Knowledge-Guided Deep Reinforcement Learning for Interactive Recommendation

Xiaocong Chen

A thesis in fulfillment of the requirements for the degree of

Master of Philosophy

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

June 2020

Thesis/Dissertation Sheet

Surname/Family Name	:	**CHEN**
Given Name/s	:	**Xiaocong**
Abbreviation for degree as give in the University calendar	:	**MPhil**
Faculty	:	**Engineering**
School	:	**Computer Science and Engineering**
Thesis Title	:	**Knowledge-Guided Deep Reinforcement Learning for Interactive Recommendation**

Abstract 350 words maximum: (PLEASE TYPE)

The interactive recommendation aims to accommodate and learn from dynamic interactions between items and users to achieve responsiveness and accuracy in recommendation systems. Reinforcement learning is inherently advantageous for coping with dynamic/interactive environments and thus has attracted increasing attention in interactive recommendation research. However, most existing works tend to learn stationary user interests, while neglecting that they are dynamic in nature.

The dissertation starts with the introduction of the recommendation system and its applications. This is then followed by the detailed literature review which covers three main related areas: Sequence-aware Recommendation, Interactive Recommendation and the Knowledge-aware Recommendation System. The dissertation also reviews the reinforcement learning based applications in recommendation systems and discuss the advantages and shortcomings. After that, this dissertation reports a general problem statement about the interactive recommendation system and the identified challenges to be tackled, including user dynamic interest modeling, and computational cost of reinforcement learning optimization, and performance degradation for reinforcement learning based recommendation systems.

In particular, we propose a set of techniques and models for the improved interactive recommendation via reinforcement learning. We propose a new model for learning a distributed interaction embedding, which can capture user's dynamic interest in a compact and expressive manner. Inspired by the recent advance in Graph Convolutional Network and the knowledge-aware recommendation, we design a Knowledge-Guided deep Reinforcement learning (KGRL) model to harness the advantages of both reinforcement learning and knowledge graphs for the interactive recommendation. This model is implemented within the actor-critic network framework. It maintains a local knowledge network to guide decision-making process during the training phase and employs the attention mechanism to discover long-term semantics between items. To reduce the computational cost of reinforcement learning, we take a further step to design an enhanced optimization strategy by narrowing down the space of updating steps and turning the reward function.

We have conducted comprehensive experiments in a simulated online environment for the three proposed methods, which show consistent improved performance of our models against the baselines and state-of-art methods in the literature. Finally, this dissertation discusses the future work and potential further improvement for interactive recommendation systems.

ORIGINALITY STATEMENT

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed ……..

Date 22/04/2020 ……..

COPYRIGHT STATEMENT

Signed Xiaocong Chen

Date 23/07/2020

AUTHENTICITY STATEMENT

‘I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.’

Signed Xiaocong Chen

Date 23/07/2020

INCLUSION OF PUBLICATIONS STATEMENT

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in their thesis in lieu of a Chapter if:

- The candidate contributed greater than 50% of the content in the publication and is the "primary author", ie. the candidate was responsible primarily for the planning, execution and preparation of the work for publication
- The candidate has approval to include the publication in their thesis in lieu of a Chapter from their supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis

Please indicate whether this thesis contains published material or not:

☐ This thesis contains no publications, either published or submitted for publication
(if this box is checked, you may delete all the material on page 2)

☒ Some of the work described in this thesis has been published and it has been documented in the relevant Chapters with acknowledgement
(if this box is checked, you may delete all the material on page 2)

☐ This thesis has publications (either published or submitted for publication) incorporated into it in lieu of a chapter and the details are presented below

CANDIDATE'S DECLARATION

I declare that:

- I have complied with the UNSW Thesis Examination Procedure
- where I have used a publication in lieu of a Chapter, the listed publication(s) below meet(s) the requirements to be included in the thesis.

Candidate's Name	Signature	Date (dd/mm/yy)
Xiaocong Chen		22/04/2020

Abstract

The interactive recommendation aims to accommodate and learn from dynamic interactions between items and users to achieve responsiveness and accuracy in recommendation systems. Reinforcement learning is inherently advantageous for coping with dynamic/interactive environments and thus has attracted increasing attention in interactive recommendation research. However, most existing works tend to learn stationary user interests, while neglecting that they are dynamic in nature.

The dissertation starts with the introduction of the recommendation system and its applications. This is then followed by the detailed literature review which covers three main related areas: Sequence-aware Recommendation, Interactive Recommendation and the Knowledge-aware Recommendation System. The dissertation also reviews the reinforcement learning based applications in recommendation systems and discuss the advantages and shortcomings. After that, this dissertation reports a general problem statement about the interactive recommendation system and the identified challenges to be tackled, including user dynamic interest modeling, and computational cost of reinforcement learning optimization, and performance degradation for reinforcement learning based recommendation systems.

In particular, we propose a set of techniques and models for the improved interactive recommendation via reinforcement learning. We propose a new model for learning a distributed interaction embedding, which can capture user's dynamic interest in a compact and expressive manner. Inspired by the recent advance in Graph Convolutional Network and the knowledge-aware recommendation, we design a Knowledge-Guided deep Reinforcement learning (KGRL) model to harness the advantages of both reinforcement learning and knowledge graphs for the interactive recommendation. This model is implemented within the actor-critic network framework. It maintains a local knowledge network to guide decision-making process during the training phase and employs the attention mechanism to discover long-term semantics between items. To reduce the computational cost of reinforcement learning, we take a further step to design an enhanced optimization strategy by narrowing down the space of updating steps and turning the reward function.

We have conducted comprehensive experiments in a simulated online environment for the three proposed methods, which show consistent improved performance of our models against the baselines and state-of-art methods in the literature. Finally, this dissertation discusses the future work and potential further improvement for

interactive recommendation systems.

Acknowledgements

There are large number of people provided help during my candidature in UNSW, and this dissertation would not have been conceivable without their support. First of all, I would like to express my thanks to my primary supervisor, Dr. Wei Liu for his support and guidance throughout my study. Thanks for Wei's generous idea which provides to my research and provided me all kind of necessary resources, while spent bunch of time in guiding and supporting me. In addition, I also want to send my appreciation to my jointly supervisor Dr. Wenjie Zhang for her advice and supporting. It's my appreciative to have chance working with Dr. Lina Yao during my candidature. Thanks for her enlightening academic discussions and insightful recommendations. Much appreciated for her illuminating scholarly conversations and insightful advice and guidance. I would like thanks again for those three academic advisors who spent lots of time on proofreading, guiding and supporting me about how to read and write a successful research paper.

Furthermore, I would like to thanks for all my co-authors: Chaoran Huang, Xianzhi Wang, Xiang Zhang, Manqing Dong, Yuanjiang Cao and Chang Ge for their brilliant ideas and the effect taken on the proofreading and discussing.

At the end, I want to thanks my parents for their love, support and encouragement. They are the source of my power, courage and confidence for me to pursue my goal.

Abbreviations

RL Reinforcement Learning

MAB Multi-Armed Bandit

UCB Upper Confidence Boundary

CF Collaborative Filtering

CNN Convolutional Neural Network

RNN Recurrent Neural Network

LSTM Long-short Term Memory

GRU Gated Recurrent Unit

MF Matrix Factorization

nDCG normalized Discounted Cumulative Gain

Prec@N Precision at top N

Rec@N recall at top N

NN Nerual Network

$\mathcal{U}$ Set of users

$\mathcal{I}$ Set of items

$\mathcal{R}$ Set of relations

$\mathcal{E}$ Set of entities

$|\cdot|$ Number of unique elements in $\cdot$

$\mathcal{S}_{u,t}$ User u's recent actions before timestamp t

$\mathcal{G} = (\mathcal{E}, \mathcal{R})$ Constructed Knowledge Graph

$\mathbb{E}$ Item embedding

W parameter matrices

S_t state space at timestamp t

a_t action space at timestamp t

d dimension of the latent space

KG Knowledge Graph

CKG Collaborative Knowledge Graph

VAE Variational Autoencoder

CQA Community Question Answering

TRPO Trust Region Policy Optimization

GNN Graph Neural Network

GCN Graph Convolutional Network

GAT Graph Attention Network

EEG Electroencephalography

fMRI Functional magnetic resonance imaging

BCI Brain Computer Interface

KL Divergence Kullback Leibler Divergence

Contents

List of Figures

List of Tables

List of Algorithms

List of Publications

This dissertation contains parts of content of the following publications:

[1] X. Chen, C. Huang, X. Zhang, X. Wang, W. Liu, and L. Yao, "Expert2vec: Distributed expert representation learning in question answering community," in *International Conference on Advanced Data Mining and Applications*, pp. 288–301, Springer, 2019. (CORE RANK: B).

[2] X. Chen, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, "Knowledge-guided deep reinforcement learning for interactive recommendation," in *The International Joint Conference on Neural Networks*, IEEE, 2020. (CORE RANK: A).

I have also participated in the following publications during my candidature as the co-author:

[5] X. Zhang, X. Chen, L. Yao, C. Ge, and M. Dong, "Deep neural network hyperparameter optimization with orthogonal array tuning," in *International Conference on Neural Information Processing*, Springer, 2019. (CORE RANK: A).

[6] X. Zhang, X. Chen, M. Dong, H. Liu, C. Ge, and L. Yao, "Multi-task generative adversarial learning on geometrical shape reconstruction from eeg brain signals," in *International Conference on Neural Information Processing*, 2019. (CORE RANK: A).

[7] Y. Cao, X. Chen, L. Yao, X. Wang, and W. E. Zhang, "Adversarial attack and detection on reinforcement learning based recommendation system," in *The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2020. (CORE RANK: A*).

Chapter 1

Introduction

Recommendation systems have been widely used by industry giants such as Amazon, YouTube, and Netflix to identify relevant, personalized content from large information spaces. Modern recommendation systems are under much pressures for coping with emerging new users, ever-changing pools of recommendation candidates, and context-dependent interests [1, 2, 3, 4]. It has also been widely used in Internet Of Things as a part of Smart Home [5, 6, 7]. However, traditional recommendation methods often focus on modelling user's recurrent preference and may not reflect the dynamics and evolution of user interest and environments. In this context, interactive recommendation arises as an effective solution that incorporates dynamic recommendation processes to improve the recommendation performance. An interactive recommendation system would recommend items to an individual user and then receive the feedback to adjust its policies during the iterations [8]. Many studies model interactive recommendation as a Multi-Armed Bandit (MAB) problem [9, 10, 11]. Such methods generally assume that a user's preference is static during the recommendation and focus on the trade-off between immediate and future

rewards. Therefore, they face challenges of handling environments with dynamically changing user preference or interest. Reinforcement learning (RL) is a promising approach to the interactive recommendation. Considerable efforts have shown the outstanding performance of RL methods in recommendation systems [12, 13, 14] for its ability to learn from user's instant feedback. Given its potential to handle dynamic interactions, RL has been widely regarded as a potential tool for the interactive recommendation. However, most existing RL techniques in interactive recommendation focus on the usefulness instead of performance. For example, Liu et al. [15] employ the RL to increase the recommendation diversity and do not focus on the efficacy. The primary reason is that the agent has partial information about the environment, making it difficult to control the decision-making process properly. Besides, interactive recommendation systems usually contain a large number of discrete candidate actions, leading to high time complexity and low accuracy of RL-based techniques. Moreover, all the Deep Q-Networks (DQN)-based studies [16, 13, 17, 5] struggled with a large number of discrete actions because DQN contains a maximise function which considers all actions. When the size of action increases, the maximise function will become extremely slow, or even get stuck. The policy gradient-based methods would get stuck in this case as well because it may converge in the local minimum instead of the global minimum. Recently, knowledge-aware recommendation systems become popular as the knowledge graph can transfer the relation to contextual information and boost the recommendation performance [18, 19].

1.1 Related Work

In this section, we will conduct a detailed literature review which includes the sequential recommendation system, interactive recommendation system and the knowledge-aware recommendation system. The interactive recommendation can be considered as a type of sequential recommendation as the interactive process is a time-series sequential process [1].

1.1.1 Sequential Recommendation System

The sequential recommendation system aims to mine user preference based on the sequential data or time-series data [20, 21]. The most common method is based on the Markov Chain. Garcin et al. [22] propose a model based on the Markov chain. In particular, it uses the variable-order Markov model which is the extension of the Markov chain. The variable-order Markov model is also known as the context tree. Their model constructs a context tree by inputting the sequential data. Then a local prediction model called expert of each context is added as the node in this tree. However, these approaches fully depend on the expert to predict and make a recommendation, which may lead to substantial bias. In this case, another variable is introduced for each node to represent the usefulness of the corresponding expert.

Wu et al. [23] propose a model that used personalized Markov Embedding for music recommendation. Their model builds a Euclidean space which generates a joint embedding space of music and users together and adopts the distance measurement to represent the quantified relationship between music and users embeddings. Given user's latest action, their model is able to perform personalized recommendations based on the 'learn to rank' algorithm.

Hidasi et al. [24] propose a recurrent neural network(RNN) based model to make next-item recommendation. The gated recurrent unit(GRU) is adopted to capture the sequential features. In addition, a method called session-parallel mini-batch is introduced to process the session data in parallel. It can generate a mini-batch which takes the inputs from the model and splits them into different sessions. Twardowski et al. [25] make some improvement based on Hidasi's work to boost the performance by employing the matrix factorization and explicit contextual session modelling . Furthermore, Tan et al. [26] provide another approach that applied RNN into session-aware recommendation system by using data augmentation to pre-process the data and account for temporal shifts in user behaviour. Quadrama et al. [27] propose the Hierarchical Neural Network which adopted two GRUs, one is for modeling the user and the other is for modeling the session. Moreover, the session-parallel mini-batch mechanism is used for user identifiers during the training. In a summary, all the works mentioned in the above are based on the GRU with modification on input's format or customization of the GRU.

Ko et al. [28] propose a new model called the Collaborative Recurrent Neural Network Model based on the RNN Language Model [29] which can analyze and track sequences of user activities. Wu et al. [30] propose a model which fed the dynamic rating-review embedding into the RNN. Tang et al. [31] propose a model that uses the convolutional neural network(CNN) to conduct the sequential recommendation system, inspired by [32] to treat the sequential data as the images in the recommendation system.

However, the Markov chain-based models will be affected by the data sparsity [33, 34] issue. The deep-learning-based method can not handle the dynamics if user's interest changes sharply, which may result in low efficient [35]. In order to improve the

performance of the sequential recommendation system, the memory network comes to a possible solution [36, 37, 38]. However, user's evolving interest modeling is still not addressed well.

1.1.2 Interactive Recommendation System

Interactive recommender systems naturally aligns with the requirement of representing dynamic user interactions. Therefore, it attracts much attention in recent years. Most existing works treat interactive recommendation as a Multi-Armed Bandit (MAB) problem. And the primary solution lies in finding an Upper Confidence Bound (UCB). Li et al. [39] employ the first linear model to calculate the UCB for each arm. Since then, many researchers combine other techniques such as matrix factorization, to find the UCB [40]. For example, Wang et al. [11] propose a new approach by choosing a dependent arm to calculate the UCB; Shen et al. [41], instead, propose to solve MAB with deep neural networks.

There are two similar concepts relevant to the interactive recommendation: conversational recommendation and active learning-based recommendation. Active learning-based recommendation [42] is a method which used on few-shot samples. Wang et al. [43] adopts the Restricted Boltzmann Machine(RBM) to enrich item information to relief the lack of the annotated data. Zhao et al. [44] integrate the transfer learning and active learning to conduct the recommendation with limited data. Conversational recommendation [45] is a type of recommendation system major focuses on mining user's intention from text data by using NLP technique. Kraus et al. utilize both explicit and implicit information to represent user's preference [46]. Ren et al. [47] adopts the reinforcement adversarial learning to capture the user's intention iteratively. Those two concepts provide new approaches about how

to utilize and mining user's intention from complex data. However, the interactive recommendation place particular emphasis on utilizing the data generated during the interactive process.

Recent studies have shown the effectiveness of reinforcement learning in modelling interactions-related recommendation processes, where the recommendation problems are usually formulated as the Markov Decision Processes. One approach is based on Deep Q-learning (DQN) [48], which maximizes the Q-value from the predicted item and the target item. Zheng et al. [14] combine the DQN with the Dueling Bandit Gradient Decent (DBGD) [49] policy to recommend news. Another thread of methods is DDPG-based [50]. Such methods aim to let the agent learn a proper policy instead of using the Q-value. For example, Liu et al. [15] adopt DDPG to promote the diversity in interactive recommendation; Zhao et al. [13] use DDPG for the page-wise recommendation. It is also worth mentioning that knowledge graphs can be useful for guiding explainable recommendation [18]. Knowledge-aware recommendation systems heavily rely on the use of relation inference to generate paths for recommendations [51]. Wang et al. [52] show graph convolutional network can help learn neighbour representations and thus boost the recommendation performance. Another approach for knowledge aware recommendation is the embedding based [53, 54].

1.1.3 Knowledge-aware Recommendation System

Recently, knowledge-aware recommendation systems become popular since knowledge graphs can transfer the topological relation to contextual information and boost the recommendation performance [18, 19]. The knowledge graph was initially used

in search engine like Google. The knowledge graph can be represented as:

$$\mathcal{G} = (\mathcal{E}, \mathcal{R})$$

The knowledge graph uses the entities $\mathcal{E}$ to represent the nodes, and the relation $\mathcal{R}$ to represent the edges. Based on the graph $\mathcal{G}$, one then can find the entities which have similar properties. However, when the dataset become larger, the knowledge graph will become extremely big and cost substantial memory, which may affect the efficiency of model learning. In addition, knowledge graph can be useful for providing guidance in explainable recommendation [18] because the existence of the relation. Knowledge-aware recommendation systems heavily relies on the usage of relation inference to generate paths for recommendations [51]. Common approach for knowledge aware recommendation is the embedding based which embeds the graph into the high dimensional latent space [53, 54]. Besides, Wang et al. [52] show that the graph convolution can help to learn the neighbour representations and thus boost the recommendation performance. All those mentioned methods are trying to reduce the graph size and boost the efficiency of the recommendation [55]. The knowledge graph embedding have several ways which includes: TransE [56], TransH [57], TransR [58], TransD [59], GCN [60] and its variant [61]. Suppose there is a triplet h, r, t which is the entity-relation-entity triplet to represent an instance of the knowledge graph. The TransE is a score function can be defined as the L-2 Norm and it assume that $\overrightarrow{h} + \overrightarrow{r} \approx \overrightarrow{t}$:

$$f_r^E(h,t) = \left\| \overrightarrow{h} + \overrightarrow{r} - \overrightarrow{t} \right\|_2^2$$

where $\overrightarrow{h}$, $\overrightarrow{t}$, $\overrightarrow{r}$ is the vector representation for h, t, r. However. the entities may have different representations, based on that the TransH use the projection to project the entities into hyper-planes which can rewrite the score function as:

$$f_r^H(h,t) = \left\| \overrightarrow{h}_\perp + \overrightarrow{r} - \overrightarrow{t}_\perp \right\|_2^2$$

where $\overrightarrow{h}_{\perp}$ is the hyper-plane $\overrightarrow{h}_{r}$ representation for h where can be defined as:

$$\overrightarrow{h}_{\perp} = \overrightarrow{h} - \overrightarrow{w}_r^{\intercal} \overrightarrow{h} \overrightarrow{w}_r$$

Similarly, the representation for $\overrightarrow{t}$ is as follows:

$$\overrightarrow{t}_{\perp} = \overrightarrow{t} - \overrightarrow{w}_r^{\intercal} \overrightarrow{t} \overrightarrow{w}_r$$

Inspired by the TransH, the TransR uses the project matrix M_r to map the relation r:

$$f_r^R(h,t) = \|\overrightarrow{h_r} + \overrightarrow{r} - \overrightarrow{t_r}\|_2^2$$

where $\overrightarrow{h_r}$ and $\overrightarrow{t_r}$ are defined as:

$$\overrightarrow{h_r} = \overrightarrow{h} M_r, \overrightarrow{t_r} = \overrightarrow{t} M_r$$

TransD by replace the projection matrix M_r with the product of two projection vector:

$$f_r^D(h,t) = \left\| \overrightarrow{h}(\overrightarrow{r}_p \overrightarrow{h}_p^{\intercal} + I) + \overrightarrow{r} - \overrightarrow{t}(\overrightarrow{r}_p \overrightarrow{t}_p^{\intercal} + I) \right\|_2^2$$

where I is the identity matrix. However, all those embedding methods focus on partial information which are entities or relations. This means that these methods may lose some important information. In such cases the GCN was proposed to learn the embedding for the whole graph. The GCN is type of neural network where the neural network layer can be written as:

$$H^{l+1} = f(H^l, A)$$

where L is the number of layers, A is the adjacency matrix which used to represent the whole graph, H is the feature matrix at l layer and function $f(\cdot,\cdot)$ is the function

which can be chosen arbitrarily. The GCN maintain a very simple propagation rule between layers which is:

$$f(H^l, A) = \sigma(AH^lW^l)$$

where W^l is the weight matrix at layer l, and $\sigma(\cdot)$ is the (non-linear) activation function . We can rewrite the above in the vector-form as follows:

$$h_{e_i}^{l+1} = \sigma\left(\sum_j \frac{1}{c_{ij}} h_{e_j}^l W^l\right)$$

where c_{ij} is the normalization constant for the edge (e_i, e_j), e_i is the node from $\mathcal{G}$, $h_{e_i}^{l+1}$ is the feature representation for node e_i at layer $l+1$. More details in relation to the above formulations are discussed in Chapter 3.

1.2 Reinforcement Learning in Recommendation System

As mentioned in the last section, the reinforcement learning can be a solution for the interactive recommendation system. In this section, we will illustrate the reinforcement learning technique and its application in the recommendation system.

Recommendation system can recommend k items to users on a single page, the user can provide some feedback by clicking one of those choices or switch the pages. After the feedback is provided, the system will record it and recommend another k items based on user's feedback. RL has two different branches which are model-based RL and model-free RL. The model-free RL based recommendation systems require a large dataset which contains a large number of user actions so that it can identify a 'good' policy. The recommendation system will require a complex

structure if the size of dataset increases. RL initially comes from the Markov decision process(MDP), which is defined as:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R) \text{ Where } P \in [0, 1]$$

where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ represents the set of actions, P is the probability of transition which is normally written as $P(s_{t+1}|s_t, a_t)$ and $s_{t+1}, s_t \in \mathcal{S}, a_t \in \mathcal{A}$ which represents the probability of a_t transfering from s_t to s_{t+1} during the period $[t, t+1]$, and R is the reward function. If we consider the discount factor γ, the MDP can be written as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$.

However, the model-based RL methods often suffer from the computation difficulty when $\mathcal{S}, \mathcal{A}$ becomes large. In this dissertation, we prefer to use the model-free RL methods. Generally speaking, the model-free RL methods involve three different approaches, i.e., value-based methods, policy-based methods and hybrid methods. The traditional value-based RL method is the Temporal difference(TD) which are trying to find the optimal value V^* by iteration with learning rate η:

$$V(s_t) \leftarrow (1 - \eta)V(s_t) + \eta \left(R(s_{t+1}) + \gamma V(s_{t+1})\right)$$

In some cases, the value will depend on both states and actions, so we have the Q-learning. In Q-learning, we use the Q-value $Q(s, a)$ to determine the optimal policy π^*, different from the MDP, the Q-value baed on the pair of action a and state s instead of using state s only. The definition of π^* can be written as:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$$

the Q^* is the optimal Q-value where can be defined as:

$$Q^*(s, a) = R(s) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1})$$

This formula was used when one knows the certain state-action pair (s, a), and γ is the discount factor which is used in a long-term RL. During the training process the Q-value will be updated iteratively based on:

$$Q(s,a) \leftarrow (1-\eta)Q_o(s,a) + \eta(R(s) + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a))$$

The Q-learning is an off-policy learning which means it will learn from different policy and try to figure out the value. And it normally work in the continuous action space [62]. However, the TD method is the on-policy learning which means it can only learn different values in the same policy. Benefiting from the neural network, the Q-learning was extended to the deep Q-learning(DQN). The DQN will pass the state s into a neural network and find out many q-values at once. The DQN have a similar target with the normal Q-learning, the DQN try to do:

$$\min[R(s) + \gamma \max_{a'} Q_w(s_{t+1}, a') - Q_w(s_t, a_t)]^2$$

where the $Q_w(s, a)$ is parametrized by the neural network weight w. During the optimization of the neural network, we only need to apply the gradient descent into the term $Q_w(s_t, a_t)$ which is the current q value for current state s_t and action a_t. In addition, to help the convergence, the DQN uses the technique called experience replay that can store recent j experience pairs $(s_t, a_t, R(s), s_{t+1})$ with a replay batch of size j. The DQN will choose action based on the greedy algorithm from the reply buffer and the current value.

In order to adopt the Q-learning method, here are some key aspects need to be declared:

- **Environment**: is the system which user can select on the top k items which provided by the recommendation system.

- **State** $s \in \mathcal{S}$: will defined as the match from the recommended items and user's exactly choose, in simple it represent the value changed in the embedding matrix.

- **Action** $a \in \mathcal{A}$: is defined as a subset $\mathbb{A} \subset k$ which those k items is the possible experts/topics show to the user. Also, the $\mathcal{A} \in \begin{pmatrix} I_t \\ k \end{pmatrix}$ where the I_t is the whole possible topics/experts which may be recommended,the $\begin{pmatrix} I_t \\ k \end{pmatrix}$ means that we select top k items from the item-set I_t at timestamp t.

- **State Transition Probability** $P(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$: it corresponds to a user behavior a_t which will give a probability from current state s to next state s_{t+1} at the timestamp t.

- **Reward Function** $R(S) \in [0, 1]$: Unlike the normal RL method, we do not have a mapping function used for reward. The reward value used in our model is the accuracy between the decoded embedding and the original data as we are aiming to use the RL to improve our embedding.

- **Policy** $\pi(s)$: is defined as the strategy on how to optimize our embedding which generated by the auto-encoder.

- **Discount Factor** γ **and Learning Rate** η : $\eta, \gamma \in [0, 1]$ are the hype-parameters in this model, and need to be adjusted manually depending on the metrics in concern. $\gamma = 0$ means the long-term reward will not be considered AND only the current reward will be taken into account. On the contrary, $\gamma = 1$ means all the reward from previous can be fully considered in current state s.

The overview of a simple RL based recommendation system is (where the recommend agent represent the whole recommendation system):

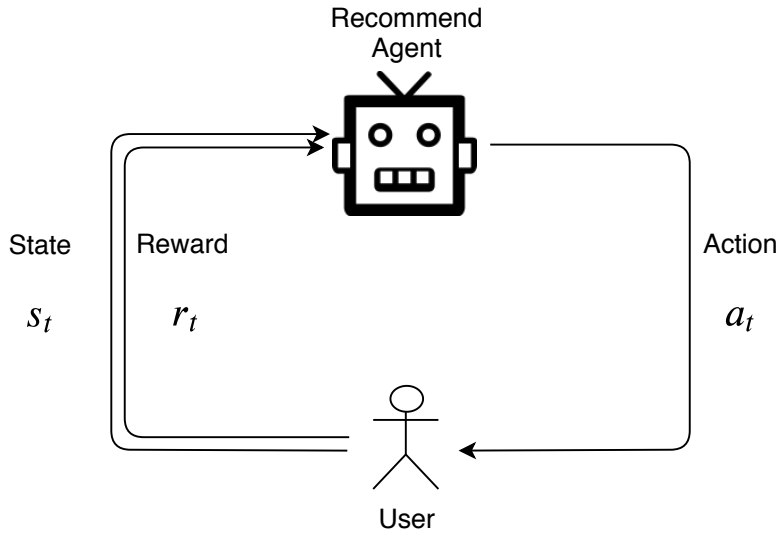


Figure 1.1: Flaw chart for RL

The above is the general aspect of the value-based RL method which is used in Recommendation system. There are two more methods in model-free RL area: Policy Based and Hybrid Method. The policy based methods aim to figure out the optimal policy π^* instead of maximizing the Q-value. The policy π is modeled with a parameterized function θ which can be express as: $\pi_\theta(a|s)$. The goal comes to find the best way to optimize θ so that the reward is the max [63]. The reward function can be defined as:

$$r(\theta) = \sum_{s \in \mathcal{S}} \mathbb{E}_\pi(s) V_\pi(s) = \sum_{s \in \mathcal{S}} \mathbb{E}_\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q_\theta(s,a)$$

where $\mathbb{E}_\pi$ is the distribution of the Markov Chain for π_θ which is the state distribution for policy π. The policy gradient aims to optimize $r(\theta)$ which is find the $\nabla_\theta r(\theta)$:

$$\nabla_\theta r(\theta) = \nabla_\theta V_\pi(s_r)$$

As the focus of the policy gradient is to work out the optimal policy, so the Q value function will be excluded. Assume the agent starts with a random state s_r, based on the policy gradient theorem, the final formula is:

$$\nabla_\theta r(\theta) = \mathbb{E}_\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q_\pi(s,a) = \mathbb{E}_\pi[Q_\pi(s,a) \nabla_\theta \ln \pi_\theta(a|s)]$$

Anther type of method is the hybrid method. The hybrid method adopt the benefit from value-based method and policy-based method. As in some case, the value based method may get stuck due to the max function. The most popular hybrid method is the actor-critic network. The critic network is the variation of the Q-learning. It uses the value based method to find the $Q(S_t, a_t)$ based on S_t. The actor network is to work out the policy parameter θ based on S_t. The critic network is used to guide actor network or measure the actor network. Actor work aims to find the best policy which can get the highest value from the critic network. Benefit from the hybrid methods, the policy search search comes to be possible [64].

All the above mentioned work assumes that the agent is fully observable, which is the definition of the MDP. However, in reality, we do not know user's next action, which means that the environment is only partially observable. In that case, it will be modelled as a Partially Observable Markov Decision Process (POMDP). Most of the POMDP components are similar to MDP excepts for two more factors that have to be considered for POMDP:

- Ω: the set of observations.
- O: the set of probabilities for the conditional observation Ω.

The updating process is slightly different from MDP. The updating process in POMDP is called the belief update by introducing the belief function $b(s)$ for the

state s. Assume the agent reached the state s, then agent observes $o \in O$ with the probability $O(o|s,a)$. The $b(\cdot)$ is a probability distribution over the state space $\mathcal{S}$, so the $b(s)$ denotes that the probability that current environment is state s, the updating process is:

$$b'(s) = \eta O(o|s,a) \sum_{s \in \mathcal{S}} P(s'|s,a) b(s)$$

$$\text{where } \eta = \frac{1}{\sum_{s' \in \mathcal{S}} O(o|s',a) \sum_{s \in \mathcal{S}} P(s'|s,a) b(s)}$$

Due to the uncertainty of the POMDP, there are a few works based on POMDP to make the recommendation [65]. Shang et al. mainly focuses on the environment reconstruction based on the POMDP by considering the hidden co-founders. This work provides a new prospective about how to model the interactive recommendation. The interactive recommendation is normally treated as the single-agent reinforcement learning which assumes the environment is static, but in the reality, it will be more suitable to build the system as the multi-agent reinforcement learning as the environment is dynamic as well.

1.3 Problem Statement

We now provide a problem statement for the interactive recommendation system. An interactive recommendation system features incorporating user's feedback dynamically during the training process. Given a set of users $\mathcal{U} = \{u, u_1, u_2, u_3, ...\}$ and a set of items $\mathcal{I} = \{i, i_1, i_2, i_3, ...\}$, the system first recommends item i_1 to user u_1 and then gets a feedback x. The system aims to incorporate feedback to improve future recommendations. To this end, it needs to figure out an optimal policy π^* regarding which item to recommend to the user to achieve positive feedback. We can formulate the problem as a Markov Decision Process (MDP) by treating the user as

the environment and the recommendation system as the agent. We define the key components of the MDP as follows (Chapter summarizes the main notations used in this paper):

- State: A state S_t is determined by the recent l items in which the user was interested before time t.
- Action: Action a_t represents a user's dynamic preference at time t as predicted by the agent.
- Reward: Once the agent chooses a suitable action a_t based on the current state S_t at time t, the user will receive the item recommended by the agent. The user's feedback on the recommended item (i.e., clicking the item, ignoring it) accounts for the reward $r(S_t, a_t)$, which will be considered to improve the recommendation policy π.
- Discount Factor γ: The discount factor $\gamma \in [0, 1]$ is used to balance between the future and immediate rewards—the agent will fully focus on the immediate reward when $\gamma = 0$ and take into account all the (immediate and future) rewards otherwise.
- State Transition: It's a probability of the transition from S_t to S_{t+1} based on the action a_t. This probability should in the range [0,1].
- Learning rate: η is the learning rate which used to let agent update its policy which should be very small like 0.005.

1.4 Existing Challenges

The first main challenge in the interactive recommendation systems is that the user's interest is evolving and dynamic. How to model this interest is a challenge. All the existing UCB based methods assume that the user's interest is static.

The second main challenge is that literature lacks RL based methods for interactive recommendation systems. The possible reasons could be: a) the interactive recommendation is a relative new topic. b) RL is a new technique in the recommendation system area when compared to the CF/MF based methods. c) training the RL will cost a significant amount of computational resource, which can be a hurdle for its applications.

Considering the evolving and dynamic nature of the user's interest, we choose the reinforcement learning to deal with the dynamic environment. As reinforcement learning has achieved impressive progress in learning representation [66], improving generative adversarial network [67] and so on. Reinforcement learning can be applied in many areas such as general game playing, recommendation system [12] and others. In this dissertation, we adopt the reinforcement learning as the main technique to make recommendation. The common reinforcement learning-based recommendation systems treat the recommendation system as sequential actions between the users and the system(agent) and try to figure out an optimal strategy to maximise the reward [68]. Different from them, we use reinforcement learning to determine the best policy to optimize our embedding instead of finding an optimal recommending strategy.

Furthermore, inspired by the knowledge-aware recommendation system and the recent advance in reinforcement learning, we propose the model KGRL and its im-

provement, i.e., RL-KGAN. Different from the KGRL, the RL-KGAN maintains a larger size of the Collaborative Knowledge Graph which has richer information than the normal knowledge graph. Considering the optimization and the training efficiency, we modify the experience replay into the sample efficient format to improve the sampling efficiency. The trust region policy optimization is used to limit the agent about updating policy. The CKG can provide a way to guide the searching and the interpretability as the knowledge graph provides the reasoning through the path [69, 70, 71, 72].

As the KGRL and RL-KGAN both use the knowledge graph to support the searching, considering the computational cost and storage cost, we adopt the Graph Neural Network(GNN) to refine the embedding of the KG. The GCN and GAT are widely used to capture the features from graph-like data [73, 74, 75, 76]. GNN shows the impressive performance in many areas like recommendation system, image classification, image generation [77], matrix completion [78], medical imaging [79, 80, 81] and others [82, 83, 84].

The structure of this dissertation is as follow. Firstly, we investigate the user's dynamic interest embedding methods and illustrate the new method Expert2Vec in Chapter 2, followed by a general solution for the interactive recommendation by adopting reinforcement learning and the knowledge graph in Chapter 3. Chapter 4 will discuss the improvement and the optimization direction of the proposed model. Chapter 5 includes the conclusion and the future directions about the interactive recommendation system. In Chapter 2, Chapter 3, and Chapter 4, the proposed methods will be detailedly illustrated with the experimental settings and the performance evaluation.

Chapter 2

Distributed Interaction Embedding

This section contains works published in: [1] **X. Chen**, C. Huang, X. Zhang, X. Wang, W. Liu, and L. Yao, "Expert2vec:Distributed expert representation learning in question answering community", in *International Conference on Advanced Data Mining and Applications*, pp. 281—301, November 2019. (CORE RANK:B)

2. Distributed Interaction Embedding

Expert Recommendation is the sub-area of the sequential recommendation. The expert recommendation face the similar challenges with the interactive recommendation which is the dynamic user's interest. In this chapter, we proposed a new embedding method which can be used for distributed interaction embedding. To be specific, we build a model which use the reinforcement learning to determine the best policy to optimize our embedding instead of finding a optimal recommending strategy where this embedding used to represent user's dynamic interest. The major contributions can be concluded as:

- We acquire some idea about the distributed representation from word2vec which is applied on the CQA problem as well as the expert recommendation. Based that, we propose a new distributed representation for user expertise which does not have many research before.
- Evolution on the big and complex data set - Stack Overflow where we got a acceptable result on several measure metrics among a few state-of-art models.
- We apply the reinforcement learning to improve the embedding so that it can get a better performance in recommendation.

The autoencoder shows the ability to capture the latent embedding for certain information in recent year [3, 85, 86]. Inspire by that, we adopted the variational autoencoder(VAE) [87] and the reinforcement learning to self-improve the learnt embedding. In this chapter, we will briefly illuminate our approach and the model structure. We will discuss the pre-processing step, how to obtain the original representation, how to get the embedding we are looking for, how to measure the accuracy between the embedding and the original representation, how reinforcement learning works in our model and how to used in a recommendation system.

2.1 Methodology

2.1.1 Pre-process of the data

The original dataset provided by Stack Overflow contain all questions and the corresponding answers which are plain text. We use the *SEWordSim* [88] which is a word similarity database for the Stack Overflow dataset which can make it more reliable and reduce some edge effects. In addition, we delete the questions which have zero response to overcome the cold-start problem for recommendation system. After deleting the unnecessary words, we extract all the users and users' answered questions and its corresponding vote received. Furthermore, the sentences are in higher dimension space, in order to reduce the dimension, we would convert all the topics and answers into vectors by using word2vec[89]. As the word2vec is a distributed representation of words which can retain the relation in the sentence, so we can use the vector form directly in the following steps.

2.1.2 Generate the User-Topic Matrix

After the pre-processing of the data, we get the formatted data which is needed for the matrix generation and the vector for each word. To keep all the information which is needed for ranking, we store all the voting and its corresponding user and topic together. As the topic is vector based and has high dimension which is hard to put it together with the topic information into the matrix. We build a hashing function $f : \mathbb{R} \mapsto \mathbb{R}$ which can simply map the topicID to its corresponding original vector, and we initialise a hash table to store all the mapping relations based on our hash function f. Thus our user-topic matrix can contain all the voting information

by ordering. If user don't have action with specific topic, the value of this cell with be null. In some cases the userId may not be continuous, and we convert all the userId with a list of continuous id so that it can easily determine the size. Also, it's easy to roll back to the original id. In addition, the userId is not important in the user-topic matrix as we only need to know the relationship between user and the topic.

The most important thing is how to rank the topic for each user. We cannot use the original voting information because less-popular questions may have very small view counts which can lead to a good answer receiving only one or zero vote. Also, the number of answers is varied for different questions which means we are unable to compare those answers through the same measurement metric due to the different number of competing answers. Therefore, we require a consistent measurement metric to help us determine the order, which means we need to make sure the votes in popular questions and less-popular questions are equivalent. To overcome this problem, we use the percentage vote(PVote) to compare answers, where we transfer all the votes receive for a certain answer j in a question q into a percentage mark:

$$\text{PVote}_q^j = \frac{V_\text{j}}{\sum_{i=0}^{n} V_i}$$

Where V_j means the vote that jth answer get on question q, n means number of answers we have on the question q. The denominator is the sum of all answers' vote. In addition, PVote can restrict the value in range $[0, 1]$ which means we do not need further processing or normalization. We do not need to consider about the case which denominator is zero as we delete all the topics which have zero response. Then, we convert all the topicId back to it's vector form as we need the topic information. After those operations we get a User-Topic Matrix $M : U \times \overrightarrow{T} \in \mathbb{R}^{T \times U}$, where U, T is the number of users and topics, $\overrightarrow{T}$ is the ranked topics.

2.1.3 AutoEncoder

As the user-topic matrix R has already been generated, the dimension of R is acceptable but the size $U \times T$ is relatively large. So we use the matrix R as the input of the variational autoencoder(VAE)[87]. Then, the VAE can learn a lower-dimension representation during the training which is the embedding E. The reason why we use the VAE is that the autoencoder is used widely on dimension reduction and features learning. Our user-topic matrix R have a high dimension topic embedding, we need to figure out a way to reduce the dimension and retain the necessary information to conduct analysis. The representation E we get from the autoencoder is the rough version of the embedding we are looking for. Then we need to calculate the similarity of the representation E. We pass the E into the decoder so that we can get a decoded matrix D_e which is supposedly the as same as the original matrix R. We use the "accuracy" to measure the similarity between D_e and R:

$$accuracy = \frac{\sum_i^n D_e^i \odot R^i}{n}$$

where n is the number of elements inside matrix R and D_e. The $\odot$ is the XNOR which used to calculate how many elements are exactly same, the D_e^i,R^i is the i-th element in matrix D_e and R. The XNOR operator has following property:

$$a \odot b = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a == b \end{cases}$$

2.1.4 Reinforcement Learning and Recommendation

As the accuracy was defined in previous section, we will use this accuracy as the reward in our reinforcement learning framework. We use the Q-learning here, the

training algorithm was described in the Algorithm 1. We will use the Q-learning to allow our model to improve the embedding E by itself. The n in algorithm refers to the number of episode. The strategy is to find a best direction of the value change in the embedding E which can acquire the highest Q-value. Once the optimal Q-value reached, it means we have figured out an optimal policy π^* which can improve our embedding representation E. Finally, we obtain an optimal embedding E^* which we can use in the recommendation system. In the real recommendation system,

ALGORITHM 1: Q-learning

Initialize Q-table,Q(s,a) randomly;

Initialize embedding E comes from the VAE;

Initialize $\eta \leftarrow \eta_{init}, \gamma \leftarrow \gamma_{init}$;

for $i = 0$ *to* n **do**

  Initialize s;

  $r =$ accuracy(E,R);

  for *each step in episode* i **do**

    choose a from s using policy derived from Q ;

    $Q(s_t, a) \leftarrow Q(s_t, a) + \eta(r + \gamma \max_a Q(s_{t+1}, a))$;

    use the q-value find the policy: $\pi \leftarrow Q(s, a)$;

    $s_t \leftarrow s_{t+1}$

  end

  use the policy update the embedding: $E \leftarrow E'$;

end

what we will use is the optimal embedding E^*. The embedding cannot be used for recommendation directly as the embedding does not have any valuable information for recommendation system. So, we need to recover the embedding E^*, through the decoder, into a matrix R' that contains the user-topic information and the

ranking information. The recommendation method we used is the collaborative filtering(CF). So, the overall structure for our proposed model in figure 2.1.

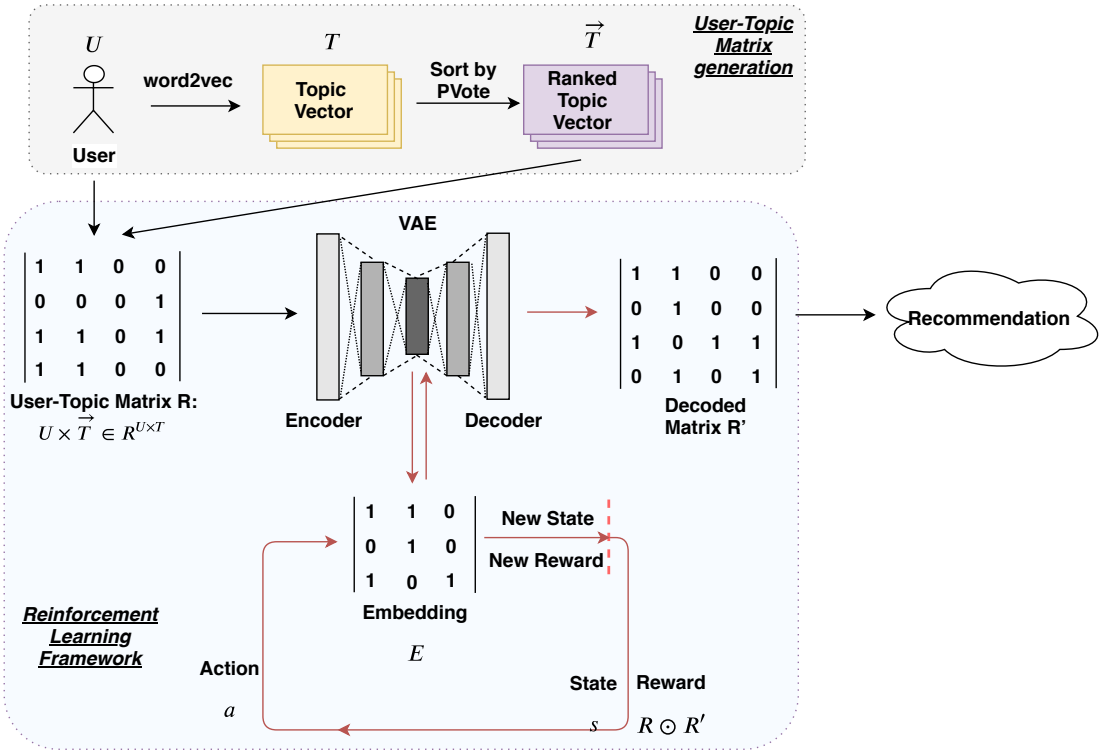


Figure 2.1: Model Structure,where the red line represent the work flow of RL. The new state is s_{t+1}, new reward is $R_{t+1} \odot R'$, t is the timestamp

2.2 Experiment

2.2.1 Experiment Setup

The data set used for experiment is the Stack Overflow which was flattened by removing all the XML markups and converted into the json format. The original data set contained 14,768,990 records including answers and questions. After filtering, the dataset was changed as 'userID:Topic' format that was described in section 4.1.

2. Distributed Interaction Embedding

After that, we had 99,220 users and 118,320 questions in total. As we conduce some data cleaning technique with the dataset, it leads to the userID not being consecutive as some users are considered inactive users. If we use the original userId as the axis it will make our matrix extremely big as the userID comes from 0 to 6,454,151, but we only need the 99,220 active users. Using the original userId as the index of matrix will get a matrix with shape [6454151,118320] which will take a huge amount of memory of computer. To overcome this problem, we replace the normal userId with our customised userId by using a hash table can map from [1,99220] to the [1,6454151]. By using the customised userId we can save $\frac{6454151-99220}{6454151} = 98.5\%$ runtime memory. So we have a user-topic matrix R which has shape [99,220,118,320] with the values 1, meaning have action, and 0, meaning no action. Also, topic we used in R is the topic id which is mapped as well(See section 4.2 for detail). The methodology we used for getting the vector is the word2vec, we use the pre-trained word2vec to transfer the topic into vector. What we do is that we firstly generate all single word's vector by using word2vec, so we get two lists which have word name and its vector. After that, the topic will be convert into vector with the dimension of 300. As the data is pretty big for training, to vertify the correctness of our approach, we just select top 20% of the samples from the dataset based on the reputation which still have over 2,000,000 records.

After finished the pre-process step, we just put the user-topic matrix R into the VAE to get a reconstructed representation E. To verify this representation is valid and have the necessary information we need, we recover it back to user-topic matrix R' by using the decoder. Then we calculate the accuracy between R and R' by using the formula mentioned in section 4.3. Then we put the embedding E and the accuracy R into the Q-learning framework to improve it. For each episode i, we take the improved embedding E_i and compare with the original matrix R to get the

new accuracy and transfer back to the Q-learning. After this optimize process, we passing the optimal embedding E' into a normal recommendation system. Then, using the Accuracy and the nDCG as the measure metric in our model where the accuracy is defined previously, and the nDCG is defined as:

$$nDCG_p = \frac{\sum_{i=1}^{p} x}{\sum_{i=1}^{|REL|} x} \text{ where } x = \frac{2^{rel_i} + 1}{\log_2(i+1)}$$

the rel_i is the real result which i supposed to be. We will use the nDCG@k, accuracy@k and the recall@k as the major measurement metric.

2.2.2 Experiment Results

As the expert recommendation is not a popular area, the state-of-art model is hard to figure out[90]. So, the baseline we used here is:

- Probabilistic matrix factorization(PMF)[91]: it's a probabilistic method which based on the matrix factorization to make the recommendation.
- Bayesian probabilistic matrix factorization(BPMF)[92]: it's a probabilistic model which combine the Bayesian classfier and matrix factorization method to make the recommendation.
- Segmented topic model(STM)[93]: it's state-of-arts model in the expert recommendation system area which consider about the topic embedding by segmentation.
- GRE4Rec[24]: it's a sequential recommendation system model whcih adopt the GRU to make the recommendation.

2. Distributed Interaction Embedding

- Adversarial Personalized Ranking for recommendation (APR)[94]: it's a model which adopt the adversarial training and make the personalized recommendation based on the sequential data.

The result can be found in figure 2.2.

2.2.3 Evaluation

It is obvious that when the accuracy increases the nDCG increases as well, which means the reinforcement learning is improving the embedding E by itself. However the nDCG@k is still not good enough which the highest value can reach 0.4767834. The reason is due to data sparsity. Even if the number of records are reduced and all the active users in the dataset are selected, it is still too sparse for the recommendation system to recommend a topic for a user. But we can see that our model is better than the others. The accuracy which is passed into the reinforcement learning is stable after a few episodes and it is stable at around 0.3, which means much information is lost during the encoding and decoding process. However, we still obtain a competitive result on the recommendation which means that our model meets the expectation. The caser is a state-of-art model which used in recommendation system area, it's sensitive with the sparsity data which we can find that the result is not good enough.

2.2.4 Discussion and future work

As discussed in section 2.1, we mentioned that in some questions they only have one answer which means user can only vote this answer or answer one. It may lead

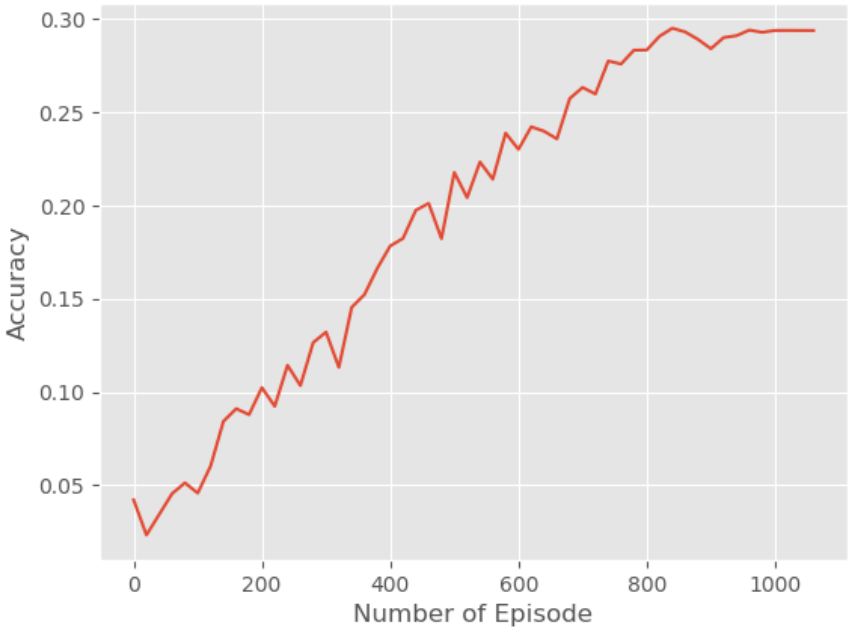
to some edge effects which will make the recommendation less efficient. Furthermore, due to the limitation of NLP technique, we may still lose information during the word2vec, and the answers are normally a paragraph which contains many sentences. We need a more efficient way to capture the relation between the word in word level and sentence level. That is the possible improvement can be done in the dataset side. From the model perspective, we can make some improvements in the recommendation system and the reinforcement learning system. For example, we can change the CF to the matrix factorization(MF) based recommendation system or a more complex model. But the challenge is that all the state-of-art recommendation systems are not working properly in CQA, in the future we may can adopt the state-of-art recommendation systems to the CQA problem so that it can make recommendation through our embedding.

As the neural network get some surprising result on reinforcement learning, we may change our reinforcement learning framework to deep reinforcement learning. One typical example is that change the Q-learning to DQN which discussed in section 3. But consider about the dataset's complexity, it will be tough to employ the complex recommendation system and the DQN framwork.

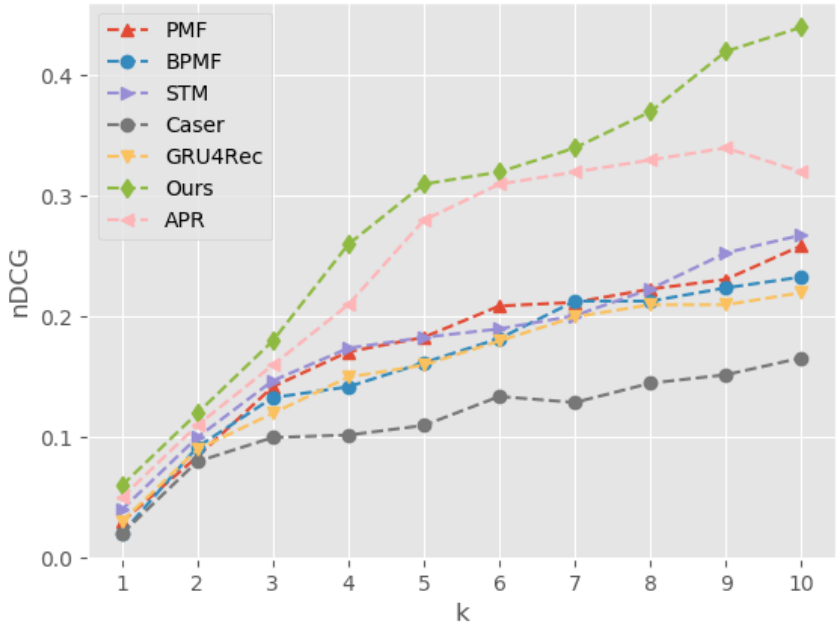
2.3 Conclusion

In this chapter we investigate a new distributed representation (expert2vec) for expert which is used on solving the CQA problem and the expert recommendation problem. Expert2vec is the distributed representation which contain the information about user and topic and its corresponding rank. We innovatively adopt the reinforcement learning framework into the expert recommendation problem to let the

model to improve the embedding by itself. Our model(Expert2Vec) got a promising result among the current expert recommendation state-of-art model.

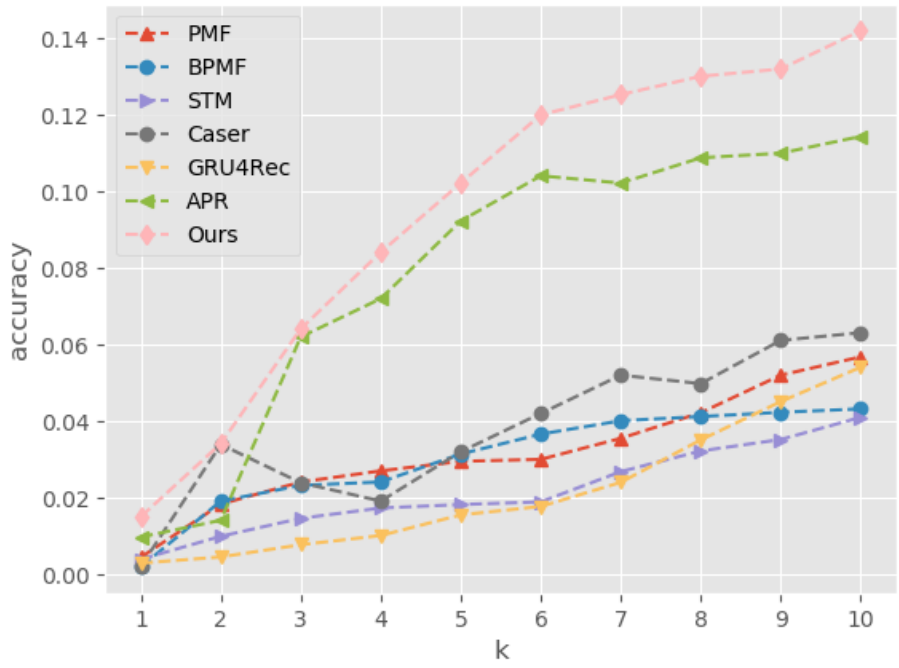


(a)

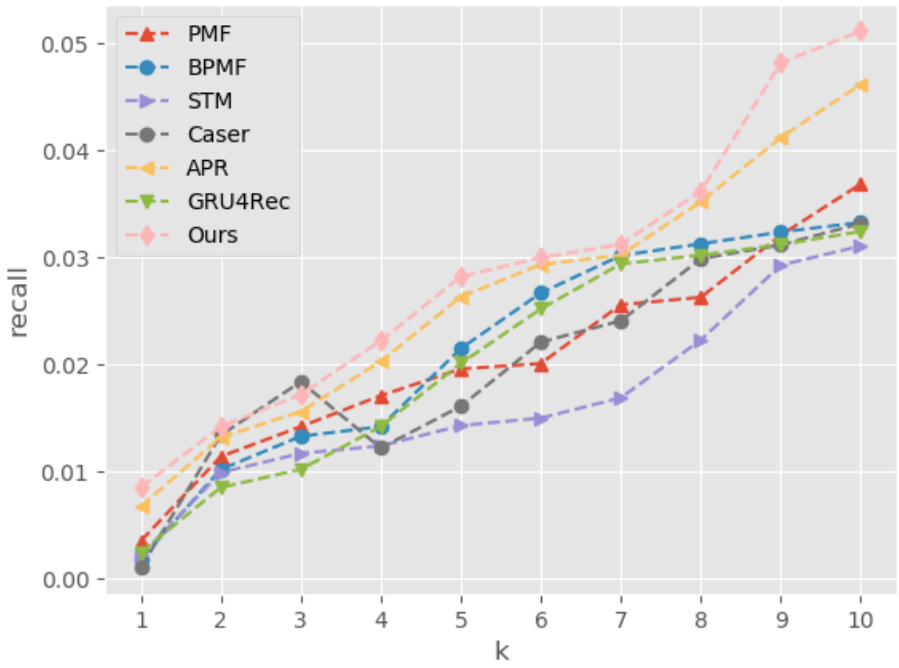


(b)

Figure 2.2: Graph (a) is the accuracy during the RL process.(b) is the model comparison result in nDCG



(a)



(b)

Figure 2.3: (a) is the comparison result in accuracy, (b) is the comparison result in recall.

Chapter 3

Side Information-augmented Interactive Recommendation

This section contains works accepted in: **X. Chen**, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, "Knowledge-guided deep reinforcement learning for interactive recommendation", in *International joint Conference on Neural Networks*,IEEE, 2020. (CORE RANK: A)

In the previous chapter, we investigate a new embedding method (Exper2Vec) which can capture user's dynamics interesting based on user's question and answering history. In this chapter, we will take a further step about the interactive recommendation system. To be specific, we consider about the side information like user's feedback about the recommended items, we adopt the actor-critic network to conduce the recommendation. The proposed model KGRL which can update itself dynamically based on the feedback and achieved a state-of-art performance. In summary, we make the following contributions in this work:

- We proposed a novel model KGRL where the knowledge graph is introduced into the reinforcement learning process to help the agent make decisions.
- To improve the efficiency, we maintain a local knowledge network which is based on the knowledge graph, to fasten the process while keeping the performance;
- Comprehensive experiments in the simulated online environment with six real-world datasets prove the performance of our propose approach.

3.1 Methodology

Our approach involves two steps: knowledge preparation and deep reinforcement recommendation

3.1.1 Knowledge Preparation

We construct the knowledge graph based on entity-relation-entity tuples $\{(i, r, j)|i, j \in \mathcal{E}, r \in \mathcal{R}\}$. For example, the tuple *(The Elements of Style, book.author, William Strunk Jr.)* means that *William Strunk Jr* authored the book *The Elements of Style*. We consider every item (e.g., *The Elements of Style*) as an entity in the knowledge graph $\mathcal{G}$ and transform the knowledge graph to represent user's preference more precisely [95]. Given a user $u \in \mathcal{U}$ and an item $q \in \mathcal{I}$, suppose $\mathcal{D}(i)$ is the set of items that has direct relationship with item i and r_{ij} denotes the relation between items i and j. We calculate the user-specific relation scores as follows:

$$f_u^{r_{ij}} = g(u, r_{ij}) \text{ where } g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

where g is a scoring function (e.g., inner product) to compute the score between user and relation; d is the dimension of user representation and relation representation; $u \in \mathbb{R}^d, r_{ij} \in \mathbb{R}^d$; $f_u^{r_{ij}}$ measures the importance of r_{ij} to user u.

Let $\mathcal{D}(i)$ be the set of candidates to recommend, we normalize the user-specific relation scores as follows:

$$\overline{f_u^{r_{ij}}} = \frac{f_u^{r_{id}}}{\sum_{d \in \mathcal{D}(i)} f_u^{r_{id}}} \in [0, 1]$$

Inspired by [96], we transform the knowledge graph into a user-specific graph A_u, which is an adjacency matrix of $\mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$. In this matrix, each position (i, j) corresponds to a score $\overline{f_u^{r_{ij}}}$, and a higher score indicates a stronger relation between two items i and j.

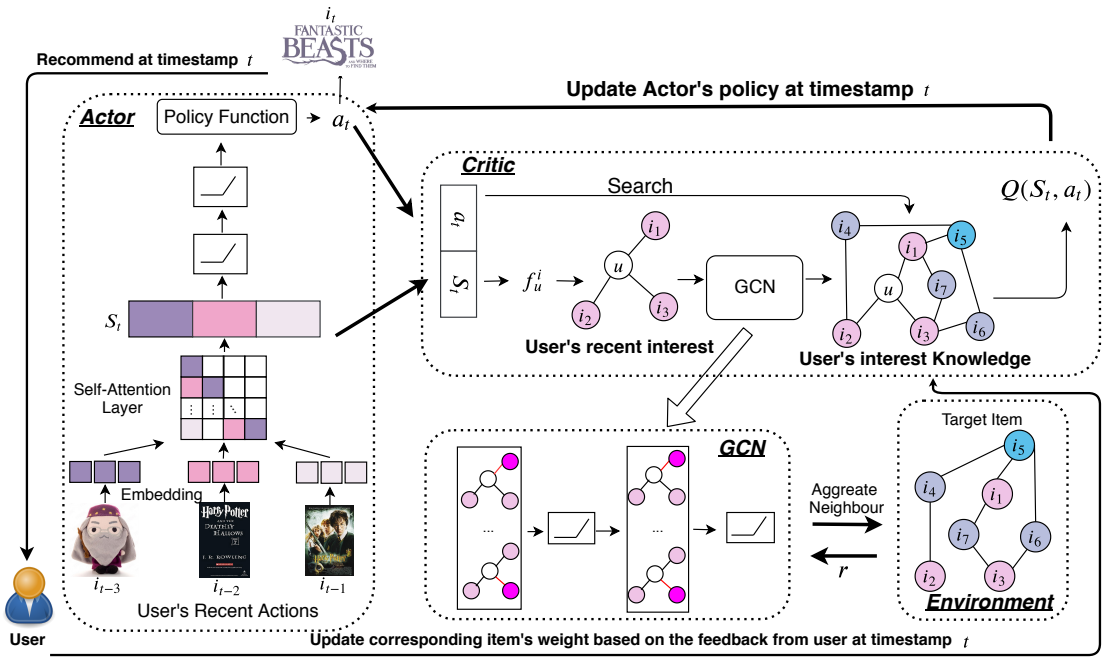


Figure 3.1: The KGRL structure. The left and right parts describe the actor network and the critic network, respectively, at time t. The model takes user's recent actions (regarding toys, books, and movies) as the input and recommends new items as the output. Those actions will be represented as the latent factor in this model. The user, in turn, provides feedback for the model to update user's interest knowledge's weights.

3.1.2 Deep Reinforced Recommendation

We develop our recommendation model (Figure 3.1) based on the Actor-Critic reinforcement learning framework [97], where the actor generates actions, the critic evaluates actions, and the actor network updates the policy based on the suggestion made by the critic.

Actor Network ϕ

Given a current state S_t, the actor network employs a neural network to infer an optimal policy π^* to work out an action a_t. Given S_t, which consists of user's recent

interests (shown in Figure 3.1, we first obtain vector representation of user's recent interest via embedding. Suppose we have a set of user'srecently interested items before time t, $\mathcal{S}_{u,t} = \{\mathcal{S}_u^1, \mathcal{S}_u^2, ..., \mathcal{S}_u^l\}$. The actor network takes an input sequence $\mathcal{S}_{u,t}$ and the corresponding feedback sequence $\{\mathcal{F}_u^1, \mathcal{F}_u^2, ..., \mathcal{F}_u^l\}$ to deliver an output sequence $\{\mathcal{S}_u^2, \mathcal{S}_u^3, ..., \mathcal{S}_u^{l+1}\}$. Given an original item embedding matrix $\mathcal{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$ (d is the dimension of the latent space), we apply positional embedding [98], $P \in \mathbb{R}^{n \times l}$, to preserve the order of user's previously interested items, which updates the item embedding into the following:

$$\mathbb{E} = \begin{bmatrix} \mathcal{M}_1 + P_1 \\ \mathcal{M}_2 + P_2 \\ \cdots \\ \mathcal{M}_l + P_l \end{bmatrix}$$

We then fed this embedding into a self-attention layer to reduce impurity in the embedding [99]. The layer uses the scaled dot-product attention [100], which is originally defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where Q, K, V denotes queries, keys, and values, respectively; $\sqrt{d_k}$ is the scaling factor to regulate the value range of QK^T. After applying the embedding $\mathbb{E}$ as the input, the attention turns into the following:

$$\text{Attention}(\mathbb{E}W^Q, \mathbb{E}W^K, \mathbb{E}W^V)$$

where $\mathbb{E}W^Q, \mathbb{E}W^K, \mathbb{E}W^V \in \mathbb{R}^{d \times d}$. We fed this embedding into two fully connected layers, which use ReLU and tanh as the activation functions, respectively as described in [98]. The output of the attention layer is the state S_t at time t.

Critic Network ψ

We design the critic network to estimate the Q-value function $Q(S_t, a_t)$ to evaluate actor's policy. The critic network takes state representation S_t and action representation a_t as the input (shown in Figure 3.1). We design a local knowledge network within the critic network to capture the high-order structural proximity among the items in the knowledge graph using graph convolutional network (GCN). Specifically, given a user-specific graph g_i^u generated from the current state S_t, we feed it into a two-layer GCN that applies the following layer-wide propagation rule:

$$H^{l+1} = \sigma(D^{-\frac{1}{2}}\hat{A}_u D^{-\frac{1}{2}} H^l W^l) \tag{3.1}$$

where H^{l+1} is the representation of entities at layer $l+1$; A_u is the input matrix that aggregates the neighbour's entities; $\hat{A}_u$ is set to $A_u + I$, where the I is an identity matrix used to avoid negligence of the old representation via self-connection; D_u is the diagonal degree matrix for $\hat{A}_u$ where $D_u^{ii} = \sum_j \hat{A}_u^{ij}$ (the symmetric normalization was applied to keep the representation H^l stable, as denoted by $D^{-\frac{1}{2}}\hat{A}_u D^{-\frac{1}{2}}$); W^l is the weight matrix for layer l; and $\sigma(\cdot)$ denotes the non-linear activation function.

Recent research shows the feasibility of searching in graphs processed by GCN [101]. Since GCN capture's all the structural information in the knowledge graph, it will not affect the search results. In this study, we assume an unweighted graph where a user is equally interested in every item. Then, we start searching with the actor predicted action a_t (i.e., predicted item i_p) to the real target i_t, based on the user's personalized interest knowledge (i.e., trained graph with all parameters θ_{kg}). Finally, we calculate the Q value by estimating the reward r based on the distance between the predicted item and the target item:

$$r = \frac{100}{\sqrt{\text{Distance}(i_p, i_t) + \epsilon}} * W_{pt}$$

where W_{pt} is the sum of weight of the shortest path from i_p to i_t; ϵ is the parameter to avoid the denominator becoming 0. We calculate the distance using the Dijkstra's algorithm with MinHeap.

3.1.3 Complexity Analysis

We analyze the time and space complexity of the critic network, especially the search part, in this section. We consider a vector composition (i.e., the combination of the state vector and action vector) and assume the transmission time as a constant c. Given a user interested in I_u items, we consider the worst case—a complete graph and each item i having M nearest non-duplicate neighbours. Thus, we get a graph with $I_u + I_u M$ nodes (exclude the centralised user node) and $(I_u + I_u M)(I_u + I_u M - 1)/2$ edges. We then calculate the time and space complexity as $\mathcal{O}((|I_u + I_u M)^2 + |I_u + I_u M| \log |I_u + I_u M|) \sim \mathcal{O}(|I_u + I_u M|^2)$ and $\mathcal{O}(2|I_u + I_u M|) \sim \mathcal{O}(|I_u + I_u M|)$. In comparison, if we feed the environment knowledge graph to the critic network directly, the time and space complexity would be $\mathcal{O}(|I + IM|^2)$ and $\mathcal{O}(|I + IM|)$. Apparently, the local knowledge network significantly improves the performance and saves the memory space in our model ($I_u \ll I$). Moreover, the local knowledge network is easier to converge as it has fewer nodes than the environment knowledge graph.

3.1.4 Training Strategy

Training the actor-critic network requires train two parts of the neural network simultaneously. We apply the Deep Deterministic Policy Gradient (DDPG) (Algorithm 1) to train our model [50], where we train the critic by minimising a loss

function:

$$l(\theta_\psi) = \frac{1}{N}\sum_{j=1}^{N}((r + \gamma\xi) - \psi_{\theta_\psi}(S_t, a_t))^2$$

$$\text{where } \xi = \psi_{\theta'_\psi}(S_{t+1}, \phi_{\theta'_\phi}(S_{t+1}))$$

where θ_ψ is the parameter in critic; θ_ϕ is the parameter in actor; N is the size of mini-batch from the replay buffer; $\psi_{\theta'_\psi}$ and $\phi_{\theta'_\phi}$ are the target critic and target actor network, respectively. Algorithm 2 describes the training of the local knowledge network, where we define the same loss function for all users for the local knowledge network :

$$l_k = \sum_{u\in\mathcal{U}}(\sum_{i:y_{ui}} J(y_{ui}, \hat{y}_{ui}))$$

where J is the cross-entropy; y_{ui} is a piece-wise function to reflect the interest/action (defined below):

$$y_{ui} = \begin{cases} 1 & \text{if u interested in i} \\ 0 & \text{otherwise} \end{cases}$$

3.2 Experiments

In this section, we report our experimental evaluation of our model in comparison with several state-of-the-art models using real-world datasets.

3.2.1 Datasets

We conducted experiments on six public real-world datasets (Table 3.1 shows the statistics). All these datasets provide the necessary information for building the

respective knowledge graphs.

Book-Crossing[1]: This dataset contains user's demographic information and book information from the Book-Crossing community. It is extremely sparse with a density of 0.0041%.

MovieLens-20M[2]: This is a well-known benchmark dataset that contains 20 million ratings from around 140 thousand users on the MovieLens website. It also provides movie tags, which can be used to build relations in the knowledge graph.

Librarything[3]: This dataset contains book review information collected from the librarything website.

Amazon CDs and Vinyl[4]: This is a highly sparse dataset that contains the product metadata, user reviews, ratings, and item relations, as part of the Amazon e-commence dataset.

Netflix Prize[5]: This dataset contains 100 million ratings from 480 thousand users and item information for yearly open competition to improve Netflix's recommendation performance.

Goodreads[6]: This dataset contains user's ratings and reviews to books on the Goodreads book review website.

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

[2] https://grouplens.org/datasets/movielens/

[3] http://cseweb.ucsd.edu/~jmcauley/datasets.html#social_data

[4] http://jmcauley.ucsd.edu/data/amazon/

[5] https://www.kaggle.com/netflix-inc/netflix-prize-data

[6] http://cseweb.ucsd.edu/~jmcauley/datasets.html#goodreads

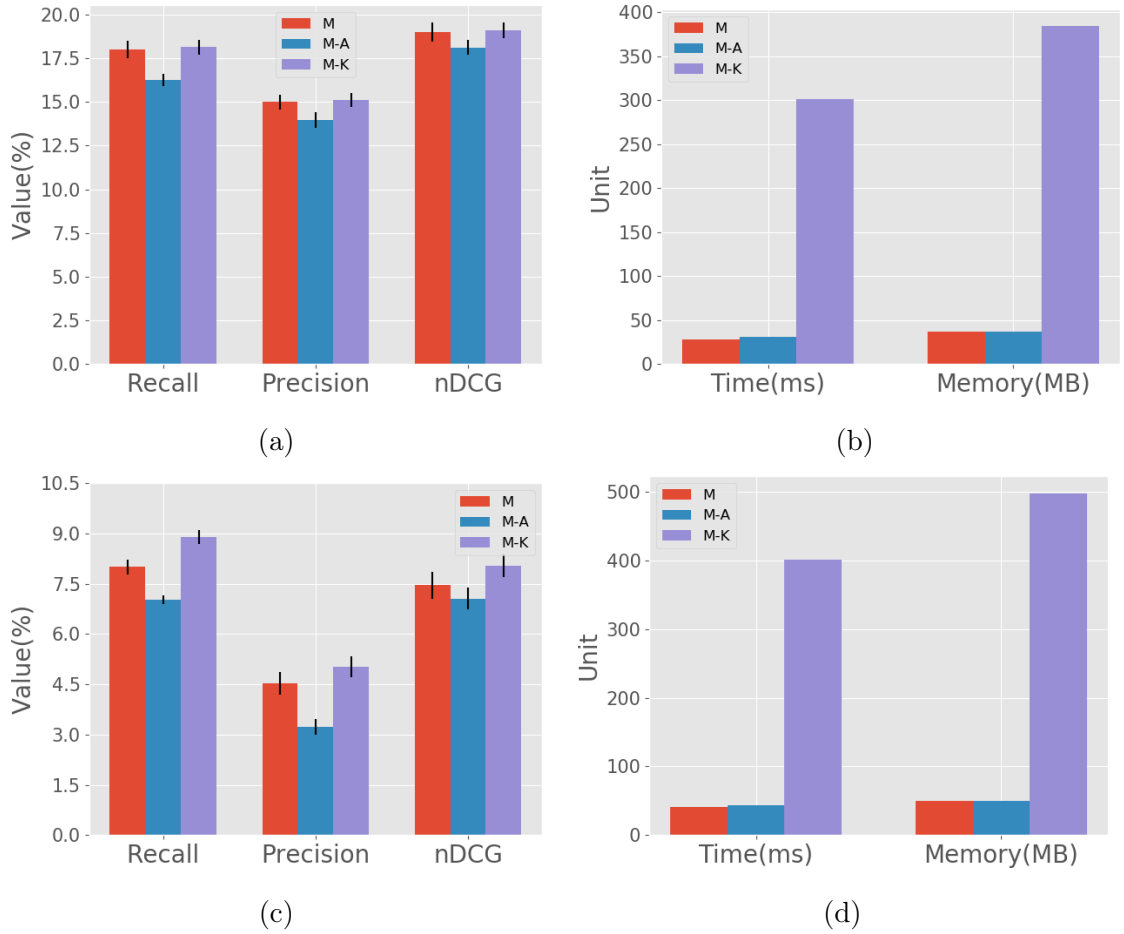
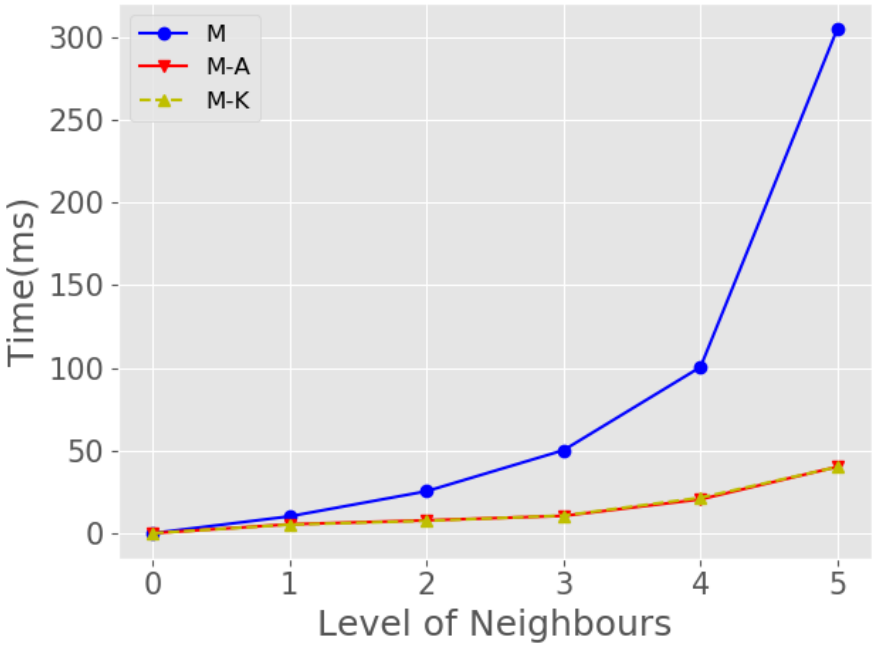
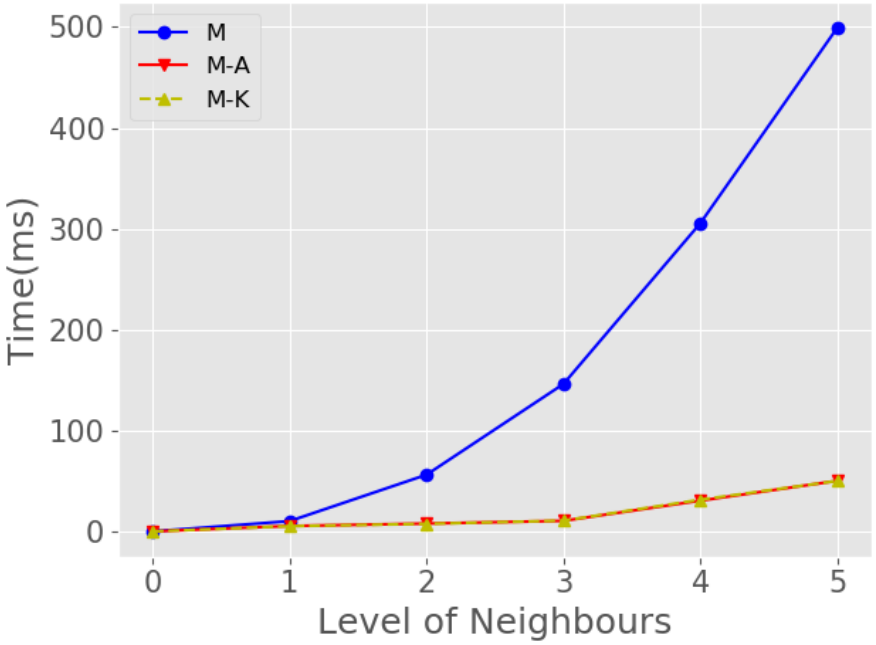


Figure 3.2: Ablation and complexity studies.

The figure 3.2 shows the ablation and complexity studies on MovieLens-20M(a,b,e) and Book-Crossing(c,d,f): (a,c) Three models' performance in Recall, Precision, and nDCG; (b,d) Three models' time and memory consumption in conducting search for a target item located among fifth level neighbours. The figure 3.3 (a,b) are the three models' time consumption along with an increasing level of the target item. *M* denotes our original model, *M-A* the model without the attention layer, meaning the item embedding will directly goes to state, and *M-K* the model deprived of the local knowledge network—in this case, the model uses GCN to learn the whole environment inside itself. The line of *M-A* and *M-K* have the very similar trend,



(a)



(b)

Figure 3.3: Time Comparison for the ablation study

consider about that case, we put the high resolution image for this part which would easy to distinguish. The level of neighbours represents the geographical location indicative of the shortest distance. For example, first-level neighbours represent the items which have a distance of 1 to the current item i.

3.2.2 Evaluation Metrics

We evaluate the performance of recommendation using three metrics: precision, recall, and normalized Discounted Cumulative Gain (nDCG). All the metrics were calculated based on the top-10 recommendations to each user for each test case. To ease processing, we removed users who have fewer than ten interactions and scaled the ratings from all datasets to the range of $[0, 5]$. Only the items with a rating score higher than three were considered a relevant item.

3.2.3 Experimental Setup

We evaluated our model in a simulated online environment built upon offline public datasets, using the algorithm proposed in [15] and the aforementioned reward function. This way, we avoided collecting private user information and expensive online training [102]. Specifically, the simulator generated feedback based on logistic matrix factorization (LMF) [103]. We randomly split each dataset into a training set (70%), a validation set (10%), and a testing set (20%) to conduct 10-fold cross-validation. The discount factor γ was initialized to 0.99.

3.2.4 Compared Methods

We compared out model with several competitive baselines:

Policy-Guided Path Reasoning (PGPR) [18]: A state-of-the-art knowledge-aware model that employs reinforcement learning for explainable recommendation.

Tree-structured Policy Gradient Recommendation (TPGR) [104]: A state-of-the-art model that uses reinforcement learning and binary tree for large-scale interactive recommendation.

HLinearUCB [9]: A contextual-bandit approach that learns extra hidden features for each arm to model the reward for interactive recommendation.

Wolpertinger [12]: A deep reinforcement learning framework that uses DDPG and KNN for recommendations in large discrete action spaces.

DeepPage [13]: A DDPG-based reinforcement learning model that learns a ranking vector for page-wise recommendation.

DRN [14]: A DQN-based recommendation method that employ deep Q learning to estimate Q-value for news recommendation.

FactorUCB [40]: A matrix factorization-based bandit algorithm for interactive recommendation .

ICTRUCB [11]: A MAB approach that uses a depend arm for online interactive collaborative filtering.

3.2.5 Results

Table 3.2 shows our evaluation results of recommendation models. We observed that our model outperformed all the baselines in all metrics almost on all the datasets—it performed only slightly worse than TPGR on the Book-Crossing dataset. This may be attributed to the specifical design of TPGR to deal with large-scale datasets. None of the MAB-based methods (HLinearUCB, FactorUCB and ICTRUCB) performed well on those datasets because they all assume static user interest and may not give up-to-date recommendations We also observed that PGPR performed worse than DRN on the Amazon CD and Book-Crossing datasets—these sparse datasets might not provide sufficient relation for PGPR to infer the recommendation path. Finally, all the models achieved their best results on the MovieLens-20M dataset, given the rich information and dense relation in the dataset.

3.2.6 Ablation and Complexity Studies

We conducted ablation studies to explore the impact of the attention mechanism and local knowledge network on the performance of our model on the above six datasets. We selectively choose MovieLens-20M and the Book-Crossing as the example because the Book-Crossing dataset is the most sparse one and the MovieLens-20M is the most dense one; they can show the capability of our model in the normal case and extreme case. Due to the exponential increase in time usage, we only show the first five level of neighbours. The results (Figure 3.2(a,d)) show that our model's performance dropped slightly (by 1% in precision, 2% in recall, and 1% in nDCG) without the attention mechanism while elevated slightly without the local knowledge network because the model already contains all the information, including abundant

relation between items to support the decision making.

We also used valgrind[7] to monitor the memory usage, which, on the other hand, reveals the huge advantages of using a local knowledge network in reducing both the time and space complexity (also see Figure 3.2(b)). We mentioned that in figure 3.2 (c,f), the model $M - K$ have an incredible increase in time consumption when the level goes over 2. One possible reason is that as the level goes higher, the graph comes more and more complex, which will affect the search critically.

3.3 Conclusion

In this chapter, we have proposed a knowledge-guided deep reinforcement learning framework (KGRL) for interactive recommendation. KGRL uses the critic-actor learning framework to harness the interaction between users and the recommendation system and employs a local knowledge network to improve the stability and quality of the critic network for better decision-making. Extensive experiments over an online simulator with six public real-world datasets demonstrate its superior performance over state-of-the-art models. To verify the effectiveness for each component, we conduct the ablation study for the local knowledge network and attention mechanism and selectively present the performance both in normal case and extreme case.

We are planning to introduce various types of user information (e.g., user's thought when browsing items) to enrich the interaction and deploy our model in online business platforms to further test the performance in the future. In addition, the cold-start problem is another big challenge to be focused on. Besides, the algorithm

[7] `http://www.valgrind.org/`

2 used to train the model still lacks the knowledge about how to update step size will affect the training time and the convergence which can be solved in the future work.

ALGORITHM 2: DDPG algorithm for our model

Initialize actor network ϕ with parameter θ_ϕ and critic network ψ with parameter θ_ψ randomly;

Initialize target network ϕ' and ψ' with weight $\theta'_\phi \leftarrow \theta_\phi$, $\theta'_\psi \leftarrow \theta_\psi$;

Initialize the local knowledge network ;

Initialize Replay Buffer $\mathcal{B}$;

for $i = 0$ *to* n **do**

  Receive the initial state S_i ;

  for $t = 1$ *to* T **do**

    Infer a action a_t according to the $\phi(\cdot)$;

    Execute the action a_t to receive a reward r_t and observe a new state S_{t+1};

    $\mathcal{B}$.append(S_t, a_t, r_t, S_{t+1}) ;

    Sample a random mini-batch of $\mathcal{N}$ transitions (S_k, a_k, r_k, S_{k+1}) from $\mathcal{B}$;

    Set $y_i = r_t + \gamma\xi$;

    Update Critic by minimise the loss $l(\theta_\psi)$;

    Update local knowledge net by Algorithm 3 ;

    Update the Actor policy by using the sampled policy gradient:

    $\nabla_{\theta_\phi}\phi = \frac{1}{N}\sum_{j=1}^{N} \nabla_a \psi(S_k, a)|_{a=\phi(S_k)} \nabla_{\theta_\phi}\phi(S_k)$;

    Update target network:

    $\theta'_\phi \leftarrow \tau\theta_\phi + (1-\tau)\theta'_\phi$;

    $\theta'_\psi \leftarrow \tau\theta_\psi + (1-\tau)\theta'_\psi$;

  end

end

ALGORITHM 3: Training the local knowledge network

input: The user specific graph g_i^u, environment KG $\mathcal{G}_e$

Initialize the parameters for GCN θ ;

Initialize the depth of graph d_g ;

Initialize the reward storage P;

for *i in* g_i^u **do**

    Receive the reward r from $\mathcal{G}_e$;

    P.append(r);

end

$r = \min(\mathrm{P})$;

while *GCN is not converge* **do**

    if $d_g < r$ **then**

        aggregate next level's neighbours into g_i^u $d_g \leftarrow d_g + 1$;

    end

    Update the GCN and its corresponding θ;

end

Table 3.1: Statistics of our experimental datasets

Dataset	# of users	# of items	# of interactions
Amazon CD	75,258	64,443	3,749,004
Librarything	73,882	337,561	979,053
Book-Crossing	278,858	271,379	1,149,780
GoodReads	808,749	1,561,465	225,394,930
MovieLens-20M	138,493	27,278	20,000,263
Netflix	480,189	17,770	100,498,277

Table 3.2: The overall results of our model comparison with several state-of-arts models in different datasets. The result was reported by using the percentage and based on top-10 recommendation as mentioned before. The highlighted result in bold is the best result.

Dataset	Amazon CD			Librarything		
Measure (%)	Recall	Precision	nDCG	Recall	Precision	nDCG
Wolpertinger	1.542 ± 0.192	1.521 ± 0.145	3.331 ± 0.201	3.441 ± 0.313	3.673 ± 0.221	4.115± 0.251
HLinearUCB	3.112 ± 0.331	2.647 ± 0.171	4.005 ± 0.341	8.102 ± 0.396	7.431 ± 0.204	8.157 ± 0.241
FactorUCB	3.531 ± 0.232	4.512 ± 0.242	6.012 ± 0.251	8.541 ± 0.241	8.162 ± 0.355	8.653 ± 0.351
ICTRUCB	4.124 ± 0.293	3.110 ± 0.395	5.982 ± 0.602	9.201 ± 0.241	7.980 ± 0.151	8.012 ± 0.466
DeepPage	7.124 ± 0.181	4.127 ± 0.134	7.245 ± 0.154	10.342 ± 0.422	9.012 ± 0.241	9.124 ± 0.673
DRN	8.006 ± 0.232	4.234 ± 0.241	6.112 ± 0.241	10.841 ± 0.112	9.412 ± 0.242	9.527 ± 0.455
TPGR	7.294 ± 0.312	2.872 ± 0.531	6.128 ± 0.541	14.713 ± 0.644	12.410 ± 0.612	13.225 ± 0.722
PGPR	6.619 ± 0.123	1.892 ± 0.143	5.970 ± 0.131	11.531 ± 0.241	10.333 ± 0.341	12.641 ± 0.442
Ours	**8.208 ± 0.241**	**4.782 ± 0.341**	**7.876 ± 0.511**	**15.128 ± 0.241**	**12.451 ± 0.242**	**14.985± 0.252**
Dataset	Book-Crossing			GoodReads		
Measure (%)	Recall	Precision	nDCG	Recall	Precision	nDCG
Wolpertinger	0.782 ± 0.121	1.235 ± 0.131	0.976 ± 0.242	6.245 ± 0.122	3.415 ± 0.207	5.315 ± 0.321
HLinearUCB	2.421 ± 0.131	1.724 ± 0.141	2.865 ± 0.322	7.917 ± 0.303	5.151 ± 0.214	6.561 ± 0.351
FactorUCB	3.123 ± 0.141	2.976 ± 0.223	3.536 ± 0.241	5.643 ± 0.441	4.129 ± 0.221	6.122 ± 0.395
ICTRUCB	3.441 ± 0.121	3.421 ± 0.333	4.001 ± 0.321	8.415 ± 0.132	6.432 ± 0.221	7.124 ± 0.241
DeepPage	5.124 ± 0.323	3.245 ± 0.142	6.976 ± 0.142	10.071 ± 0.212	7.961 ± 0.232	8.329 ± 0.232
DRN	7.124 ± 0.122	4.123 ± 0.112	7.433 ± 0.142	10.620 ± 0.123	8.432 ± 0.241	9.461 ± 0.442
TPGR	7.246 ± 0.321	**4.523 ± 0.442**	**7.870 ± 0.412**	13.219 ± 0.323	10.322 ± 0.442	9.825 ± 0.642
PGPR	6.998 ± 0.112	3.932 ± 0.121	7.333 ± 0.133	11.421 ± 0.223	10.042 ± 0.212	9.234 ± 0.242
Ours	**8.004 ± 0.223**	4.521 ± 0.332	7.459 ± 0.401	**13.444 ± 0.321**	**10.331 ± 0.331**	**11.641 ± 0.446**
Dataset	MovieLens-20M			Netflix		
Measure (%)	Recall	Precision	nDCG	Recall	Precision	nDCG
Wolpertinger	7.821 ± 0.171	2.341 ± 0.142	4.002 ± 0.151	3.924 ± 0.222	2.911 ± 0.141	3.425 ± 0.261
HLinearUCB	13.591 ± 0.281	10.601 ± 0.132	12.537 ± 0.285	5.142 ± 0.314	5.052 ± 0.362	6.007 ± 0.425
FactorUCB	14.421 ± 0.412	11.229 ± 0.365	11.422 ± 0.611	5.643 ± 0.432	4.129 ± 0.233	6.122 ± 0.442
ICTRUCB	14.345 ± 0.212	9.923 ± 0.222	11.051 ± 0.423	7.00 1± 0.312	6.212 ± 0.432	9.112 ± 0.523
DeepPage	12.472 ± 0.312	10.161 ± 0.332	13.129 ± 0.322	8.431 ± 0.212	7.324 ± 0.133	9.872 ± 0.223
DRN	14.742 ± 0.223	14.092 ± 0.342	16.245 ± 0.242	12.310 ± 0.144	10.213 ± 0.142	16.562 ± 0.153
TPGR	16.431 ± 0.369	13.421 ± 0.257	18.512 ± 0.484	12.512 ± 0.556	11.512 ± 0.595	17.425 ± 0.602
PGPR	14.234 ± 0.207	9.531 ± 0.219	11.561 ± 0.228	10.982 ± 0.181	10.123 ± 0.227	17.134 ± 0.243
Ours	**18.021 ± 0.498**	**14.989 ± 0.432**	**19.007 ± 0.543**	**13.009 ± 0.343**	**11.874 ± 0.232**	**19.082 ± 0.348**

Chapter 4

Structural Interactive Recommendation

This section contains works submitted to: **X. Chen**, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, "Reinforced Graph Attention Network for Interactive Recommendation", in *29th ACM International Conference on Information and Knowledge Management*, 2020. (CORE RANK: A)

In the previous chapter, we investigate a new RL based method for interactive recommendation (KGRL) which use the actor-critic structure to empower the interactive recommendation system. However, the proposed method still lack some technique to boost the converge of the RL and limit the update policy's step size. In this chapter, we will investigate this problem.

4.1 Methodology

Our proposed approach, RL-KGAN, employs the actor-critic algorithm to support deep reinforcement learning. Figure 4.1 shows the overall framework.

4.1.1 User's Embedding

Given a current state S_t, the actor network employs a neural network to infer an optimal policy π^* and to work out an action a_t. We use user's embedding to represent state S_t.

Inspired by [105], we model the embedding trajectories of user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ as a sequence of temporal user-item interactions $I_t = (u_t, i_t, f_t)$, where t denotes the timestamp, f_t denotes user's feedback. For each user, we maintain two embeddings that used to represent the recent interest and long-term interest, respectively. Given a user u, the long-term interest embedding, $\overline{E_u}$, is relatively stable and remain unchanged over time.

Then, we employ an RNN layer to capture the user's recent dynamic interest E_u.

Finally, we concatenate the embedding $\overline{E_u}$ and E_u as the input and work out the

policy function $\pi(S_t)$ using two fully connected layers with ReLU as the activation function.

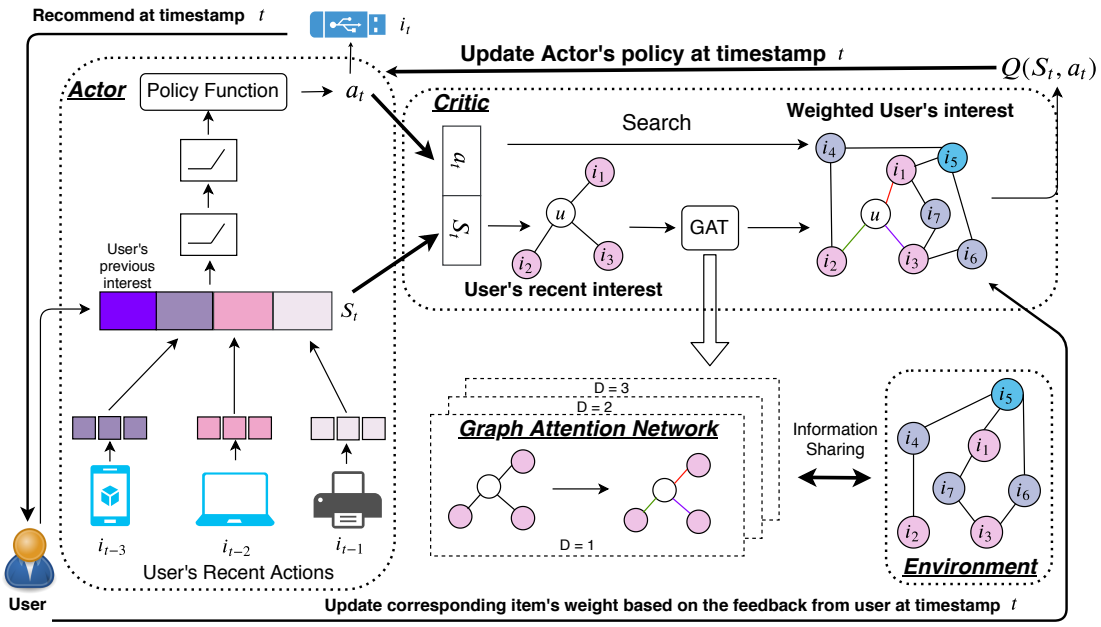


Figure 4.1: The proposed model structure. The left and right parts describe the actor network and the critic network, respectively, at time t. The model takes user's recent actions (regarding toys, books, and movies) which will be considered as the Embedding E_u, and the user's long-term embedding $\overline{E}_u$ will be the output of an LSTM layer. The user, in turn, provides feedback for the model to update the user's interest knowledge's weights.

4.1.2 Policy Update

We design a critic network with a domain knowledge graph to evaluate the actor's policy. The critic network takes state S_t and action a_t as the input to supervise the actor network. User interest may change over time, leading to differed importance of relations in the graph. Therefore, we employ a graph attention network to embeds user's behavior and item's property, to generate weights for the relations [61]. Let $\{(i, r, j)|i, j \in \mathcal{E}, r \in \mathcal{R}\}$ be a entity-relation-entity tuple, we use all the entity-

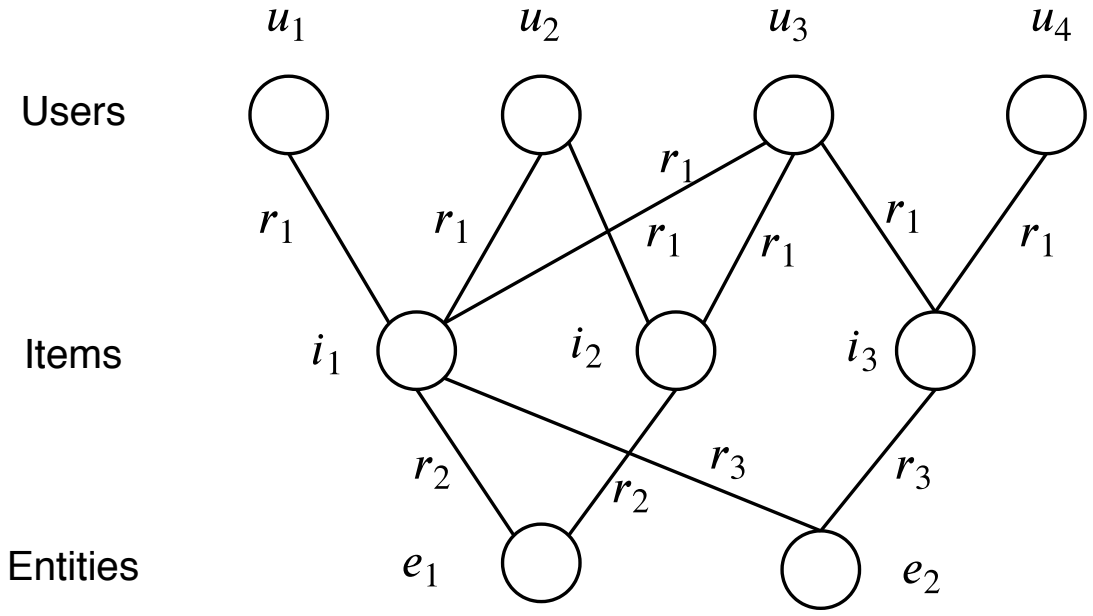


Figure 4.2: CKG

relation-entity tuples to construct the graph, $\mathcal{G} : (\mathcal{E}, \mathcal{R})$. Then, we define the concept as a collaborative knowledge graph. Consider a new triplet $\{(i, r, j, f)|r \in \mathcal{R}, i, j \in \mathcal{E}\}$, where the f represent the feedback from user u to the relation r. Suppose a user u_1 purchase/interact with item i_1 previously, we figure out a path in knowledge graph for u_1: $u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1 \xrightarrow{r_2} i_2$. The path indicates a share entity (e_1) between items i_1 and i_2. We employ Graph Attention Network (GAT) to capture such high-level relations and to map those relations $\mathcal{R}$ into weight W, to indicate their importance. In knowledge graph, one entity can be involved in several triplets which play as the bridge to connect different triplets and share information. Consider about two examples for user u_1: $u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1$ and $u_1 \xrightarrow{r1} i_2 \xrightarrow{r_2} e_2$, the item i_1 have two different entities e_1, e_2 which means those to entitles need to be used to enrich the item features, then it can help to understand u_1's preference better. This process can be replaced by the information propagation between u_1 and e_1, e_2. To make this process simpler, we want to propagation between entities and its neighbors. Given

a triplet $T = \{(i, r, j)|(i, r, j) \in \mathcal{G}, i, j \in \mathcal{E}, r \in \mathcal{R}\}$,

we consider this triplet as an ego-network [106] and use the liner combination to generate its representation l_i:

$$l_i = \sum_{(i,r,j)\in T} d(i, r, j)\mathbf{e}_j$$

where $d(i, r, j)$ is the decay factor that controls the propagation decay through the path (i, r, j); it represents the amount of information transferred through relation r from entity i to entity j.

We distinguish the importance of relations by control the decay factor $d(i, r, j)$:

$$d(i, r, j) = a(\mathbf{W}_r\mathbf{e}_i, \mathbf{W}_r\mathbf{e}_j)$$

where a is the shared attention mechanism $a : \mathbb{R}^F \times \mathbb{R}^F = \mathbb{R}$; F is number of features; $\mathbf{e}$ is the entity representation; $\mathbf{W}$ is an affine transformation with the weight matrix W for relation r; $d(i, r, j)$ is the importance about entity j to entity i through relation r. In some situation that every node may have relation with every other node, and the attention score may lie in different range which is hard to compared. So we normalize the score by the following:

$$\alpha_{i,r,j} = \frac{\exp(d(i, r, j))}{\sum_{i,r',j'\in l_i} \exp(d(i, r', j'))}$$

In this chapter, the attention mechanism a is a single feed-forward neural network which the weight matrix is $\mathbf{a} \in \mathbb{R}^{2F}$. In this chapter, the LeakyReLU (with Negative slope coefficient 0.2) was used as the non-liner activation function. Based on that, the attention score formula can be expressed as:

$$\alpha_{i,r,j} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\intercal[\mathbf{W}_r\mathbf{e}_i \| \mathbf{W}_r\mathbf{e}_j]))}{\sum_{i,r',j'\in l_i} \exp(\text{LeakyReLU}(\mathbf{a}^\intercal[\mathbf{W}_{r'}\mathbf{e}_i \| \mathbf{W}_{r'}\mathbf{e}_{j'}]))}$$

Different from GAT [61], we additionally include the relationship in the representation to better reflect the user's preference.

In our model, the path searching is the last and the most important step. We can get a weight graph for user u, g_u^t at this timestamp which contains all the items he may interested in based on the attention score. Furthermore, it's common that an item may have several entities, to make them on the same scale, we average the score. In addition, we also connect the item which not appear in user's interest but have higher-order relation with the connected item with weight. The weight is determined by the existing attention score's minimal value to minimize the interference from this operation. It's hard for critic network to evaluate the policy directly, so we will evaluate the action which is the recommended item instead. Suppose the item recommended by the actor-network is the i_p, but the target item is i_t. The critic network uses the Dijkstra algorithm to calculate the shortest distance between i_p and i_t to evaluate this action. The reward is designed for training process as:

$$r_{pt} = \frac{1}{\text{Distance}(i_p, i_t)} + 50 \cdot \text{in}(i_p, g_u^t)$$

where we empirically set the constant to 50; the function, $\text{in}(\cdot, g_u^t)$, checks whether an item exists in the graph—the algorithm will automatically terminate if the item i_p is not in the graph, except special cases.

Specifically, we calculate the similarity between the item recommended with the item user selected, and provide positive feedback when the similarity surpass 0.5. If the item already exists in the graph, we apply logistic matrix factorization and rating to simulate the feedback (to be discussed in the next section). In addition, we also consider the feedback x_i^t's impact on the system. We add the penalty parameter, which will directly affect the weight of the graph. If the feedback is positive, the weight will not be changed. If the feedback is negative, we will penalize the cor-

responding recommendation route by reducing the weight. Precisely, we maintain a discount factor $\lambda \in [0, 1]$ to adjust the rank of all the penitential recommendation results. In this chapter, the λ is set to 0.1 if it was penalized otherwise λ is set to 1.

4.1.3 Attention Mechanism

The attention mechanism which used in the GAT is pretty rough and may suffer the low efficiency problem. Consider about that case, and benefit from recent year's advance of the transformer, we can adapt the LSHAttention to relief this problem. The multi-head attention is memory inefficient due to the size of Q, K and V. Assume that the Q, K, V have the shape $[|batch|, length, d_{model}]$ where $|\cdot|$ represents the size of the variable. The term QK^T will produce a tensor in shape $[length, length, d_{model}]$. Given the standard image size, the length $\times$ length will take most of the memory. Kitaev et al. [107] proposed a Locality Sensitive Hashing(LSH) based Attention to address this issue. Firstly, we rewire the basic attention formula into each query position i in the partition form:

$$a_i = \sum_{j \in P_i} \frac{\exp(q_i \cdot k_j - z(i, P_i))v_j}{\sqrt{d_k}} \text{ where } P_i = \{j : i \geq j\}$$

where the function z is the partition function, P_i is the set which query position i attends to. During model training, we normally conduct the batching and assume that there is a larger set $P_i^L = \{0, 1, \cdots, l\} \supseteq P_i$ without considering elements not in P_i:

$$a_i = \sum_{j \in P_i^L} \frac{\exp(q_i \cdot k_j - N(j, P_i) - z(i, P_i))v_j}{\sqrt{d_k}} \tag{4.1}$$

$$\text{where } N(j, P_i) = \begin{cases} 0 & j \in P_i \\ \infty & j \notin P_i \end{cases} \tag{4.2}$$

Then, with a hash function $h(\cdot)$: $h(q_i) = h(k_j)$, we can rewire the P_i as:

$$P_i = \{j : h(q_i) = h(k_j)\}$$

In order to guarantee that the number of keys can uniqually match with the number of quires, we need to ensure that $h(q_i) = h(k_i)$ where $k_i = \frac{q_i}{\|q_i\|}$. During the hashing process, some similar items may fall in different buckets because of the hashing. The multi-round hashing provides an effective way to overcome this issue. Suppose there is n_r round, and each round has different hash functions $\{h_1, \cdots, h_{n_r}\}$, so we have:

$$P_i = \bigcup_{g=1}^{n_r} P_i^g \text{ where } P_i^g = \{j : h^g(q_i) = h^g(q_j)\} \tag{4.3}$$

Considering the batching case, we need to get the P_i^L for each round g:

$$\widehat{P_i^L} = \left\{j : \left\lfloor \frac{i}{m} \right\rfloor - 1 \leq \left\lfloor \frac{j}{m} \right\rfloor \leq \left\lfloor \frac{i}{m} \right\rfloor \right\} \tag{4.4}$$

where $m = \frac{2l}{n_r}$. The last step is to calculate the LSH attention score in parallel. With the formula (4.1) and (4.3), we can derive:

$$a_i = \sum_{g=1}^{n_r} \frac{\exp(z(i, P_i^g) - z(i, P_i))a_i^g}{\sqrt{d_k}}$$

$$\text{where } a_i^g = \sum_{j \in \widehat{P_i^L}} \frac{\exp(q_i \cdot k_j - m_{i,j}^g - z(i, P_i^g))v_j}{\sqrt{d_k}}$$

$$\text{with } m_{i,j}^g = \begin{cases} \infty & j \notin P_i^g \\ 10^5 & i = j \\ \log|\{g' : j \in P_i^{g'}\}| & otherwise \end{cases}$$

where $|\cdot|$ represents the number of elements.

4.1.4 Training and Optimization

Training the off-policy actor-critic network [108] requires train two parts of the neural network simultaneously. We apply the Deep Deterministic Policy Gradient (DDPG) to train our model, where we train the critic by minimising a loss function:

$$l(\theta_\psi) = \frac{1}{N}\sum_{j=1}^{N}((r+\gamma\xi) - \psi_{\theta_\psi}(S_t, a_t))^2$$

$$\text{where } \xi = \psi_{\theta'_\psi}(S_{t+1}, \phi_{\theta'_\phi}(S_{t+1}))$$

where θ_ψ and θ_ϕ are the parameters for the critic network and the actor network, respectively; N is the size of mini-batch; γ is the discount factor; $\psi_{\theta'_\psi}$ and $\phi_{\theta'_\phi}$ are the target critic and target actor network, respectively. Training an actor-critic network can be time-consuming; therefore, we use the actor-critic network with experience replay structure (ACER) [109] to speed up the off-policy training by policy gradient of the importance weighted:

$$\widehat{g} = (\prod_{t=0}^{k}\rho_t)\sum_{t=0}^{k}(\sum_{i=0}^{k-t}\gamma r_{t+i})\nabla_\theta \log(\pi_\theta(a_t|S_t))$$

where ρ_t is the importance factor. As the $\prod_{t=0}^{k}\rho_t$ may generate extremely large result as time goes by will make the model not stable. Based on that we treat each single step i policy update as:

$$g_i = \mathbb{E}_\beta[\rho_t \nabla_\theta \log \pi_\theta(a_t|S_t) Q_\pi(S_t, a_t)]$$

The ρ_t can be represented as:

$$\rho_t = \frac{\pi(a_t|S_t)}{\mu(S_t|a_t)}$$

The ρ_t is not stable as the value of $\pi(a_t|S_t)$ increasing, in order to control this value, the importance weight truncation with bias correction [110] is used.

$$
\begin{aligned}
g =& \mathbb{E}_{S_t a_t}[\rho_t \nabla_\theta \log \pi_\theta(a_t|S_t) Q_\pi(S_t, a_t)] \\
=& \mathbb{E}_{S_t}\left[E_{a_t}[\overline{\rho}_t \nabla_\theta \log \pi_\theta(a_t|S_t) Q_\pi(S_t, a_t)] + \mathbb{E}_{a\sim\pi}\left(\left[\frac{\rho_t(a) - c}{\rho_t(a)} \right]_+ \nabla_\theta \log(a_t|S_t) Q_\pi(S_t, a_t) \right) \right]
\end{aligned}
$$

where the $\overline{\rho}_t = \min\{c, \rho_t\}$ and the function $[\cdot]_+$ is the ReLU function. The above formula only consider about the correlation term, but the $Q_\pi(S_t, a_t)$ is unknown, so we need to estimate this term by using Retrace [111, 112].

$$
Q^r(S_t, a_t) = r_t + \gamma \overline{\rho}_{t+1}[Q^r(S_{t+1}, a_{t+1}) - Q(S_{t+1}, a_{t+1})] + \gamma V(S_{t+1})
$$

We use Q_{θ_v} to represent the approximation of the $Q_\pi(S_t, a_t)$ By combining all the formula above we can get the final form of the g:

$$
\begin{aligned}
\widehat{g} =& \mathbb{E}_{S_t}\Bigg[E_{a_t}[\overline{\rho}_t \nabla_\theta \log \pi_\theta(a_t|S_t) Q^r(S_t, a_t)] + \\
& \mathbb{E}_{a\sim\pi}\left(\left[\frac{\rho_t(a) - c}{\rho_t(a)} \right]_+ \nabla_\theta \log(a_t|S_t) Q_{\theta_v}(S_t, a_t) \right) \Bigg]
\end{aligned}
$$

So the importance weight truncation with bias correction applied in the ACER and the policy gradient comes to:

$$
\begin{aligned}
\widehat{g}_t^{acer} = & \overline{\rho}_t \nabla_\theta \log(\pi_\theta(a_t|S_t))[r(S_t, a_t) - V_{\theta_v}(S_t)] + \mathbb{E}_{a\sim\pi} \\
& \left[\frac{\rho_t(a) - c}{\rho_t(a)} \right]_+ \log(\pi_\theta(a_t|S_t))[r(S_t, a) - V_{\theta_v}(S_t)]
\end{aligned}
$$

where the classical baseline $V_{\theta_v}(S_t)$ is used to reduce variance, and c is a constant determine by $p_t(a)$. As aforementioned, we assume that the user's interest will not change sharply in a short time period [113]. In addition, the experience replay focuses on the actor-network, so we focus on the policy gradient here. Based on the condition mentioned previously, we need to make sure when the model training does not have a suitable step size. A bad step size will make the model dis-converge.

This problem is called trust region policy optimization (TRPO) [114], which aims to limit the agent to update the policy inside the trust-region so that it will not go out-of-board. Mathematically, we want to control the KL-divergence not change sharply. For our model, the trust region policy optimization can be written as:

$$\begin{aligned}&\underset{z}{\text{minimize}}\ \frac{1}{2}\|\widehat{g}_t^{acer} - z\|_2^2\\&\text{subject to}\nabla_{\theta_\phi(S_t)} D_{KL}[f(\cdot|\theta_{\phi_a}(S_t))\|f(\cdot|\theta_\phi(S_t))]^\intercal, z \leq \delta\end{aligned}$$

we can work out that:

$$\begin{aligned}z^* &= \widehat{g}_t^{acer} - \max\left\{0, \frac{k^\intercal \widehat{g}_t^{acer} - \delta}{\|k\|_2^2}\right\}\\&\text{where } k = \nabla_{\theta_\phi(S_t)} D_{KL}[f(\cdot|\theta_{\phi_a}(S_t))\|f(\cdot|\theta_\phi(S_t))]\end{aligned}$$

In summary, based on those result we can use the parameter update rule for actor ϕ as:

$$\theta \leftarrow \theta + \frac{\partial\theta(S)}{\partial\theta} z^*$$

Based on the derivation above, it's easy to find that the TRPO can efficiently limit the step size of the agent when updating the policy. Which means that every step can not over than $\frac{\partial\theta(S)}{\partial\theta} z^*$.

4.2 Experiments

4.2.1 Experimental Setup

We conducted experiments on four public real-world datasets. We constructed a knowledge graph for each dataset and then split it into user-specific collaborative

knowledge graphs. After that, we map items into entities via title matching if there is a mapping relation. We evaluated our model in a simulated online environment built upon offline public datasets. This way, we avoided collecting private user information and expensive online training. Specifically, the simulator generated feedback based on logistic matrix factorization and the user's rating. For example, during the training, the system recommends item i to user u at time t, and the critic finds that the item i inside the knowledge graph g_u^t if u already provide the rating for i, we just simply map the rating into feedback as positive if ration larger than or equal to 4. If not, we just let LMF decide the feedback. We randomly split each dataset into a training set (70%), a validation set (10%), and a testing set (20%) to conduct 10-fold cross-validation. The discount factor γ was initialized to 0.99.

4.2.2 Overall Comparison

We compared our model with several competitive baselines. Results (Table 4.1) show our model consistently outperforms the baselines on all the six public datasets. MAB-based methods generally perform poorly due to their inablility to find the upper confidence boundary on the extremely sparse datasets; but they are more stable on the most sparsity dataset, Book-Crossing, resulting in a much lower error-range than other interactive recommendation methods (TPGR and ours). Consider about the trade-off between the training time and the performance improvement, we report the first order in the table 4.1.

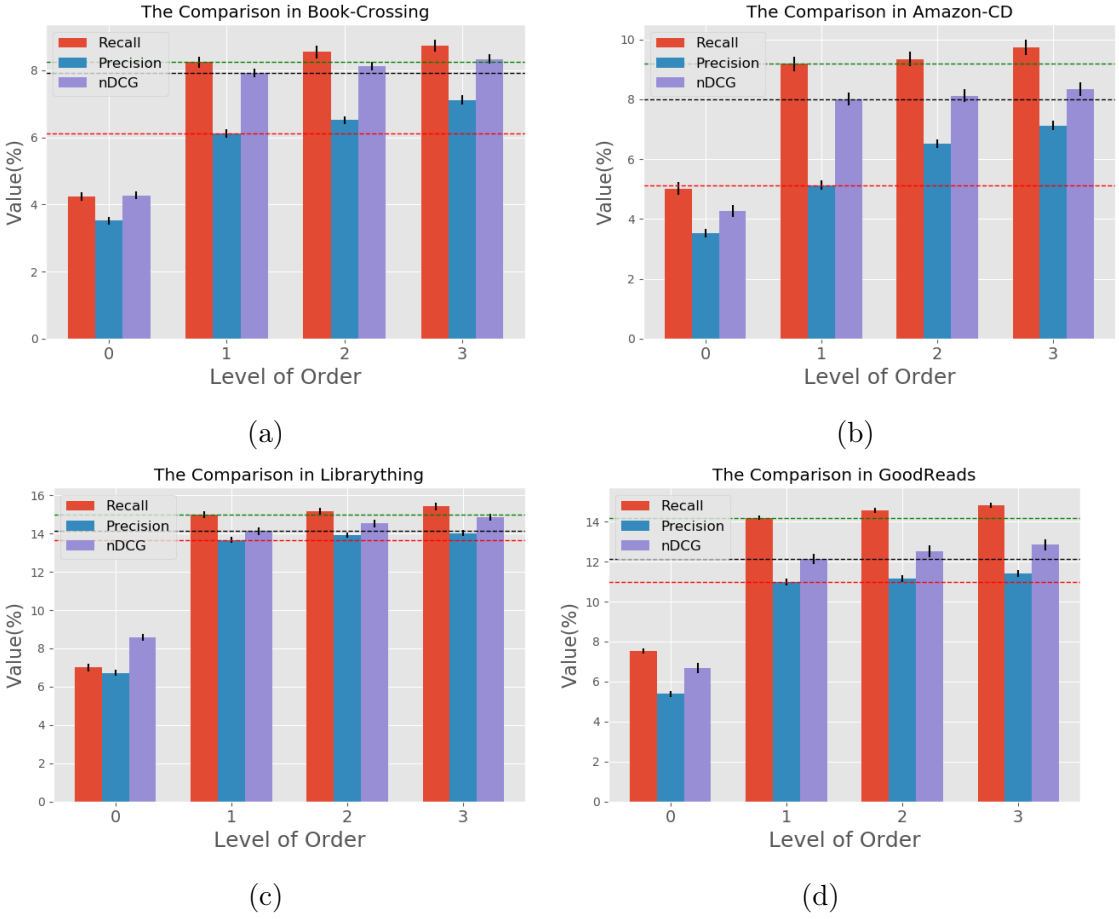


Figure 4.3: The figures (a)-(d) show the effect of various levels of attention. The level of order indicate the number of Attention layer we employed. The level 0 indicates the deployment without attention network. The dot lines are used for indicate the baseline results where the green line used for the recall, red line for precision and black line for nDCG.

4.2.3 Ablation Study

Our ablation studies confirm our assumption that an increased level of order improves, but only slightly improves, the performance, given that most users are likely interested in the item which have strong relation with previous purchases (or other actions). The graph attention network affects the result significantly because the whole model is carefully design for the generated attentive graph. If all the attention

Table 4.1: The overall results of our model comparison with several state-of-arts models in different datasets. The result was reported by using the percentage and based on top-10 recommendation as mentioned before. The highlighted result in bold is the best result.

Dataset	Amazon CD			Librarything		
Measure (%)	Recall	Precision	nDCG	Recall	Precision	nDCG
HLinearUCB [9]	3.112 ± 0.331	2.647 ± 0.171	4.005 ± 0.341	8.102 ± 0.396	7.431 ± 0.204	8.157 ± 0.241
FactorUCB [40]	3.531 ± 0.232	4.512 ± 0.242	6.012 ± 0.251	8.541 ± 0.241	8.162 ± 0.355	8.653 ± 0.351
ICTRUCB [11]	4.124 ± 0.293	3.110 ± 0.395	5.982 ± 0.602	9.201 ± 0.241	7.980 ± 0.151	8.012 ± 0.466
kNN Bandit [115]	3.944 ± 0.231	2.901 ± 0.192	4.004 ± 0.124	8.787 ± 0.121	8.002 ± 0.144	8.989 ± 0.211
DRN [14]	8.006 ± 0.232	4.234 ± 0.241	6.112 ± 0.241	10.841 ± 0.112	9.412 ± 0.242	9.527 ± 0.455
TPGR [104]	7.294 ± 0.312	2.872 ± 0.531	6.128 ± 0.541	14.713 ± 0.644	12.410 ± 0.612	13.225 ± 0.722
KGAT [116]	8.234 ± 0.244	**5.339 ± 0.297**	6.442 ± 0.342	12.965 ± 0.146	9.438 ± 0.440	10.401 ± 0.312
PGPR[18]	6.619 ± 0.123	1.892 ± 0.143	5.970 ± 0.131	11.531 ± 0.241	10.333 ± 0.341	12.641 ± 0.442
Ours	**9.178 ± 0.241**	5.132 ± 0.155	**8.002 ± 0.213**	**14.981 ± 0.184**	**13.667 ± 0.151**	**14.128 ± 0.199**

Dataset	Book-Crossing			GoodReads		
Measure (%)	Recall	Precision	nDCG	Recall	Precision	nDCG
HLinearUCB	2.421 ± 0.131	1.724 ± 0.141	2.865 ± 0.322	7.917 ± 0.303	5.151 ± 0.214	6.561 ± 0.351
FactorUCB	3.123 ± 0.141	2.976 ± 0.223	3.536 ± 0.241	5.643 ± 0.441	4.129 ± 0.221	6.122 ± 0.395
ICTRUCB	3.441 ± 0.121	3.421 ± 0.333	4.001 ± 0.321	8.415 ± 0.132	6.432 ± 0.221	7.124 ± 0.241
kNN Bandit	2.333 ± 0.122	1.980 ± 0.111	2.501 ± 0.301	8.321 ± 0.124	7.008 ± 0.104	8.541 ± 0.222
DRN	7.124 ± 0.122	4.123 ± 0.112	7.433 ± 0.142	10.620 ± 0.123	8.432 ± 0.241	9.461 ± 0.442
TPGR	7.246 ± 0.321	4.523 ± 0.442	7.270 ± 0.412	13.219 ± 0.323	10.322 ± 0.442	9.825 ± 0.642
KGAT	7.335 ± 0.256	5.956 ± 0.069	7.653 ± 0.163	12.659 ± 0.315	10.239 ± 0.221	10.569 ± 0.158
PGPR	6.998 ± 0.112	3.932 ± 0.121	7.333 ± 0.133	11.421 ± 0.223	10.042 ± 0.212	9.234 ± 0.242
Ours	**8.241 ± 0.173**	**6.125 ± 0.122**	**7.923 ± 0.133**	**14.199 ± 0.123**	**10.992 ± 0.177**	**12.141 ± 0.268**

score lost, the search will get stuck and make random selections in the constructed unweighted user-specific graph, which will affect the actor network's policy.

4.3 Conclusion

In this chapter, we have proposed a Reinforcement Learning based Knowledge Graph Attention Network (RL-KGAN) for interactive recommendation. RL-KGAN uses the actor-critic learning framework to harness the interactions between users and the recommendation system. We employ the experience replay and optimize the model through trust range policy optimization to speed up convergence. Our extensive experiments over an online simulator with six public real-world datasets demonstrate its superior performance.

Chapter 5

Conclusion

This dissertation studies the interactive recommendation. We firstly overview, compare and discuss the existing state-of-the-art methods for the reinforcement learning-based recommendation systems. We briefly discuss the advantages and disadvantages of the existing methods. Secondly, we identified the major existing challenges in interactive recommendation system, i.e., the dynamic user interest and the efficiency problem of reinforcement learning. To address these two challenges, we proposed several methods. We first introduced a new distributed representation (expert2vec) for expert which is used on solving the CQA problem and the expert recommendation problem. Expert2vec is the distributed representation which contains the information about user and topic and its corresponding rank. It is able to capture the dynamics of user interest. Then, we proposed a side information-augmented method for the interactive recommendation. It uses the critic-actor learning framework to harness the interactions between users and the recommendation systems and employs a local knowledge network to improve the stability and quality of the critic network for better decision-making. Extensive experiments over

an online simulator with six public real-world datasets demonstrate its superior performance over state-of-the-art models. To verify the effectiveness of each component, we also conduct the ablation study for the local knowledge network and attention mechanism and present the performance both in normal cases and extreme cases. Finally, we investigate the problem which exists on the KGRL, we apply the TRPO to the optimization process which can limit the update steps and boost the convergence. In addition, we extend the normal knowledge graph into the Collaborative Knowledge Graph which can enrich the side information.

In the future, the interactive recommendation system can be further improved following the recent advances in the Brain-Computer Interface (BCI) [117]. The BCI provides a new way to understand the user's intention by using the EEG signal [118]. Current research can recover from the brain wave to user's vision [119, 120, 121, 6]. In the future, it may be possible to use the BCI as the intention detection to represent the user's interaction. Brain's activity can be more accurate than the user's behaviour [48, 122]. Besides the EEG, the fMRI can be another approach to understand user's behaviour [123, 124, 125]. A recent study in understanding human mind [126] provides the direction about how to adopt those techniques into the recommendation system. In summary, for further studies, it is possible to fuse the EEG signal and the fMRI image [127, 128] to understand user's intention more accurately. Based on that, the interactive recommendation system can be further enhanced.

Bibliography

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, p. 5, 2019.

[2] S. Zhang, Y. Tay, L. Yao, B. Wu, and A. Sun, "Deeprec: an open-source toolkit for deep learning based recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 6581–6583, AAAI Press, 2019.

[3] F. Yuan, L. Yao, and B. Benatallah, "Adversarial collaborative neural network for robust recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1065–1068, 2019.

[4] L. Yao, Q. Z. Sheng, A. H. Ngu, H. Ashman, and X. Li, "Exploring recommendations in internet of things," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 855–858, 2014.

[5] L. Yao, Q. Z. Sheng, X. Wang, W. E. Zhang, and Y. Qin, "Collaborative location recommendation by integrating multi-dimensional contextual information," *ACM Transactions on Internet Technology (TOIT)*, vol. 18, no. 3, pp. 1–24, 2018.

[6] L. Yao, Q. Z. Sheng, A. H. Ngu, and X. Li, "Things of interest recommendation by leveraging heterogeneous relations in the internet of things," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 2, pp. 1–25, 2016.

[7] L. Yao, X. Wang, Q. Z. Sheng, S. Dustdar, and S. Zhang, "Recommendations on the internet of things: Requirements, challenges, and directions," *IEEE Internet Computing*, vol. 23, no. 3, pp. 46–54, 2019.

[8] X. Zhao, W. Zhang, and J. Wang, "Interactive collaborative filtering," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, 2013.

[9] H. Wang, Q. Wu, and H. Wang, "Learning hidden features for contextual bandits," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 1633–1642, ACM, 2016.

[10] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[11] Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz, and G. Grabarnik, "Online interactive collaborative filtering using multi-armed bandit with dependent arms," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[12] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[13] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "a," in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 95–103, ACM, 2018.

[14] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, pp. 167–176, IW3C2, 2018.

[15] Y. Liu, Y. Zhang, Q. Wu, C. Miao, L. Cui, B. Zhao, Y. Zhao, and L. Guan, "Diversity-promoting deep reinforcement learning for interactive recommendation," *arXiv preprint arXiv:1903.07826*, 2019.

[16] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1040–1048, ACM, 2018.

[17] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, "Stabilizing reinforcement learning in dynamic environment with application to online recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1187–1196, ACM, 2018.

[18] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 285–294, ACM, 2019.

[19] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," in *Advances in Neural Information Processing Systems*, pp. 2731–2741, 2019.

[20] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54–77, 2017.

[21] Q. Zhao and S. S. Bhowmick, "Sequential pattern mining: A survey," *ITechnical Report CAIS Nayang Technological University Singapore*, vol. 1, no. 26, p. 135, 2003.

[22] F. Garcin, C. Dimitrakakis, and B. Faltings, "Personalized news recommendation with context trees," in *Proceedings of the 7th ACM conference on Recommender systems*, pp. 105–112, ACM, 2013.

[23] X. Wu, Q. Liu, E. Chen, L. He, J. Lv, C. Cao, and G. Hu, "Personalized next-song recommendation in online karaokes," in *Proceedings of the 7th ACM conference on Recommender systems*, pp. 137–140, ACM, 2013.

[24] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[25] B. Twardowski, "Modelling contextual information in session-aware recommender systems with neural networks," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 273–276, ACM, 2016.

[26] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 17–22, ACM, 2016.

[27] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 130–137, ACM, 2017.

[28] Y. J. Ko, L. Maystre, and M. Grossglauser, "Collaborative recurrent neural networks for dynamic recommender systems," in *Journal of Machine Learning Research: Workshop and Conference Proceedings*, vol. 63, 2016.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[30] C.-Y. Wu, A. Ahmed, A. Beutel, and A. J. Smola, "Joint training of ratings and reviews with recurrent recommender networks," 2016.

[31] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," 2018.

[32] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[33] N. Ye *et al.*, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166, p. 169, West Point, NY, 2000.

[34] H. Zhang, W. Ni, X. Li, and Y. Yang, "Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-markov approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[35] J. Xu, T. Xing, and M. Van Der Schaar, "Personalized course sequence recommendations," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5340–5352, 2016.

[36] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 108–116, 2018.

[37] J. L. Vincent, A. Z. Snyder, M. D. Fox, B. J. Shannon, J. R. Andrews, M. E. Raichle, and R. L. Buckner, "Coherent spontaneous activity identifies a hippocampal-parietal memory network," *Journal of neurophysiology*, vol. 96, no. 6, pp. 3517–3531, 2006.

[38] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95, IEEE, 2017.

[39] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 661–670, ACM, 2010.

[40] H. Wang, Q. Wu, and H. Wang, "Factorization bandits for interactive recommendation," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[41] Y. Shen, Y. Deng, A. Ray, and H. Jin, "Interactive recommendation via deep neural memory augmented contextual bandits," in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 122–130, ACM, 2018.

[42] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan, "Active learning in recommender systems," in *Recommender systems handbook*, pp. 809–846, Springer, 2015.

[43] W. Wang, H. Yin, Z. Huang, X. Sun, and N. Q. V. Hung, "Restricted boltzmann machine based active learning for sparse recommendation," in *International Conference on Database Systems for Advanced Applications*, pp. 100–115, Springer, 2018.

[44] L. Zhao, S. J. Pan, E. W. Xiang, E. Zhong, Z. Lu, and Q. Yang, "Active transfer learning for cross-system recommendation," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[45] R. Li, S. E. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards deep conversational recommendations," in *Advances in neural information processing systems*, pp. 9725–9735, 2018.

[46] M. Kraus, F. Fischbach, P. Jansen, and W. Minker, "A comparison of explicit and implicit proactive dialogue strategies for conversational recommendation," in *Proceedings of The 12th Language Resources and Evaluation Conference*, pp. 429–435, 2020.

[47] X. Ren, H. Yin, T. Chen, H. Wang, N. Q. V. Hung, Z. Huang, and X. Zhang, "Crsal: Conversational recommender systems with adversarial learning," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 4, pp. 1–40, 2020.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[49] A. Grotov and M. de Rijke, "Online learning to rank for information retrieval: Sigir 2016 tutorial," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, 2016.

[50] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[51] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 635–644, ACM, 2017.

[52] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2019.

[53] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 353–362, ACM, 2016.

[54] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 505–514, ACM, 2018.

[55] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, pp. 1835–1844, 2018.

[56] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, pp. 2787–2795, 2013.

[57] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[58] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[59] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 687–696, 2015.

[60] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[61] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[62] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

[63] T. Jie and P. Abbeel, "On a connection between importance sampling and the likelihood ratio policy gradient," in *Advances in Neural Information Processing Systems*, pp. 1000–1008, 2010.

[64] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, pp. 1–9, 2013.

[65] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, and J. Ye, "Environment reconstruction with hidden confounders for reinforcement learning based recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 566–576, 2019.

[66] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, and P. S. Thomas, "Learning action representations for reinforcement learning," in *International Conference on Machine Learning*, 2019.

[67] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *International Conference on Machine Learning*, pp. 1052–1061, 2019.

[68] X. Zhao, L. Xia, Y. Zhao, D. Yin, and J. Tang, "Model-based reinforcement learning for whole-chain recommendations," *arXiv preprint arXiv:1902.03987*, 2019.

[69] A. Piscopo, C. Phethean, and E. Simperl, "What makes a good collaborative knowledge graph: group composition and quality in wikidata," in *International Conference on Social Informatics*, pp. 305–322, Springer, 2017.

[70] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *International Semantic Web Conference*, pp. 542–557, Springer, 2013.

[71] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[72] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[73] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, pp. 593–607, Springer, 2018.

[74] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018.

[75] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

[76] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[77] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Advances in neural information processing systems*, pp. 6410–6421, 2018.

[78] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[79] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler, "Fast interactive object annotation with curve-gcn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5257–5266, 2019.

[80] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5177–5186, 2019.

[81] R. D. Soberanis-Mukul, S. Albarqouni, and N. Navab, "An uncertainty-driven gcn refinement strategy for organ segmentation," *arXiv preprint arXiv:1906.02191*, 2019.

[82] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[83] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[84] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[85] S. Zhang, L. Yao, X. Xu, S. Wang, and L. Zhu, "Hybrid collaborative recommendation via semi-autoencoder," in *International Conference on Neural Information Processing*, pp. 185–193, Springer, 2017.

[86] S. Zhang, L. Yao, and X. Xu, "Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 957–960, 2017.

[87] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[88] Y. Tian, D. Lo, and J. Lawall, "Sewordsim: Software-specific word similarity database," in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 568–571, ACM, 2014.

[89] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[90] X. Wang, C. Huang, L. Yao, B. Benatallah, and M. Dong, "A survey on expert recommendation in community question answering," *Journal of Computer Science and Technology*, vol. 33, no. 4, pp. 625–653, 2018.

[91] B. Yang and S. Manandhar, "Tag-based expert recommendation in community question answering," in *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pp. 960–963, IEEE, 2014.

[92] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*, pp. 880–887, 2008.

[93] L. Du, W. Buntine, and H. Jin, "A segmented topic model based on the two-parameter poisson-dirichlet process," *Machine learning*, vol. 81, no. 1, pp. 5–19, 2010.

[94] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial personalized ranking for recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 355–364, ACM, 2018.

[95] 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 *The World Wide Web Conference*, pp. 151–161, ACM, 2019.

[96] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *The World Wide Web Conference*, pp. 3307–3313, ACM, 2019.

[97] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[98] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018.

[99] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[100] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[101] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Advances in Neural Information Processing Systems*, 2018.

[102] W. Zhang, U. Paquet, and K. Hofmann, "Collective noise contrastive estimation for policy transfer learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[103] C. C. Johnson, "Logistic matrix factorization for implicit feedback data," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[104] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3312–3320, 2019.

[105] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[106] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2110–2119, 2018.

[107] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[108] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 179–186, 2012.

[109] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *ICLR*, 2016.

[110] R. Munos, "Q (λ) with off-policy corrections," in *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings*, vol. 9925, p. 305, Springer, 2016.

[111] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare, "Safe and efficient off-policy reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

[112] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 1995–2003, 2016.

[113] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31, JMLR. org, 2017.

[114] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.

[115] J. Sanz-Cruzado, P. Castells, and E. López, "A simple multi-armed nearest-neighbor bandit for interactive recommendation," in *13th ACM Conference on Recommender Systems*, 2019.

[116] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[117] X. Zhang, L. Yao, X. Wang, J. Monaghan, and D. Mcalpine, "A survey on deep learning based brain computer interface: Recent advances and new frontiers," *arXiv preprint arXiv:1905.04149*, 2019.

[118] W. Chen, S. Wang, X. Zhang, L. Yao, L. Yue, B. Qian, and X. Li, "Eeg-based motion intention recognition via multi-task rnns," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 279–287, SIAM, 2018.

[119] D. Zhang, L. Yao, X. Zhang, S. Wang, W. Chen, R. Boots, and B. Benatallah, "Cascade and parallel convolutional recurrent neural networks on eeg-based intention recognition for brain computer interface," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[120] X. Zhang, L. Yao, Q. Z. Sheng, S. S. Kanhere, T. Gu, and D. Zhang, "Converting your thoughts to texts: Enabling brain typing via deep feature learning of eeg signals," in *2018 IEEE international conference on pervasive computing and communications (PerCom)*, pp. 1–10, IEEE, 2018.

[121] X. Zhang, L. Yao, X. Wang, W. Zhang, S. Zhang, and Y. Liu, "Know your mind: Adaptive brain signal classification with reinforced attentive convolutional neural networks," *arXiv preprint arXiv:1802.03996*, 2018.

[122] X. Zhang, L. Yao, X. Wang, W. Zhang, S. Zhang, and Y. Liu, "Know your mind: Adaptive cognitive activity recognition with reinforced cnn," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 896–905, IEEE, 2019.

[123] O. J. Arthurs and S. Boniface, "How well do we understand the neural origins of the fmri bold signal?," *TRENDS in Neurosciences*, vol. 25, no. 1, pp. 27–31, 2002.

[124] E. M. Husband, L. A. Kelly, and D. C. Zhu, "Using complement coercion to understand the neural basis of semantic composition: evidence from an fmri study," *Journal of cognitive neuroscience*, vol. 23, no. 11, pp. 3254–3266, 2011.

[125] R. A. Poldrack, "The future of fmri in cognitive neuroscience," *Neuroimage*, vol. 62, no. 2, pp. 1216–1220, 2012.

[126] T. Ohnishi, Y. Moriguchi, H. Matsuda, T. Mori, M. Hirakata, E. Imabayashi, K. Hirao, K. Nemoto, M. Kaga, M. Inagaki, *et al.*, "The neural network for the mirror system and mentalizing in normally developed children: an fmri study," *Neuroreport*, vol. 15, no. 9, pp. 1483–1487, 2004.

[127] S. Debener, M. Ullsperger, M. Siegel, and A. K. Engel, "Single-trial eeg–fmri reveals the dynamics of cognitive function," *Trends in cognitive sciences*, vol. 10, no. 12, pp. 558–563, 2006.

[128] H. Laufs, A. Kleinschmidt, A. Beyerle, E. Eger, A. Salek-Haddadi, C. Preibisch, and K. Krakow, "Eeg-correlated fmri of human alpha activity," *Neuroimage*, vol. 19, no. 4, pp. 1463–1476, 2003.