A novel recommendation algorithm with knowledge graph

L Liu and P Zhang

School of Software Engineering, South China University of Technology, Guangzhou 510641, China
E-mail: 201921044035@mail.scut.edu.cn

Abstract. Personalized recommendation is an important topic in recommendation algorithm research. Traditional collaborative filtering recommendation algorithms suffer from the sparseness and cold start problems. Existing knowledge graph based recommendation algorithms miss the high-order similarity at the subgraph level. This paper introduces a RNN-based distributed representation model of knowledge graph called KG-GRU, which uses a sequence containing nodes and relationships to model the subgraph similarity in the same embedded vector space. Then, a personalized recommendation algorithm based on knowledge graph called KG-GRU4Rec, is proposed. KG-GRU4Rec implements an end-to-end predictive user rating model. Experimental results demonstrate that KG-GRU learns more accurate knowledge about entities and relationships in the graph and KG-GRU4Rec outperforms the comparison algorithm in both hit ratio and average reciprocal ranking.

1. Introduction

There are several typical categories of personalized recommendation technologies: rule-based recommendation, collaborative filtering recommendation [1], content-based recommendation, social network-based recommendation technology [2] and knowledge graph [3] based recommendation technology.

Association rules are an unsupervised machine learning method that mines hidden associations in the target dataset to describe the rules and patterns of items. The main disadvantage is that it is difficult to evaluate the model and the rules. Collaborative filtering, while explores common interest among users, prefers to recommend popular items, resulting in a lack of novelty in recommendations.

In recent years, the recommendation system [4] emerges to rely on knowledge graph that integrates multi-source heterogeneous data to improve the effect of recommendation algorithms. However, traditional distributed representation methods for graph lack high-order similarities at the subgraph level when learning structure of the graph. This paper introduces a RNN-based distributed representation model for knowledge graph called KG-GRU, which uses multiple paths of entities and relationships to model subgraph similarity, and represent the relationship and entity in the same embedded vector space. Then, this paper proposes JUST, a jump or stay strategy, to guide random walks, avoiding manual construction of meta-paths. Finally, we present a novel recommendation algorithm based on knowledge graphs, KG-GRU4Rec, which is an end-to-end model that predicts user ratings, resolving the sparse and cold start issues.

The main contributions of this paper are double-folded:

1. A RNN-based distributed representation model for knowledge graph with a jump or stay strategy to guide random walks to sample the knowledge graphs.
2. A knowledge graph based personalized recommendation algorithm that implements an end-to-end model to predict user ratings.

2. Distributed representation of knowledge graph

2.1. KG-GRU model framework

KG-GRU, a RNN-based distributed representation model for knowledge graph, consists of two modules: the subgraph extraction module and the RNN training module, as shown in Figure 1 and 2 respectively. Particularly, RNN is implemented with gated GRU [5]. The subgraph extraction module adopts the JUST strategy to generate single-hop or multiple-hop paths for target node pairs, which are then batched into the RNN network. KG-GRU optimizes the representation vector and the weight parameters by maximizing the subgraph similarity.

Let \( E = \{ e_1, e_2, \ldots, e_m \} \) be the entity set, \( R = \{ r_1, r_2, \ldots, r_g \} \) be the relation set and \( G = (E, E, R) \) be the knowledge graph. \( E \) is the set of edges. Denote \( |E| = m \). Each edge \( e \in E \) corresponds to an element in the relation set \( R \), and \( g \ll m \). \( \psi(e) \in R \), \( \psi(\cdot) \) is the relational mapping function; each entity \( e \in E \) belongs to a certain ontology \( q \in Q \), that is, \( \phi(e) \in Q \), \( \phi(\cdot) \) is an ontology mapping function.

![Figure 1. KG-GRU subgraph extraction module.](image1)

![Figure 2. KG-GRU training module.](image2)

The goal of KG-GRU is to map the knowledge graph into a low-dimensional vector space, while retaining the semantic and structural information of the graph, to obtain a vector representation of entities and relationships: \( e_i \to E_i \in \mathbb{R}^d \ \forall i \in [n] \) and \( r_j \to R_j \in \mathbb{R}^d \ \forall j \in [g] \). On this basis, we can directly use representation vectors to compute the semantic similarity of entities in knowledge graph and apply them to downstream intelligent scenarios.

2.2. Knowledge subgraph extraction

Given two entities \( e_s, e_o \in E \), subgraph \( G_{so} \in G \) contains \( p_{so} \cup S_{so} \), where \( p_{so} = (s, r, o) \) is a single triple or a one-hop path; \( S_{so} = \{ p_1, p_2, \ldots, p_n \} \) is a multi-hop path set, \( p_i = \{ e_s, r_1, e_1, r_2, e_2, \ldots, e_o \} \). If there is no path connecting \( e_s \) and \( e_o \), then \( G_{so} = \emptyset \).

Assume the current node is \( e_t \), define \( E_{stay}(e_t) = \{ e | (e_t, e) \in E, \ \phi(e) = \phi(e_t) \} \) and \( E_{jump}(e_t) = \{ e | (e_t, e) \in E, \ \phi(e) = q \} \). The JUST strategy works through the following rules:

1. Jump or stay. The probability that JUST strategy chooses to stay in the same ontology is:

\[
Pr_{stay}(e_t) = \begin{cases} 1, & \text{if } E_{jump}(e_t) \cap Q, q \neq \phi(e_t) = \emptyset \setminus (1) \\
0, & \text{if } E_{stay}(e_t) = \emptyset \setminus \alpha, otherwise \\
\end{cases}
\]

where \( \alpha \in [0,1] \) is the initial stay probability, \( l \) represents the number of entities continuously visited in the same ontology. If there is no entity that belongs to the same ontology connected to \( e_t \),
then, jump. If there is no entity of the same ontology, then, stay. Otherwise, JUST chooses to stay with the exponential decay probability $\alpha$ and the next entity will be selected uniformly from $E_{\text{stay}}(e_i)$.

(2) If jump, which ontology to jump to. To balance the distribution of sampling nodes on different ontology, when selecting the next ontology $q$, we should exclude the $m$ recently adopted ontology. Let the queue $Q_{\text{hist}}$ of length $m$ hold the $m$ most recently used ontology in a FIFO manner. The ontology $q$ of the next entity will be uniformly selected from the following set:

$$Q_{\text{jump}}(e_i) = \begin{cases} \{ q \mid q \in Q \land q \notin Q_{\text{hist}}, E_q^{\text{jump}}(e_i) \neq \emptyset \}, & \text{if not empty} \\ \{ q \mid q \in Q, q \neq \phi(e_i), E_q^{\text{jump}}(e_i) \neq \emptyset \}, & \text{otherwise} \end{cases}$$ (2)

2.3. KG-GRU training process

The KG-GRU embedding layer learns the distributed representation for each element in a path and the GRU hidden layer learns the representation of the entire path. Suppose KG-GRU is training the $t$-th element $e_t$ on path $p_t$, the state and output of the hidden layer at step $t$ are given by:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$ (3)

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$ (4)

$$\tilde{h}_t = \sigma(W_{\tilde{h}} \cdot [r_t \cdot h_{t-1}, x_t])$$ (5)

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$ (6)

where $x_t$ is representation vector of the current element $e_t$, $h_{t-1}$ represents the hidden state passed down from the previous element, which contains information about the previous path, and $\tilde{h}_t$ represents the candidate hidden state of the current element. For a single path, GRU will output the representation vector $p_t$ of the entire path in the hidden layer of the last element. In $g_{s_o}$, the similarity between the one-hop path and the multi-hop path can be obtained by the dot product between vectors:

$$s(p_{s_o}, p_t) = p_{s_o}^T \cdot p_t$$ (7)

Therefore, we can calculate the similarity between all multi-hop paths and one-hop paths in the $s_{s_o}$ set and store it in the set $\{s_1, s_2, ..., s_n\}$. To model the similarity of subgraph, we use the Log-Sum-Exp function to calculate the similarity between $s_{s_o}$ and $p_{s_o}$:

$$g(p_{s_o}, s_{s_o}) = \log \left( \sum_{i=1}^{n} \exp(s_i) \right)$$ (8)

To optimize the similarity of subgraph, loss function of the subgraph $g_{s_o}$ is:

$$L(p_{s_o}, s_{s_o}) = \sum_{p'_{s_o} \in \Delta_{p_{s_o}}} \max[0, \gamma_1 - g(p_{s_o}, s_{s_o}) + g(p'_{s_o}, s_{s_o})]$$ (9)

where $\gamma_1$ represents the predefined interval value; $\max[0, \cdot]$ represents the hinge loss function; $\Delta_{p_{s_o}}$ is the negative sample set associated with $(s, r, o)$, defined as follows:

$$\Delta_{p_{s_o}} = \{(s', r, o)\} \cup \{(s, r', o)\} \cup \{(s, r, o')\}$$ (10)

To improve training efficiency and learn the first-order similarity of entities and relationships, we utilizes the TransE [6] translation model as constrains:

$$f(p_{s_o}) = -\|s + r - o\|_2$$ (11)

and the training loss function of triplet $p_{s_o}$ is defined as follows:

$$L(p_{s_o}) = \sum_{p'_{s_o} \in \Delta'_{p_{s_o}}} \max[0, \gamma_2 - f(p_{s_o}) + f(p'_{s_o})]$$ (12)

Combining the above two objective functions yields the final training loss function of KG-GRU:
\[ \mathcal{L} = \sum_{\mathcal{P}_{so} \in \Delta} \left( \mathcal{L}(\mathcal{P}_{so}) + \mathcal{L}(\mathcal{P}_{so}, \mathcal{S}_{so}) \right) \]  

(13)

The vector initialization hyper-parameter \( m \) is generally set to 6.

**Algorithm 1: KG-GRU Training Process**

1. **Input:** triples training set \( \Delta = \{ \mathcal{P} \} \); each triple corresponds to multi-hop path set \( \{ \mathcal{S} \} \); entity and relationship set: \( E \) and \( R \); predefined hinge function interval values \( \gamma_1 \) and \( \gamma_2 \); dimension \( d \) of representation vector; vector initialization hyper-parameter \( m \)
2. **Output:** KG-GRU network weight parameter; representation vector of entity \( e \in E \) and relation \( r \in R \)
3. **Initialization:** \( \forall e \in E, \forall r \in R; e, r \leftarrow \text{uniform}\left( -\sqrt{\frac{m}{d}}, \sqrt{\frac{m}{d}} \right) \)
4. Repeat the following steps until the model converges
5. \( e \leftarrow \frac{e}{\|e\|}, r \leftarrow \frac{r}{\|r\|} //\text{Normalized representation vector} \)
6. \( \Delta_{batch}, \mathcal{S}_{batch} \leftarrow \text{sample}(\Delta, \{ \mathcal{S} \}, \text{batch}_\text{size}) \)
7. \( \mathcal{C}_{batch} \leftarrow \emptyset //\text{Construct positive and negative sample sets of triples} \)
8. **for** \( \mathcal{P} \) **in** \( \Delta_{batch} : \)
9. \( \mathcal{P}' \leftarrow \text{sample}(\Delta_{\mathcal{P}}) //\text{Extract negative samples according to the formula} \)
10. \( \mathcal{C}_{batch} \leftarrow \mathcal{C}_{batch} \cup \{ (\mathcal{P}, \mathcal{P}') \} \)
11. Adam optimizes the model according to the following loss function:
\[ \sum_{(\mathcal{P}, \mathcal{P}') \in \mathcal{C}_{batch}} \nabla \left( \mathcal{L}(\mathcal{P}) + \mathcal{L}(\mathcal{P}', \mathcal{S}) \right) \]

3. Design of recommendation algorithms

3.1. **KG-GRU4Rec model**

KG-GRU4Rec framework is shown in Figure 3 below, the input data is a set of multi-hop paths in knowledge graph, which are generated by the random walk guided via the JUST strategy; the output result is the user’s prediction score. We treat the recommendation problem as a binary classification problem. Therefore, KG-GRU4Rec uses binary cross-entropy as the training function.

**Figure 3.** KG-GRU4Rec recommendation algorithm.

3.2. **KG-GRU4Rec score prediction**

While the attention mechanism [7] focuses the importance of different elements in one path, we use pooling operations to distinguish the importance of paths. Assume that final output of the path \( p_i \) through the hidden layer of GRU is \( h_{iT_{end}} \), the maximum pooling operation for the \( n \) paths is defined as follows:

\[ h[m] = \max_{1 \leq i \leq n} h_{iT_{end}}[m] \]  

(14)
where $h[m]$ is the element at the $m$-th dimension. Finally, KG-GRU4Rec uses fully connection layer to quantify the degree of association between user $u_i$ and item $e_j$:

$$\hat{r}_{ij} = \sigma(W_r \cdot h + b_r)$$

(15)

where $W_r$ is the regression coefficient, $b_r$ is the paranoid term, and $\sigma(\cdot)$ is the sigmoid nonlinear function, which ensures that the model output control in the range of [0, 1].

3.3. KG-GRU4Rec Training Process

The loss function of KG-GRU4Rec is defined as follows:

$$L = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{r_{ij} \in \mathcal{D}_{\text{train}}} \text{BCELoss}(\hat{r}_{ij}, r_{ij})$$

(16)

where $\mathcal{D}_{\text{train}} = \{(u_i, v_j, \hat{r}_{ij}, p(u_i, v_j)) | u_i \in U, v_j \in V, r_{ij} \in R_{nxm}\}$ represents the training data set, and $p(u_i, v_j)$ represents multiple paths between user and item.

Algorithm 2: The training process of KG-GRU4Rec

1. **Input:** implicit rating matrix $R_{nxm}$; user set $U$ and item set $V$; knowledge graph $\mathcal{G}$; dimension $d$ of representation vector; vector initialization hyper-parameter $m$
2. **Output:** KG-GRU4Rec network weight parameter; entity and relationship representation vector of embedding layer
3. **Initialization:** $\forall e \in \mathcal{G}, \forall r \in \mathcal{G} : e, r \leftarrow \text{uniform}(-\frac{m}{\sqrt{d}}, \frac{m}{\sqrt{d}})$
4. **for** each $u_i \in U$ **do**
5. $\Delta_+ = \{(u_i, v_j)| r_{ij} = 1\}$ //Based on $R_{nxm}$, get the training set of positive samples
6. $\Delta_- = \{(u_i, v_k)| r_{ik} = 0\}$ //Negative sample training set
7. **for** each $(u, v) \in \Delta$ **do**
8. $e \leftarrow \frac{e}{\|e\|}, r \leftarrow \frac{r}{\|r\|}$ //Normalized representation vector
9. $\Delta_{\text{batch}}, p_{\text{batch}} \leftarrow \text{sample}(\Delta, \{p\}, \text{batch} \_ \text{size})$
10. **for** each $(u, v) \in \Delta_{\text{batch}}$ **do**
11. $c_h \leftarrow \emptyset$ //Construct the hidden layer state collection
12. $\forall p_j \in p(u, v)$ **do**
13. $h \leftarrow \text{pool}(c_h)$
14. $\hat{r}_{ij} = \sigma(W_r \cdot h + b_r)$
15. Adam optimizer optimizes the model according to the loss function

4. Experiments

4.1. Dataset
This experiment builds a movie knowledge graph based on the MovieLens_1M dataset [8]. After knowledge mapping and extraction, the experimental data contains rating data of 3625 movies from 6040 users. The overall sparsity of rating data is as high as 95.53%.

4.2. KG-GRU4Rec experiment

4.2.1. Evaluation indicators. To evaluate the recommendation performance of each algorithm model, each model first needs to predict the score \( test(u_i) \) of each user \( u_i \) on the test item, and generate a Top-10 recommendation list in descending order. We use the widely used Hit and MRR as evaluation indicators. MRR represents the importance of recommendation ranking:

\[
MRR = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{v_j \in test(u_i)} \frac{1}{\text{rank}(u_i, v_j)} \right)
\]

where \( test(u_i) \) represents the user’s movie test set, \( n \) represents the total number of test users; \( \text{rank}(u_i, v_j) \) represents the ranking of positive sample \( v_j \) in test set in the recommendation list.

4.2.2. Experimental results and analysis. The baseline models selected in this section are: MostPopular, KPRN [9]. The average results of Top-10 recommendation tasks for each model are shown in the following table 1:

| metrics          | Hit@10 | MRR@10 |
|------------------|--------|--------|
| MostPopular      | 0.164  | 0.035  |
| KPRN             | 0.582  | 0.277  |
| KG-GRU4Rec (max) | 0.653  | 0.295  |
| KG-GRU4Rec (avg) | 0.679  | 0.304  |

MostPopular, as a non-personalized recommendation strategy, has worse performance than other algorithm models. KPRN, which calculates path score first, and then pools multiple scores with weights, is inferior to the KG-GRU4Rec model proposed in this paper. To distinguish the importance of different paths between users and movie nodes, KG-GRU4Rec uses two pooling operations on the path vector output by GRU: max-pooling and avg-pooling. Experimental results show that the average pooling strategy is more effective and more suitable for Top-N recommendation tasks based on implicit interactive data. This is consistent with our intuitive conjecture that user preferences for items are determined by a combination of factors.

5. Conclusion

This paper focuses on the personalized scheme based on the knowledge graph. By mining high-order similarity at the subgraph level, we present a RNN-based knowledge graph distributed representation model KG-GRU to sample knowledge graph using the JUST strategy. Based on these, we design a novel recommendation algorithm KG-GRU4Rec with additional diversities and interpretability. Experimental results demonstrate that it is superior to competing ones.

Acknowledgment

This work is supported by the project “Construction and Intelligent Applications of Industrial Knowledge Graph for Typical Scenarios in Electronic and Information Industry” (Project#:TC2008032).
References

[1] Goldberg D, Nichols D A, Oki B M, et al. Using Collaborative Filtering to Weave an Information Tapestry[J]. ACM, 1992, 35(12): 61–70.

[2] Ozsoy M G, Polat F, Alhajj R. Time preference aware dynamic recommendation enhanced with location, social network and temporal information[C]// ACM International Conference on Advances in Social Networks Analysis and Mining. IEEE Computer Society, 2016: 999-916.

[3] Pan J Z, Vetere G, Gomez-Perez J M, et al. Exploiting Linked Data and Knowledge Graphs in Large Organisations[M]. Springer International Publishing, 2017.

[4] Szekely P, Knoblock C A, Slepicka J, et al. Building and using a knowledge graph to combat human trafficking[C]// International Semantic Web Conference on New York, 2015: 205–221.

[5] Dey R, Salem M F. Gate-variants of Gated Recurrent Unit (GRU) neural networks[C]// IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017: 1597-1600.

[6] Bordes A, Usunier N, Garcia-Duran A, et al. Translating Embeddings for Modeling Multi-relational Data[C]// Advances in neural information processing systems, 2013:2787-2795.

[7] Kelvin Xu, Jimmy Ba, Ryan Kiros, et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[C]// In International Conference on Machine Learning. ICML, 2015: 2048–2057.

[8] Movie Lens datasets[EB/OL]. https://grouplens.org/datasets/movielens/, 2016.

[9] Xiang Wang, Dingxian Wang, Canran Xu. Explainable Reasoning over Knowledge Graphs for Recommendation[C]// AAAI Conference on Artificial Intelligence, 2019: 5329-5336.