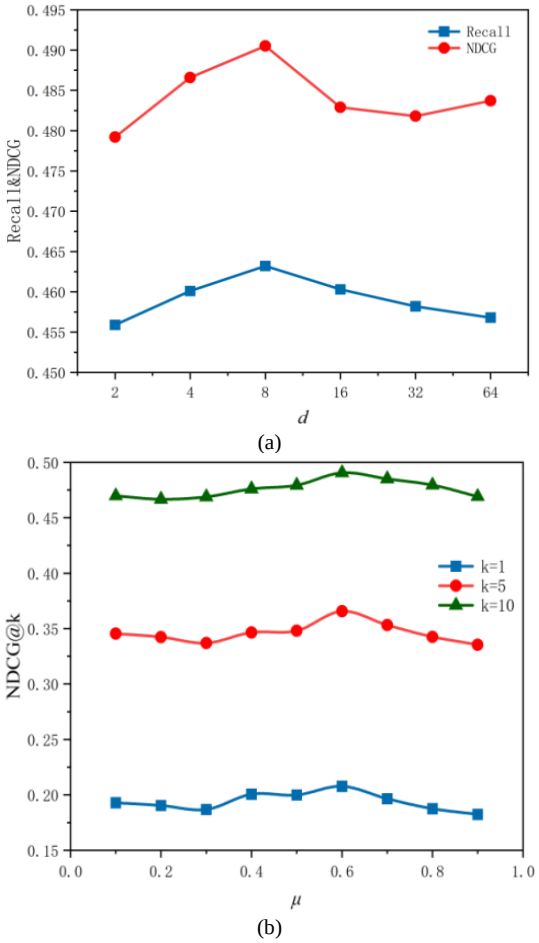


Fig. 5. Performance variation with (a) Parameter $d$; (b) Parameter $\mu$.

After conducting a series of experiments, the optimal parameter configuration for the LSRec model was identified as $\mu$ = 0.6, $k$ = 10, and $d$ = 8. Experiments were performed by integrating the three major modules of the proposed model LSRec to finally generate a recommendation list for the target user. The performance of the baselines was compared with that of the proposed model on three open-source datasets to test the effectiveness of the model. The experimental results are demonstrated in Tables II and III.

TABLE II. COMPARISON RESULTS BETWEEN LSREC AND BENCHMARK MODELS

| Models | Douban | | | Delicious | | | Yelp | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Rec** | **Pre** | **AUC** | **Rec** | **Pre** | **AUC** | **Rec** | **Pre** | **AUC** |
| CKE | 0.344 | 0.323 | 0.612 | 0.348 | 0.397 | 0.546 | 0.345 | 0.389 | 0.521 |
| RippleNet | 0.389 | 0.398 | 0.623 | 0.343 | 0.421 | 0.591 | 0.367 | 0.428 | 0.548 |
| MKR | 0.395 | 0.403 | 0.632 | 0.394 | 0.445 | 0.644 | 0.426 | 0.447 | 0.558 |
| $DGRec_{self}$ | 0.416 | 0.453 | 0.646 | 0.425 | 0.454 | 0.651 | 0.435 | 0.463 | 0.613 |
| TKGR | 0.434 | 0.466 | 0.659 | 0.445 | 0.476 | 0.664 | 0.452 | 0.479 | 0.639 |
| AEMC | 0.440 | 0.470 | 0.669 | 0.450 | 0.478 | 0.673 | 0.476 | 0.485 | 0.649 |
| KGRec | 0.489 | 0.478 | 0.674 | 0.457 | 0.487 | 0.689 | 0.484 | 0.496 | 0.659 |
| LSRec | **0.493** | **0.482** | **0.682** | **0.473** | **0.527** | **0.698** | **0.498** | **0.521** | **0.669** |

TABLE III. NDCG@10 COMPARISON RESULTS BETWEEN LSREC AND BENCHMARK MODELS

| Models | Douban | Delicious | Yelp |
|---|---|---|---|
| | NDCG@10 | NDCG@10 | NDCG@10 |
| CKE | 0.0782 | 0.3178 | 0.1240 |
| RippleNet | 0.0860 | 0.3243 | 0.1278 |
| MKR | 0.0893 | 0.3394 | 0.1364 |
| $DGRec_{self}$ | 0.0924 | 0.3598 | 0.1406 |
| TKGR | 0.1110 | 0.4581 | 0.1427 |
| AEMC | 0.1124 | 0.4690 | 0.1489 |
| KGRec | 0.1149 | 0.4821 | 0.1512 |
| LSRec | **0.1167** | **0.4905** | **0.1563** |

The experimental outcomes reveal that the LSRec model notably surpasses other baseline models on both the Delicious and Yelp datasets in terms of the key metrics, Recall, precision and AUC. Specifically, on the Delicious dataset, the LSRec model demonstrates a 16.7% and 3.4% improvement in Recall, along with a 15.6% and 7.6% boost in precision, as well as a 7.7% and 1.2% improvement in AUC, when compared to the MKR and KGRec baseline models, respectively. Similarly, on the Yelp dataset, the LSRec model exhibits a 14.4% and 2.8% increase in Recall, as well as a 14.2% and 4.7% boost in precision, along with a 16.6% and 1.5% boost in AUC, in comparison to the MKR and KGRec baseline models, respectively.

These enhancements can be primarily attributed to the LSRec model's thorough exploration of the relationship between users and the knowledge graph, as well as its proficient capture of users' short-term interests. However, on the Douban dataset, the LSRec model shows more modest improvements in both Recall, precision and AUC and NDCG metrics. On the Douban dataset, the LSRec model exhibits a 23.5% and 1.5% increase in NDCG in comparison to the MKR and KGRec baseline models, respectively. This may be associated with the unique characteristics of the Douban dataset, where users typically consume an item only once. This consumption pattern could pose challenges for the model in enhancing its recommendation performance. Despite this, the LSRec model still outperforms other baseline models on the Douban dataset, further attesting to its superiority and generalization capabilities.

In conclusion, by thoroughly considering the relationships between users and the knowledge graph, as well as users' short-term interests, the LSRec model achieves remarkable improvements in recommendation performance across multiple datasets.

The core strengths of the LSRec method reside in its comprehensiveness and multi-faceted considerations. Firstly, this approach introduces temporal knowledge graphs, which not only enhance the comprehension of potential relationships among items but also facilitate a deeper exploration of user interests. By leveraging multi-task learning, LSRec efficiently integrates the training process of the knowledge graph with the recommendation task, thereby optimizing recommendation performance. The convergence during training is shown in Fig. 6.

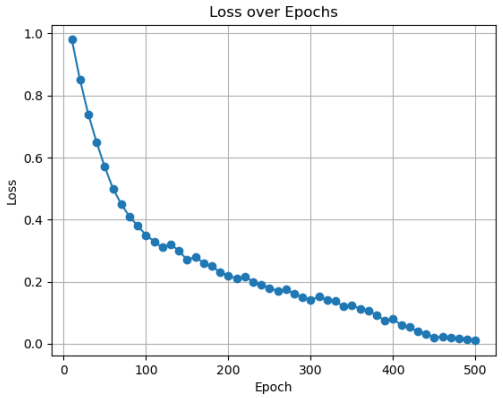


Fig. 6. Loss convergence curve.

Secondly, the LSRec algorithm employs the RippleNet algorithm 17 to extract user features, adeptly capturing the associations between users and knowledge graph entities. These connections serve as invaluable supplementary information, markedly improving the accuracy of recommendations. Furthermore, LSRec delves into user preferences from the user's perspective, resulting in more personalized recommendations.

Lastly, LSRec also meticulously accounts for users' short-term preferences during the recommendation process. By designing and implementing a weighting mechanism, LSRec can precisely evaluate the significance of short-term preferences in recommendations, resulting in more rational and precise suggestions. This comprehensive advantage distinguishes the LSRec algorithm within the realm of recommendation systems, offering users a more considerate and efficient recommendation experience.

### *E. Ablation Study*

This section aims to explore the impact of the bidirectional GRU network and RippleNet on the performance of recommendation models. The experiment validates the importance of each module in the model by comparing the performance before and after removing the modules.

The LSRec model consists of two core modules: the short-term preference capturing module and the long-term preference capturing module. The model primarily utilizes the bidirectional GRU network and attention mechanism to capture users' interest fluctuations, and employs the RippleNet framework to uncover the relationships between user preferences and knowledge graph entities.

The experiment separately demonstrates the model performance with only the bidirectional GRU network and the combination of RippleNet. The experimental results of using GRU alone and integrating the RippleNet module across three datasets are presented in Table IV. A comparison of the experimental results indicates that the removal of any module leads to a decline in recommendation performance. This suggests that both the short-term and long-term preference capturing modules play a crucial role in enhancing the accuracy of the recommendation model. When only the bidirectional GRU network is used, the model performs relatively well, as this

module effectively filters the recommendation range by focusing on recent user behavior changes, providing relatively accurate recommendations. However, when the RippleNet module is incorporated, evaluation metrics on all three datasets show significant improvement. This indicates that by capturing long-term preference changes, the model can better understand the user's actual needs, offering more precise recommendations.

Through a comprehensive analysis of the combined effects of both modules, the experimental results show that the organic integration of the two modules leads to optimal recommendation performance. By considering both contextual constraints and user preference changes, the recommendation system can more comprehensively and accurately reflect shifts in user interests, thereby providing users with more forward-looking and precise recommendation services.

TABLE IV. RESULT OF THE ABLATION EXPERIMENT

| Model | Datasets | Recall@10 | NDCG@10 |
|---|---|---|---|
| GRU | Douban | 0.1537 | 0.1083 |
| | Delicious | 0.4382 | 0.4980 |
| | Yelp | 0.0832 | 0.1329 |
| LSRec | Douban | 0.1682 | 0.1167 |
| | Delicious | 0.4632 | 0.4905 |
| | Yelp | 0.0894 | 0.1563 |

## VI. CONCLUSION

In this study, we propose a knowledge graph recommendation method, termed LSRec, which is grounded on both long-term and short-term user preferences. This approach leverages the RippleNet framework to thoroughly investigate the interconnections between users and knowledge graph entities. Additionally, it incorporates recurrent neural networks and attention mechanisms to adeptly capture user preferences. To further bolster recommendation accuracy, LSRec introduces a parameter that harmonizes users' long-term and short-term preferences. Through parameter tuning, LSRec significantly improves recommendation performance over baseline models by considering both user-entity relationships and short-term interests. Future research could explore multimodal fusion, context-aware recommendations, improved interpretability, cross-domain applications, realtime systems, user feedback integration, and privacy safeguards. These efforts aim to boost generalization, adaptability, efficiency, and user trust, paving the way for the continued evolution of LSRec and its derivatives, ultimately providing users with richer, more accurate, and secure recommendation experiences.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

Guanfeng Li conducted the research; Guanfeng Li and Feizhou Qin analyzed the data; Yuyin Chen and Yunli Wang wrote the paper. All authors had approved the final version.

## FUNDING

This work was supported in part by the Key Research and Development Program of Ningxia under Grant 2023BSB03066; and in part by the National Science Foundation of China under Grant 62066038; and in part by the Natural Science Foundation of Ningxia under Grant 2024AAC03098.

## REFERENCES

[1] M. Li, W. Zheng, Y. Xiao, K. Zhu, and W. Huang, "Exploring temporal and spatial features for next POI recommendation in LBSNs," *IEEE Access*, vol. 9, pp. 35997–36007, 2021.

[2] P. Duraisamy, S. Yuvaraj, Y. Natarajan, and V. Niranjani, "An overview of different types of recommendations systems—A survey," in *Proc. 2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT)*, IEEE, 2023, pp. 1–5.

[3] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4425–4445, 2022.

[4] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo, "A review of content-based and context-based recommendation systems," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, no. 3, pp. 274–306, 2021.

[5] B. Walek,and V. Fojtik, "A hybrid recommender system for recommending relevant movies using an expert system," *Expert Systems with Applications*, vol. 158, 113452, 2020.

[6] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T. S. Chua, "Contrastive learning for cold-start recommendation," in *Proc. the 29th ACM International Conference on Multimedia*, 2021, pp. 5382–5390.

[7] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T. S. Chua, "Contrastive learning for cold-start recommendation," in *Proc. the 29th ACM International Conference on Multimedia*, 2021, pp. 5382–5390.

[8] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proc. the 10th ACM Conference on Recommender Systems*, 2016, pp. 191–198.

[9] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, H. Shah *et al*., "Wide & deep learning for recommender systems," in *Proc. the 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 7–10.

[10] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, K. Gai *et al*., "Deep interest network for click-through rate prediction," in *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1059–1068.

[11] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in *Proc. the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 165–174.

[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 639–648.

[13] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *Proc. the ACM Web Conference 2022*, 2022, pp. 2320–2329.

[14] H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: Deep knowledge-aware network for news recommendation," in *Proc. the 2018 World Wide Web Conference*, 2018, pp. 1835–1844.

[15] Y. Cao, X. Wang, X. He, Z. Hu, and T. S. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *Proc. the World Wide Web Conference*, 2019, pp. 151–161.

[16] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

[17] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proc. the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 417–426.
[18] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 353–362.
[19] H. Wang, F Zhang, M Zhao, W Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *Proc. the World Wide Web Conference*, 2019, pp. 2000–2010.
[20] S. S. Dasgupta, S. N. Ray, and P. Talukdar, "Hyte: Hyperplane-based temporally aware knowledge graph embedding," in Proc. the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2001–2011.
[21] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 555–563.
[22] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 555–563.
[23] Q. Shambour, "A deep learning based algorithm for multi-criteria recommender systems," *Knowledge-Based Systems*, vol. 211, 106545, 2021.
[24] C. Ma, L. Ma, Y. Zhang, R. Tang, X. Liu, and M. Coates, "Probabilistic metric learning with adaptive margin for top-k recommendation," in *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1036–1044.
[25] R. Jagerman, Z. Qin, X. Wang, M. Bendersky, and M. Najork, "On optimizing top-k metrics for neural ranking models," in *Proc. the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 2303–2307.