Paper Report

Tuo Liu

August 26, 2022

Contents

Chapter 1

Explainable Interaction-driven User Modeling over Knowledge Graph for Sequential Recommendation

1.1 Main Contribution

1. The authors propose a novel explainable user modeling algorithm that captures the interaction-level user dynamic preference. It is a high-level representation which contains auxiliary semantic information.
2. The authors endow the recommendation system the ability of pathwise explanation by modeling the semantic explicit paths between user-item pairs instead of implicit item embeddings.
3. The authors adopt a joint learning manner for better representation learning by employing multi-modal fusion which benefits from the structural constraints in KG, where three kinds of modalities are involved in.

1.2 Model

1.2.1 Framework

The framework EIUM consists of three main components:

1. Multi-modal Fusion
2. Interaction Representation
3. Sequential Interactions Modeling

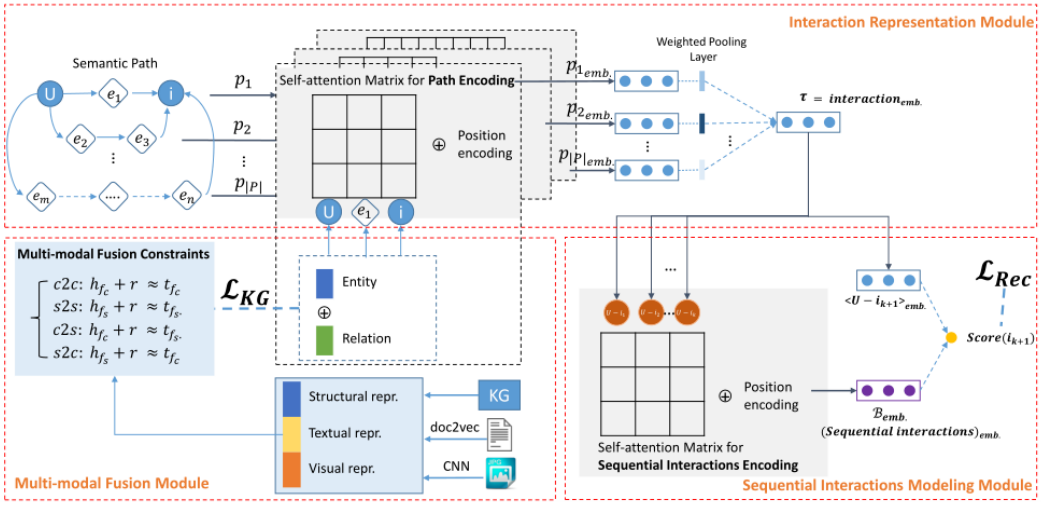

Figure 1.1: Structure

1.2.2 Multi-modal Fusion

To incorporate more auxiliary information into our model, the authors adopt the content and structural features to assist in item representation.

Content-features

Content-features include textual features and visual features. For textual features, we collect the item's title and description, and then fastText is applied to extract the text representations.

For visual features, the authors collect the visual descriptions of items such as movie's posters from IMDB website 2 through corresponding movie-id. The authors employ AlexNet pre-trained on ImageNet to extract 4096-dim visual semantic features from those posters. Due to the possible redundancy and noise of visual information, we reduce the high dimensional deep feature to 300-d with PCA to obtain the final visual feature.

Therefore, the content feature can be represented as:

$$f_c = \sigma\left(W_f\left[\text{ item }^{\text{textual}},\ \text{item }^{\text{visual}}\right] + b_f\right)$$

Structural features

The authors construct the structural representation of items by introducing external KG which contains abundant objective knowledge. The simple KGE method is used to embed entities and relationships into continuous vector space while preserving their structure information. In EIUM, the authors learn the structural feature of entities and relations through the structural constraints of KG.

multi-modal fusion constraints

The multi-modal fusion is implemented by incorporating textual, visual, structural knowledge into network where the different modal features satisfy the constraints of the structure information of entities and relations.

$$\begin{array}{ll} c2c : h_{f_c} + r \approx t_{f_c} & s2s : h_{f_s} + r \approx t_{f_s} \\ c2s : h_{f_c} + r \approx t_{f_s} & s2c : h_{f_s} + r \approx t_{f_c} \end{array}$$

1.2.3 Interaction Representation

The authors adopt self-attention mechanism with position encoding module to model the path.

The input of self-attention layer is a sequence in which each element consists of two raw feature vectors e and r that describe entity and relation, respectively. The authors concatenate the embedding of entity and relation, and then embed it in a latent space by a fully-connected layer. The output is regarded as the element x_i:

$$x_i = \sigma\left(W_x\left[e_i, r_i\right] + b_x\right)$$

Take the path sequence $x = x_1, x_2, ..., x_L$ as input to the masked self-attention model, $f(x_i, x_j)$ denotes the correlations between the two elements:

$$f\left(x_i, x_j\right) = W^T \sigma\left(W_1 x_i + W_2 x_j\right) + \gamma M_{i,j}$$

Then, the attention score between x_i and x_j is defined as:

$$a_{ij} = \frac{e^{[f(x_i, x_j)]}}{\sum_{j=1}^{|L|} e^{[f(x_i, x_j)]}}$$

After obtaining attention scores over all entities, the output for x_j is defined as:

$$o_j = \sum_{i=1}^{L} a_{ij} \odot x_i$$

To obtain the unified embedding of a single path, we perform the mean-pooling operation on the output path sequence:

$$pl_{emb.} = \text{ mean-pooling } (o_j)_{j=1}^{L} = \frac{1}{L}\sum_{j=1}^{L} o_j$$

Since different paths are generated by the combination of different relations with their corresponding intermediate entities, each path contains different semantic information and plays a different role in modeling user-item interaction. Hence, we use a weighted pooling layer to help distinguish the path importance. Attention mechanism is adopted to address this issue. We use the user u and item i as the query to learn the weighted score :

$$\begin{aligned} \text{query } &= \sigma_q\left(W_q[u,i] + b_q\right) \\ w(\mathcal{P}(u,i)) &= \left[w_1, w_2, \ldots, w_{|\mathcal{P}|}\right] \end{aligned}$$

The unified interaction representation is obtained by aggregating the weighted paths:

$$\tau = \text{ ineraction }_{emb\cdot} = \sum_{l=1}^{|\mathcal{P}|} w_l \cdot p_{l_{emb}}$$

1.2.4 Sequential Interactions Modeling

A user's historical records are a sequence in chronological order, thus her/his subsequent item can be predicted by SR methods. Since the marked self-attention network is able to capture and characterize the temporal dependency in sequence effectively and efficiently, we still adopt the architecture to model sequential interactions.

$$\begin{aligned} f\left(\tau_i, \tau_j\right) &= W_\tau^T \sigma\left(W_\tau^1 \tau_i + W_\tau^2 \tau_j\right) + \gamma M_{i,j} \\ a_{\tau_i,\tau_j} &= \frac{e^{[f(\tau_i,\tau_j)]}}{\sum_{j=1}^{T} e^{[f(\tau_i,\tau_j)]}} \\ e_{\tau_j} &= \sum_{\tau=1}^{T} a_{\tau_i,\tau_j} \odot \tau_i \\ \mathcal{B}_{emb.} &= \sum_{t=1}^{T} w_t \cdot e_{\tau_t} \end{aligned}$$

The output embedding $B_{emb.}$ denotes user dynamic preference which can be used to predict probability of user u engaging item v through a predicting function f, which can be inner product or H-layer MLP:

$$p_{u,v} = \sigma\left(f\left(\mathcal{B}_{emb\cdot}, \tau_v\right)\right)$$

1.2.5 Joint Learning of Sequential Recommendation and Multi-modal Fusion

The ultimate goal of the task is to rank the ground-truth item higher than all other items, which is defined as follows through cross-entropy function:

$$\mathcal{L}_{rec} = \sum_{u}\sum_{v}\sum_{v' \in \hat{\mathcal{V}}} -\log\sigma\left[D(uv) - D\left(uv'\right)\right]$$

In addition, we also consider the structural information based on KG to facilitate sequential recommendation. As described above, since the items in KG satisfy the multi-modal fusion constraints. The four losses from KG can be included into our model:

$$\begin{aligned} \mathcal{L}_{kg} &= \mathcal{L}_{c2c} + \mathcal{L}_{s2s} + \mathcal{L}_{c2s} + \mathcal{L}_{s2c} \\ &= \frac{1}{4}\sum_{i} \|h + r - t\|, i \in \{c2c, s2s, c2s, s2c\} \end{aligned}$$

Therefore, we train our proposed EIUM model by minimizing the overall objective function that jointly evaluates the performance:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{kg}$$

1.3 Experiments

The paper experiments with a benchmark dataset: **MovieLens-20M**. The baselines include **BPR, Bi-LSTM, Bi-LSTM with attention, ATRank, CKE, KTUP**. The results are as follows:

Datasets	Methods	Evaluation Metrics					
		AUC	MAP	Hit@5	Hit@10	NDCG@5	NDCG@10
MovieLens-20M + Freebase	BPR	0.9065 (-0.18%)	0.3312 (-21.05%)	0.4920 (-19.65%)	0.6653 (-16.68%)	0.3578 (-21.95%)	0.3990 (-21.19%)
	Bi-LSTM	0.8800 (-3.09%)	0.3278 (-21.86%)	0.5583 (-8.82%)	0.7180 (-10.08%)	0.3749 (-18.22%)	0.4147 (-18.09%)
	Bi-LSTM+att.	0.8897 (-2.03%)	0.3606 (-14.04%)	0.5944* (-2.92%)	0.7409* (-7.21%)	0.4095 (-10.67%)	0.4460 (-11.91%)
	ATRank	0.8724 (-3.93%)	0.3454 (-17.66%)	0.5561 (-9.18%)	0.7057 (-11.62%)	0.3877 (-15.42%)	0.4250 (-16.06%)
	CKE	0.9054 (-0.30%)	0.3869* (-7.77%)	0.5790 (-5.44%)	0.7175 (-10.14%)	0.4261* (-7.05%)	0.4571* (-9.72%)
	KTUP	**0.9187*** (+1.17%)	0.3829 (-8.72%)	0.5662 (-7.53%)	0.7085 (-11.27%)	0.4196 (-8.46%)	0.4516 (-10.80%)
	EIUM	0.9081	**0.4195**	**0.6123**	**0.7985**	**0.4584**	**0.5063**

Figure 1.2: Performance

1. BPR gets the worst performance because it is a pair-wise ranking method with no additional auxiliary information, which results in failure of capturing user's accurate preferences.
2. As for SR baseline methods which refer to ATRank, LSTM, Bi-LSTM+attention, Bi-LSTM with attention approach performs the best because the bi-directional structure uses the information from the past and the future.
3. As for Knowledge-based baseline methods which refer to CKE, KTUP, CKE achieves better performance than KTUP, because CKE utilizes the KG embeddings as the additional information which preserve the information in different two sources to enhance the recommendation.
4. Bi-LSTM and CKE achieve comparable results, which indicates that time information is as important as external knowledge in sequence recommendation task.
5. Our proposed EIUM model performs better than other baseline methods in most cases. This is due to the introduction of the external KG information into recommendation model.

1.4 My Idea

In this paper, the authors propose a model named EIUM that utilizes KG to do the sequential recommendation. In my opinion, the most novel part of this paper is using u-i interaction to represent the history instead of only using the item. In this way, we can bring in the paths in knowledge graph and thus make the recommendation explainable. We can learn from the way how the authors utilize the KG.

Chapter 2

Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks

2.1 Main Contribution

It is the first time that sequential recommender is integrated with external memories by leveraging existing KB information. For evaluating our model, we prepare four RS datasets, and then link items of the four datasets with Freebase entities. Extensive results on the four datasets have shown the superiority of the proposed model in both effectiveness and interpretability.

2.2 Method

2.2.1 A GRU-based Sequential Recommender

The authors adopt the GRU network as the base sequential recommender in our work, since it is simpler and contains fewer parameters than LSTM. The network is as follow:

$$\boldsymbol{h}_t^u = \text{GRU}\left(\boldsymbol{h}_{t-1}^u, \boldsymbol{q}_{i_t}; \Theta\right)$$

where $\boldsymbol{h}_t^u$ is called sequential preference representation of user u. To generate the sequential recommendation, the authors rank a candidate item i by computing the recommendation score according to

$$s_{u,i,t} = g(u, i, t) = \boldsymbol{h}_t^{u\top} \cdot \boldsymbol{q}_i$$

2.2.2 Augmenting Sequential Recommender with Knowledge-Enhanced Memory Networks

The authors utilize Key-Value Memory Network (KV-MN) to maintain the KB knowledge, and then integrate the KV-MNs with the base sequential recommender.

Modeling Attribute-level User Preference with Key-Value Memory Networks

The authors use KV-MNs, which has been proposed to split a memory slot into a key vector and a value vector, and then associate the key vector with the value vector in a memory slot. Such an architecture perfectly matches the structure of KB triples, which typically correspond to entity attribute information.

By storing attribute information (a.k.a., features) of items in the key vectors and attribute-specific user preference in value vectors, we are able to model long-term preference evolving in the attribute level.

Formally, the authors frame the user-specific KV-MNs as a set of vector pairs (k, v), where k is the key vector for attribute a and v is the value vector corresponding to attribute a for user u.

And then we can do two operations to maintain and monitor the evolving process of attribute-level preferences for users: read and write.

1. READ: At each time t, the sequential preference representation h_t^u from the GRU network in Eq. 1 is taken as the query to the KV-MNs, which is used to address and visit the memory of key vectors in K, and then the associated value vectors are combined using some strategy as the return.

$$\boldsymbol{m}_t^u \leftarrow \text{READ}\left(\left\{\left(\boldsymbol{k}_1, \boldsymbol{v}_1^u\right), \cdots, \left(\boldsymbol{k}_A, \boldsymbol{v}_A^u\right)\right\}, \underline{\boldsymbol{h}_t^u}\right)$$

   We call m_t^u attribute-based preference representation of user u.

2. WRITE: Once the KV-MNs receives a new interaction record between user u and item i, the Write operation is run using the entity embedding of item i as a reference vector, and then update the associated user-specific value vectors according to some strategy.

$$\left\{\boldsymbol{v}_1^u, \cdots, \boldsymbol{v}_A^u\right\}^{new} \leftarrow \text{Write}\left(\left\{\left(k_1, \boldsymbol{v}_1^u\right), \cdots, \left(\boldsymbol{k}_A \boldsymbol{v}_A^u\right)\right\}^{\text{old}}, \underline{\boldsymbol{e}_i}\right)$$

   where e_i is some embedding representation of item i.

Enhancing KV-MVs with KB Information

A key problem to be solved is how to set the key matrix K with appropriate attribute information from the item side. Here, the authors propose to use KB information for setting the key matrix, which is able to flexibly characterize attribute information of entities from various domains. To learn KB embedding, we use the simple and efficient model TransE.

To this end, we have obtained the embeddings for both the entities and relations. KB relations usually correspond to attribute information of entities. Hence, we fill the key matrix by the relation embeddings.

Instantiating the Read and Write Operations

1. For the Read operation, at time t, we use the following attentive combination for user u

$$m_t^u \leftarrow \sum_{a=1}^{A} w_{t,u,a} \cdot v_a^u$$

   where $w_{t,u,a}$ is the attention weight of the attribute a for user u at time t defined as

$$w_{t,u,a} = \frac{\exp\left(\gamma \tilde{\boldsymbol{h}}_t^u \cdot \boldsymbol{k}_a\right)}{\sum_{a'=1}^{A} \exp\left(\gamma \tilde{\boldsymbol{h}}_t^u \cdot \boldsymbol{k}_{a'}\right)}$$

2. For the Write operation, the case becomes a bit complicated. The update from item i relative to attribute a is computed as

$$\boldsymbol{e}_a^i = \boldsymbol{e}_i + \boldsymbol{r}_a$$

   The authors first compute a gate vector to determine the proportion of information that is to be updated for each attribute in user-specific value vectors. The gate weight for each attribute a is computed as

$$z_a = \text{sigmoid}\left(\boldsymbol{v}_a^{u\top} \cdot \boldsymbol{e}_a^i\right)$$

   With the update weight and update vector, we update each value vector in the value matrix V of user u accordingly

$$\boldsymbol{v}_a^u \leftarrow (1 - z_a) \cdot \boldsymbol{v}_a^u + z_a \cdot \boldsymbol{e}_a^i$$

2.2.3 The Complete Sequential Recommender

The scoring function is as below:

$$s_{u,i,t} = g(u, i, t) = \text{MLP}\left(\boldsymbol{p}_t^u\right)^\top \cdot \text{MLP}\left(\tilde{q}_i\right)$$

The framework of the recommender is as follow:

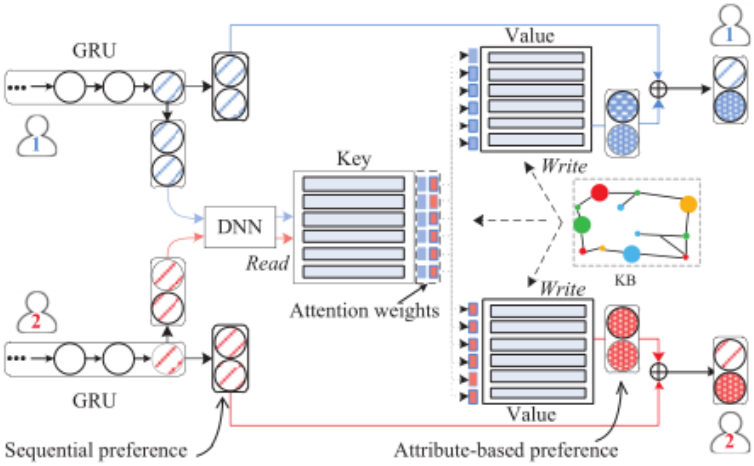

Figure 2.1: Performance

First, the GRU network is able to effectively capture temporal dependency, yielding a sequential representation for user preference. Second, the KV-MN part is able to characterize the detailed user interests over item attributes, yielding an attribute-based representation for user preference. Third, the hidden sequential preference representation is used to dynamically generate a set of attention weights over the explicit attributes, which provides the capacity of explaining the latent sequential preference in the attribute level.

For entity and relation embeddings, the authors learn them using the classic TransE model, and fix them in the learning process. While, for other parameters, we adopt the pairwise loss following BPR:

$$L = \sum_{u \in \mathcal{U}} \sum_{t=2}^{n_u} \sum_{j \in I_u^-} \log \sigma \left(g\left(u, i_t\right) - g(u, j) \right)$$

2.3 Experiments

The paper experiments with four benchmark datasets: **Music, ml-20m, ml-1m, Book**. The baselines include **BPR, NCF, CKE, FPMC, RUM, GRU, GRU++, GRUF, KSR**. The results are as follows:

Datasets	Methods	Next-Item Recommendation				Next-Session Recommendation			
		MAP	MRR	Hit@10	NDCG@10	MAP	MRR	Hit@10	NDCG@10
ml-20m	BPR	0.128†	0.128†	0.276†	0.144†	0.086†	0.165†	0.340†	0.099†
	NCF	0.094†	0.094†	0.205†	0.101†	0.083†	0.162†	0.354†	0.095†
	CKE	0.178†	0.178†	0.382†	0.209†	0.087†	0.153†	0.345†	0.104†
	FPMC	0.129†	0.129†	0.273†	0.144†	0.084†	0.157†	0.328†	0.096†
	RUM^I	0.267†	0.267†	0.523†	0.312†	0.122†	0.193†	0.399†	0.141†
	RUM^F	0.248†	0.248†	0.515†	0.295†	0.121†	0.197†	0.399†	0.141†
	GRU	0.282†	0.282†	0.522†	0.325†	0.079†	0.121†	0.264†	0.085†
	GRU++	0.277†	0.277†	0.549†	0.327†	0.127†	0.195†	0.397†	0.145†
	GRU_F	0.279†	0.279†	0.550†	0.329†	0.127†	0.197†	0.397†	0.146†
	KSR	**0.294** (+6.1%)	**0.294** (+6.1%)	**0.571** (+4.0%)	**0.344** (+5.2%)	**0.135** (+6.3%)	**0.209** (+7.2%)	**0.419** (+5.5%)	**0.156** (+7.6%)
ml-1m	BPR	0.178†	0.178†	0.396†	0.211†	0.171†	0.231†	0.472†	0.210†
	NCF	0.163†	0.163†	0.355†	0.189†	0.141†	0.192†	0.414†	0.172†
	CKE	0.158†	0.158†	0.350†	0.185†	0.134†	0.174†	0.366†	0.160†
	FPMC	0.305†	0.305†	0.549†	0.349†	0.245†	0.287†	0.517†	0.282†
	RUM^I	0.323†	0.323†	0.627†	0.382†	0.252†	0.290†	0.560†	0.298†
	RUM^F	0.263†	0.263†	0.577†	0.323†	0.220†	0.265†	0.555†	0.270†
	GRU	0.315†	0.315†	0.593†	0.368†	0.210†	0.222†	0.439†	0.241†
	GRU++	0.336†	0.336†	0.626†	0.393†	0.256†	0.291†	0.555†	0.304†
	GRU_F	0.340†	0.340†	0.636†	0.399†	0.259†	0.293†	0.559†	0.306†
	KSR	**0.356** (+6.0%)	**0.356** (+6.0%)	**0.655** (+4.6%)	**0.417** (+6.1%)	**0.276** (+7.8%)	**0.313** (+7.6%)	**0.570** (+2.7%)	**0.324** (+6.6%)
Music	BPR	0.227†	0.227†	0.458†	0.265†	0.151†	0.157†	0.320†	0.163†
	NCF	0.386†	0.386†	0.549†	0.413†	0.206†	0.228†	0.378†	0.224†
	CKE	0.371†	0.371†	0.541†	0.399†	0.215†	0.225†	0.386†	0.233†
	FPMC	0.349†	0.349†	0.489†	0.369†	0.140†	0.158†	0.290†	0.151†
	RUM^I	0.386†	0.386†	0.587†	0.422†	0.210†	0.220†	0.395†	0.229†
	RUM^F	0.332†	0.332†	0.562†	0.374†	0.201†	0.212†	0.399†	0.222†
	GRU	0.420†	0.420†	0.538†	0.436†	0.104†	0.131†	0.232†	0.112†
	GRU++	0.403†	0.403†	0.595†	0.437†	0.214†	0.224†	0.397†	0.233†
	GRU_F	0.404†	0.404†	0.594†	0.438†	0.214†	0.225†	0.397†	0.233†
	KSR	**0.427** (+1.7%)	**0.427** (+1.7%)	**0.607** (+2.0%)	**0.460** (+5.5%)	**0.223** (+4.2%)	**0.233** (+4.0%)	**0.403** (+1.5%)	**0.241** (+3.4%)
Book	BPR	0.222†	0.222†	0.505†	0.272†	0.216†	0.221†	0.505†	0.265†
	NCF	0.284†	0.284†	0.513†	0.325†	0.282†	0.290†	0.534†	0.327†
	CKE	0.248†	0.248†	0.494†	0.291†	0.246†	0.252†	0.512†	0.292†
	FPMC	0.147†	0.147†	0.324†	0.171†	0.149†	0.153†	0.338†	0.174†
	RUM^I	0.292†	0.292†	0.596†	0.350†	0.282†	0.288†	0.597†	0.341†
	RUM^F	0.300†	0.300†	0.610†	0.360†	0.278†	0.284†	0.590†	0.335†
	GRU	0.265†	0.265†	0.501†	0.305†	0.141†	0.144†	0.291†	0.157†
	GRU++	0.305†	0.305†	0.619†	0.366†	0.299†	0.305†	0.621†	0.360†
	GRU_F	0.306†	0.306†	0.619†	0.367†	0.299†	0.305†	0.620†	0.360†
	KSR	**0.353** (+15.7%)	**0.353** (+15.7%)	**0.653** (+5.5%)	**0.413** (+12.8%)	**0.345** (+15.4%)	**0.353** (+15.7%)	**0.661** (+6.4%)	**0.407** (+13.1%)

Figure 2.2: Performance

1. Among non-sequential recommendation baselines, BPR performs well on the two dense movie datasets, but poorly on music and book datasets, which are more sparse.

2. Among sequential recommendation baselines, the classic model FPMC performs worst (but still better than BPR in most cases).

3. Finally, we compare our proposed model KSR with all the baselines. It is clear to see that KSR is consistently better than these baselines by a large margin.

2.4 My Idea

In this paper, the authors proposed to extend the GRU-based sequential recommender by integrating it with knowledge-enhanced KV-MNs. The novel part is that the authors use the KG as a outer helper network instead of using it as a part of recommender. It can be easily used in basket recommendation, where we should change the item embedding method.

Chapter 3

KERL: A Knowledge-Guided Reinforcement Learning Model for Sequential Recommendation

3.1 Main Contribution

1. We formalize the sequential recommendation task into a Markov Decision Process (MDP), and fuse KG information to enhance the recommendation performance. To our knowledge, it is the first time that knowledge graph data has been explicitly discussed and utilized in RL-based sequential recommenders, especially for the exploration process.

2. We make three novel extensions in the MDP framework for sequential recommendation, including state representation, reward function and learning strategy. With the three major extensions, KG information has been effectively utilized and integrated into the RL-based sequential recommenders.

3. Empirical results on four real-world datasets show that our model can consistently outperform state-of-the-art baselines on both next-item and next-session recommendation tasks under different metrics.

3.2 Method

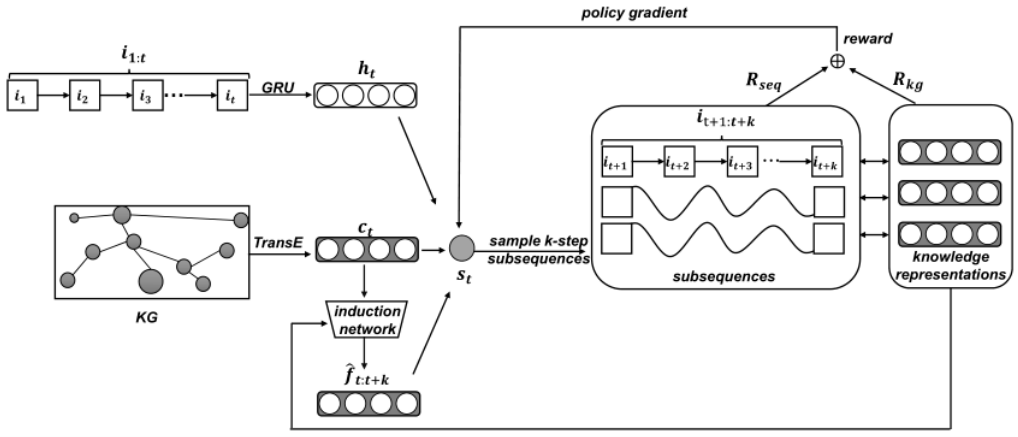

Figure 3.1: Structure

3.2.1 A MDP Formulation for Our Task

Just common MDP, only the state is special. which is enhanced by KG:

$$s_t = [i_{1:t}, \mathcal{G}]$$

The policy is as follow:

$$\pi\left(a_t \mid s_t\right)=\frac{\exp\left\{\mathbf{q}_{i_{j(a_t)}} \mathbf{W}_1 \mathbf{v}_{s_t}\right\}}{\sum_{i_j \in \mathcal{I}} \exp\left\{\mathbf{q}_{i_j} \mathbf{W}_1 \mathbf{v}_{s_t}\right\}}$$

Thus, the transition is as follow:

$$s_{t+1}=T\left(s_t, a_t\right)=T\left(\left[u, i_{1:t}, \mathcal{G}\right], i_{j(a_t)}\right)$$

3.2.2 Learning Knowledge-Enhanced State Representation

The total state is devided into three parts as below:

1. Sequence-level State Representation: For the first kind of state representations, we adopt a standard recurrent neural network for encoding previous interaction sequence:

$$\mathbf{h}_t=\mathrm{GRU}\left(\mathbf{h}_{t-1}, \mathbf{q}_{i_t} ; \Phi_{gru}\right)$$

2. Current Knowledge-level State Representation: The authors utilize the widely used KG embedding method TransE to derive an embedding vector for an item it. Furthermore, the authors use a simple average pooling method to aggregate all the KG embeddings of the historical items that a user has interacted with:

$$\mathbf{c}_t=\sum_{i=1}^{t} \mathrm{Average}\left(\mathbf{v}_{e_i}\right)$$

3. Future Knowledge-level State Representation: Based on current preference, our idea is to develop an induction network to directly predict the future preference. Specially, we construct a neural network using a Multi-Layer Perception. At time step t, we predict a k-step future preference representation taking as input the current preference representation ct:

$$\mathbf{f}_{t:t+k}=\mathrm{MLP}\left(\mathbf{c}_t ; \Phi_{mlp}\right)$$

Final state can be written as below:

$$\mathbf{v}_{S_t}=\mathbf{h}_t \oplus \mathbf{c}_t \oplus \mathbf{f}_{t:t+k}$$

3.2.3 Setting the Reward with Knowledge Information

Reward Decomposition

Based on the above motivation, at time step t, we define the k-step reward function by integrating two different reward functions:

$$R\left(s_t, a_t\right)=R_{seq}\left(i_{t:t+k}, \hat{i}_{t:t+k}\right)+R_{kg}\left(i_{t:t+k}, \hat{i}_{t:t+k}\right)$$

Sequence-level Reward

Inspired by the works in the evaluation of machine translation, we borrow the metric of BLEU for sequence recommendation. Formally, reward function is defined as:

$$R_{\mathrm{seq}}\left(i_{t:t+k}, \hat{i}_{t:t+k}\right)=\exp\left(\frac{1}{M} \sum_{m=1}^{M} \log prec_m\right)$$

where $prec_m$ is the modified precision and calculated as:

$$\mathrm{prec}_m=\frac{\sum_{p_m \in i_{t:t+k}} \min\left(\#\left(p_m, i_{t:t+k}\right), \#\left(p_m, \hat{i}_{t:t+k}\right)\right)}{\sum_{p_m \in i_{t:t+k}} \#\left(p_m, i_{t:t+k}\right)}$$

As we can see, such a reward function advocates the recommendation algorithm to generate more consistent m-grams from the actual sequence. It naturally measures the sequence-level performance for our task.

Knowledge-level Reward

In the second reward function, we do not focus on the exact match with item IDs. Instead, we consider measuring the quality of knowledge-level characteristics reflected in the sequences. In this paper, the authors use cosine similarity:

$$R_{kg}\left(i_{t:t+k}, \hat{i}_{t:t+k}\right) = \frac{\mathbf{c}_{t:t+k} \cdot \hat{\mathbf{c}}_{t:t+k}^{\top}}{\|\mathbf{c}_{t,t+k}\| \cdot \|\hat{\mathbf{c}}_{t:t+k}\|}$$

3.2.4 Learning

Training with Truncated Policy Gradient

In our task, our goal is to learn a stochastic policy π that maximizes the expected cumulative reward $\nabla J(\Theta)$ for all uses. The derivative of $\nabla J(\Theta)$ can be given below:

$$\nabla J(\Theta) = E_{\pi}\left[\sum_{u}\sum_{j=t}^{n} \gamma^{j-t} R_j \frac{\nabla \pi\left(a_t \mid s_t; \Theta\right)}{\pi\left(a_t \mid s_t; \Theta\right)}\right]$$

We employ a truncated policy gradient strategy to learn the parameters. Specifically, for each state s_t, KERL simulates a truncated k-step subsequence according to policy function. The learning process is written as follows:

$$\nabla\Theta = \sum_{j=t}^{t+k} \gamma^{j-t} R_j \frac{\nabla \pi\left(\hat{i}_t^{(l)} \mid s_t; \Theta\right)}{\pi\left(\hat{i}_t^{(l)} \mid s_t; \Theta\right)}$$

Training the Induction Network

To train such a neural network, a straightforward method is to fit it with ground-truth preference vector using regression loss. However, the KG information is likely to contain noise or irrelevant information for the recommendation task. Besides, the supervision signal for real-vector regression is sparse in our scenario. To better learn the induction network, the authors propose a pairwise ranking strategy to improve the training process.

Based on the simulated subsequences for state s_t, we can derive their corresponding knowledge representations. By calculating their reward values, we can form pairwise comparisons as additional constraints to improve the learning.

3.3 Experiments

The paper experiments with four benchmark datasets: **Amazon, LastFM**. The baselines include **FPMC, DREAM, GRU4Rec, Ripple, KGAT, GRUF, KSR**. The results are as follows:

Task	Dataset	Evaluation metric	Sequential-based Models			Knowledge-based Models		Hybrid Models			Improvement
			FPMC	DREAM	GRU4Rec	Ripple	KGAT	GRU_F	KSR	KERL	
Next-item recommendation	Beauty	Hit-Ratio@10	30.6	32.1	39.4	42.2	44.0	49.2	51.0*	**54.1**	3.1%
		NDCG@10	18.1	23.3	29.5	21.4	27.6	32.8*	32.2	**36.5**	3.7%
	CD	Hit-Ratio@10	21.3	24.5	50.5	58.4	63.4	64.2	68.3*	**73.7**	5.4%
		NDCG@10	11.6	15.6	32.9	37.6	41.7	40.6	45.0*	**50.8**	5.8%
	Books	Hit-Ratio@10	19.6	21.3	56.2	63.8	70.2	68.3	75.1*	**80.0**	4.9%
		NDCG@10	11.0	20.9	38.5	42.8	45.8	43.1	52.4*	**57.1**	4.7%
	LastFM	Hit-Ratio@10	31.0	35.5	52.8	52.5	55.8	58.2	62.7*	**64.2**	1.5%
		NDCG@10	19.7	22.6	40.7	38.2	42.1	44.1	48.1*	**50.1**	2.0%
Next-session recommendation	Beauty	Hit-Ratio@10	18.8	20.3	24.2	—-	—-	43.2	45.9*	**48.2**	2.3%
		NDCG@10	10.1	11.2	14.0	—-	—-	28.0	29.3*	**31.4**	2.1%
	CD	Hit-Ratio@10	17.0	20.5	31.3	—-	—-	54.5	57.2*	**60.5**	3.3%
		NDCG@10	8.71	10.2	16.7	—-	—-	31.4	32.2*	**36.8**	4.6%
	Books	Hit-Ratio@10	15.5	18.9	28.9	—-	—-	52.8	54.6*	**58.2**	3.6%
		NDCG@10	7.94	11.3	17.8	—-	—-	29.1	30.0*	**35.2**	5.2%
	LastFM	Hit-Ratio@10	19.1	20.2	22.0	—-	—-	31.9	41.5*	**44.1**	2.6%
		NDCG@10	11.0	11.4	12.7	—-	—-	18.8	22.8*	**25.0**	2.2%

Figure 3.2: Performance

1. For sequential-based models, comparing with the shallow model FPMC, the deep models DREAM and GRU4Rec obtain a better performance on all evaluation metrics.
2. By incorporating KG into recommender systems, the knowledgebased methods perform better than sequential-based methods on all evaluation metrics.
3. For hybrid models (i.e., knowledge+sequential), both GRUF and KSR perform better than the above two classes of methods.
4. Proposed approach KERL achieves the best performance among all the methods on four datasets.

3.4 My Idea

In this paper, the authors have presented a novel knowledge-guided reinforcement learning model, called KERL, for fusing KG information into a RL framework for sequential recommendation. The authors incorporate the KG information into RL learning, and then use RL to do the sequential recommendation. We can find out if there is any method using RL in next basket recommendation and then combine them together.