DRprofiling: deep reinforcement user profiling for recommendations in heterogenous information networks

Article

Accepted Version

Liang, H. (2020) DRprofiling: deep reinforcement user profiling for recommendations in heterogenous information networks. Transactions on Knowledge and Data Engineering. ISSN 1558-2191 doi: https://doi.org/10.1109/TKDE.2020.2998695 Available at http://centaur.reading.ac.uk/90823/

DRprofiling: Deep Reinforcement User Profiling for Recommendations in Heterogenous Information Networks

Huizhi Liang

Abstract—Recommender systems are popular for personalization in online communities. Users, items, and other affiliated information such as tags, item genres, and user friends of an online community form a heterogenous information network. User profiling is the foundation of personalized recommender systems. It provides the basis to discover knowledge about an individual user's interests to items. Typically, users are profiled with their direct explicit or implicit ratings, which ignored the inter-connections among users, items, and other entity nodes of the information network. This paper proposes a deep reinforcement user profiling approach for recommender systems. The user profiling process is framed as a sequential decision making problem which can be solved with a Reinforcement Learning (RL) agent. The RL agent interacts with the external heterogenous information network environment and learns a decision making policy network to decide whether there is an interest or preference path between a user and an unobserved item. To effectively train the RL agent, this paper proposes a multi-iteration training process to combine both expert and data-specific knowledge to profile users, generate meta-paths, and make recommendations. The effectiveness of the proposed approaches is demonstrated in experiments conducted on three datasets.

Index Terms—Reinforcement Learning, Recommender Systems, User Profiling

1 Introduction

Recommender systems are popular for personalisation. They can help to reduce information overload for users in online communities, i.e., making suggestions regarding which information is most relevant to an individual user [1]. User profiling is the foundation of personalised recommender systems. It provides the basis to discover knowledge about users' interests, preferences, and information needs, from user behavioural information such as ratings, purchasing, social tagging, and clicks [2]. A typical online community not only has explicit or implicit rating data, but also has other affiliated information such as item genres, categories, tags, and friends. Together with users and items, they form heterogeneous information networks or graphs [3]. Because of the profoundness of human beings and the complexity of heterogenous information network, how to profile users effectively and make quality recommendations remain important open research questions.

The most commonly used user profiling approach in recommender systems is to represent users directly based on their explicit or implicit ratings or called direct user-item interactions. For example, a user can be profiled with a set of observed items based on the binary user-item interactions (e.g., purchase transactions). Based on user profiles, collaborative filtering approaches (CF) [1] such as *neighborhood* based and matrix factorisation approach [4] are used to make rating predictions or Top-N item recommendations [1]. However, the recommendation quality is largely limited due to the commonly existing data sparsity problem of direct user-item interactions [5]. Moreover, these approaches ignored those useful affiliated information and fail to consider the interconnections among all the nodes in information networks. To alleviate this problem, graph-based algorithms and link prediction algorithms [5] have been proposed to explore transitive user-item associations.

The recommendation problem can be viewed as a link prediction problem in heterogenous information networks, which is to predict wether there is a link between a user node and an unobserved item node or to infer the connectivity probability between them in information networks [6]. To achieve quality recommendations, it is critical to find the most informative or predictive paths between user nodes and item nodes. For a target user, if we can find the predictive paths that lead to this user's observed items, then it is more likely that these paths will help to find those unobserved items that he or she will be interested. For easy explaination, we define the process of finding the predictive paths between a user node and item nodes in heterogenous information networks as *User Profiling* process in this paper.

Deep reinforcement learning techniques [7] have emerged as a promising framework for various applications, such as game playing [7], decision making [8]. Value based and policy based approaches are two main approaches to solve RL problems [8]. RL framework has been successfully applied to many game settings, such as Atari and Go [7]. Applying RL framework to recommender systems arouses increasing interests in both academic and industry recently. Comparing with traditional CF and deep neural network based approaches, reinforcement learning can optimise a sequence of recommendation decisions for a long term goal such as profit, loyalty, or user long term engagement [9, 10, 11]. Most existing work proposed approaches to train a RL recommender agent to interact with a logged online target user. For these approaches, the environment is usually a logged online target user [12, 11]. This setting of environment has a few challenges in recommendation systems, because of the complexity of human behaviour and the difficulty of getting large samples from online users to train a good policy [12]. One important fact that existing approaches ignored is that the target user is connected to a heterogeneous information network that formed by other users, items, and other information such as item content, genres, taxonomy categories, tags, review, friends, occupations, and social networks. In this paper, the heterogenous information network is used as important information source to model the user decision making patterns.

This paper sets the heterogeneous information network as the environment, then frames the user profiling process of inferring whether a user is interested in an item in heterogenous information networks as a Markov Decision Process (MDP). A Reinforcement Learning (RL) agent is proposed to solve the Markov Decision Process. The RL agent will interact with the external heterogenous information network environment and receive rewards. It will learn a decision making model to decide which actions to take to find potential interesting item nodes. The training process will be guided by both expert knowledge and data-specific knowledge. The RL agent will conduct both random search and meta-path guided search to find potential interesting item nodes in the external information network environment.

Moreover, meta-paths based approaches have been popularly used to make recommendations in heterogenous information networks [13, 14, 15]. However, these existing approaches usually assumed that meta-paths are given by experts, while very few work discuss how to generate quality meta-paths [16]. As the proposed reinforcement user profiling approach can be regarded as a meta-path generating approach, this paper also bridges the gap of current research through combining both expert and data-specific knowledge to generate quality meta-paths. To make use of the meta-paths, this paper proposes an approach to construct path based user profiles that record each user's potential preference weights to items following a set of meta-paths. Based on the path based user profiles, a user-based collaborative filtering approach is proposed to make Top-N recommendations.

The major contributions of this paper are as follows.

- This is the first work to use reinforcement learning methods for user profiling and recommendation in heterogenous information network environment.
- This work combines both expert knowledge and data-specific knowledge to learn a policy network for an RL agent to generate user profiles.
- This work bridges the gap of lacking effective approaches to generate quality meta-paths in heterogenous information networks.
- This work proposed neighbourhood based collaborative filtering recommendation approach based on the generated meta-paths and user profiles.

The rest of the paper is organized as follows. In Section 2, the related work will be briefly reviewed. Then the proposed approaches will be discussed in details in Section 3. In this section, the problem definition will be given first. Followed by the detailed discussion of the proposed reinforcement user profiling framework, the training process will be described. In Section 4, the experiments and results will be discussed. The conclusions will be given in Section 5.

2 Related Work

Recommender systems have been an active research area for more than a decade, with the main focus being recommendation approaches based on explicit ratings. Popular recommender systems applications include predicting ratings and recommending items to a user. Rating prediction is the task of predicting the rating a user will give to an item, while item recommendation is the task of recommending a set of unobserved/unrated or new items to a target user [1]. Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are widely used to measure the accuracy of rating predictions, while precision and recall are commonly used to evaluate Top-N item recommendation. For explicit ratings, both tasks are applicable, while for implicit ratings, Top-N recommendation is more applicable [1]. Recommender systems can be broadly classified into three categories: content-based, collaborative filtering (CF), and hybrid approaches [1]. *neighborhood* methods [1] and latent factor methods [4] are the two primary collaborative filtering approaches.

User profiling is the base of personalization and recommendations [1]. A typical user profile of recommender systems consists of a set of items rated or preferred by the user together with ratings to these items. However, not all users like to be involved to vote or rate explicitly. Thus, explicit ratings are not always available or applicable in real life applications [1]. Implicit ratings are important information sources for user profiling and recommendation generation [1, 17]. Rather than directly use rating information, many approaches such as MF use learned latent representations to profile users. Besides explicit or implicit ratings, other side information about users and items such as item content, genres, taxonomy categories, tags, review, social media, and social networks are popularly available. Together with users and items, all these information form heterogenous information networks [3].

Meta-paths based approaches are popularly used to make recommendations in heterogenous information networks [3]. The meta-path similarity measure framework [13] of heterogenous information networks provides a powerful mechanism for a user to measure the possibility of an unobserved user-item interaction in the information network under different semantic assumptions. The work [3] proposed a duel similarity regularisation to integrate the similarity information of users and items based on different semantic meta-paths. Link prediction has been an important problem in network modelling and has recently been studied in social network, genetic interaction network, literature citation networks, and recommender systems [5]. Some work [5] considered the user-item interactions in graphs and employ link prediction approaches to explore transitive user-item associations. However, how to profile users and make recommendations based on heterogenous information networks still need to be explored. Moreover, in many meta-path based approaches in heterogenous information networks, meta-paths are assumed to be given by experts. Very few work has discussed how to generate quality meta-paths [16]. This work [16] discussed an approach to generate data-specific meta-paths from data. How to combine both expert knowledge and data-specific knowledge to generate quality meta-paths remains an open research question.

Deep learning techniques [8] have recently emerged as a promising framework for automatically training models in various tasks such as object detection, speech recognition, and language translations over large-scale high-dimensional data (e.g., images, text, and audio) [8]. Deep learning has been applied in recommender systems to take the side information of user-item rating matrix into consideration [18, 19]. For example, more recently, Cheng et al. [20] proposed to apply wide and deep network to make explicit rating predictions. The side information is wide network. Wang et al. [21] proposed a RippleNet to consider the hop-n item knowledge graph to propagate user preferences to unknown items. RippleNet only considers the item network that formed by item related features, ignored the interactions of users and items. However, these user models that represented by a learned latent vector fail to model the decision making process of each user explicitly, for example, either a user likes items with specific topics, or because her friend likes that item. More recently, Wang et al. [22] proposed a graph attention network to take all the hop-n connections between user and item nodes into consideration. The input graphs of both RippleNet [21] and knowledge graph attention network [22] are expanded based on the original graphs, which challenge the scalability of these approaches.

Deep reinforcement learning revolutionises the field of AI and represents a step towards building autonomous systems [8]. Value based and policy based approaches are two main approaches to solve RL problems [8]. Different with traditional CF and deep neural network based approaches that usually optimise one recommendation process, reinforcement learning usually optimise a sequence of recommendation decisions for a long term goal [9, 23, 10, 11]. The majority of reinforcement learning based recommender system are model free approaches, which typically requires lots of interactions with the environment in order to learn a good policy [24, 25]. For example, Heocharous et al. [26] discussed a personalised Advertisement recommendation systems for life-time value optimisation with off policy evaluation guarantees. Zhou et al. [11] proposed a MDP-based solution to track user's interests shift and directly optimise both instant metrics and delayed metrics of user engagement. As an online user will quickly abandon the service if the recommendation looks random and do not meet his/her interests, model-based RL approaches has been proposed to avoid the large sample complexity of model-free approaches [12]. For example, more recently, Chen et al. [12] proposed a model based RL approach to learn a user model based on page view and interaction patterns. One common drawback of existing approaches is that they set the target user as the environment and ignored the fact that the target user is connected to a heterogeneous information network.

Different with existing reinforcement learning based recommender systems, this paper defines the environment as a heterogenous information network that formed by users, items, and other information such as item content, genres, taxonomy categories, tags, review, friends, occupations, and social networks. Reinforcement learning has been used in knowledge graphs to conduct relation reasoning tasks such as link prediction and fact prediction in knowledge graphs [27]. DeepPath [27] is a reinforcement learning method for knowledge graph reasoning. It used a policy-based agent with continuous states based on knowledge graph embeddings [28, 29, 30]. DeepPath [27] outperformed path ranking approach and knowledge graph embedding methods [28, 29] in the tasks of link prediction and fact prediction in knowledge graphs. However, relation reasoning task is to find a set of paths that are equivalent to a given type of relation, it is different with recommendation generation task that predicting which item a user will be interested. Thus, it is not applicable to directly apply relation reasoning approaches to make recommendations. This paper proposes to apply deep reinforcement learning to the area of user profiling in heterogenous information networks and combine both expert and data-specific knowledge to profile users, generate meta-paths, and make recommendations.

3 Proposed Approach

In this section, the problem definition will be given first. Then the proposed reinforcement user profiling framework and training process will be discussed. After that, the recommendation generation process will be discussed.

3.1 Problem Definition

To describe the proposed approach, we define some key concepts used in this paper:

- **Network schema**. *A network schema* $\mathbb{C} = (\mathbb{T}, \mathbb{R})$, *consists of a set of node types* $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_m\}$ *and a set of relation types* $\mathbb{R}$. $\mathbb{R}$ includes relations among node types, i.e. $\mathbb{R} = \{\mathcal{R}_{ij} | i = 1...|\mathbb{T}|, j = 1...|\mathbb{T}|, i \neq j\}$, where $\mathcal{R}_{ij}$ denotes the connection relation of type $\mathcal{T}_i$ and $\mathcal{T}_j$. If the number of node types $|\mathbb{T}| > 1$, the network is called *heterogeneous information network*; otherwise, it is called a *homogeneous information network.* $\mathcal{R}_{ij}^{-1}$ denote the reversed relation of $\mathcal{R}_{ij}$. Note this paper do not differentiate the term "network" and "graph".
- **Information network.** *An information network is defined as a directed graph* $G = (\mathcal{V}, \mathcal{E})$ *with a network schema* $\mathbb{C}$. Each node $v \in \mathcal{V}$ belongs to one particular node type of $\mathbb{T}$. Each edge $e \in \mathcal{E}$ belongs to a particular type of relation of $\mathbb{R}$.
- **Meta-Path.** *A meta-path* $\mathcal{M}$ *is a sequence of node types and relation types in an information network schema* $\mathbb{C}$. A meta-path $\mathcal{M} = (ij...kl) = \mathcal{T}_i \overset{\mathcal{R}_{ij}}{\rightarrow} \mathcal{T}_j ...\mathcal{T}_k \overset{\mathcal{R}_{kl}}{\rightarrow} \mathcal{T}_l$. A **path** is a sequence of nodes and relations in an information network G. The **length** of $\mathcal{M}$ (denoted as $|\mathcal{M}|$) is the number of relations (i.e., **hops**) in $\mathcal{M}$.

In recommendation scenario, users and items are two basic node types in an information network G. **Users:** $U = \{u_1, u_2, ..., u_k\}$ is the set of all users in an online community. **Items** (e.g., Products or Businesses): $P = \{p_1, p_2, ..., p_z\}$ is the set of all items rated by users in U. The explicit and implicit rating behaviour form user-item relationship $\mathcal{R}_{UP}$ and connected these two types of nodes in G. For simplicity, only binary implicit ratings are discussed in this paper.

The relationship $\mathcal{R}_{UP}$ is the basic relation type that describes whether a user is interested in an item or not. Based on this relation type, we can construct user profiles. Let $\vec{u}_i$ denote the *rating based user profile* of u_i, for example in the form of a set of items with binary ratings,

$\vec{u}_i = \{(p_1, r_{i1}), (p_2, r_{i2}), ..., (p_z, r_{iz})\}$, where $r_{ij} \in \{0, 1\}$. If $r_{ij} = 1$, then there is an edge (i.e., link) between nodes u_i and p_j in G, otherwise, there is no edge between them. Graph G is an unweighted graph.

From the perspective of rating prediction, for a target user u_i and an unobserved item p_j, the recommendation task is to predict or infer whether there is a **hop**-$n(n > 1)$ path between them in G.

Example 1: The left graph of Figure 1 shows an example heterogeneous information network. It contains three types of nodes: Users $U = \{u_1, u_2, ..., u_3\}$, Items $P = \{p_1, p_2, ..., p_4\}$ and Tags $T = \{t_1, t_2, ..., t_4\}$, six types of **hop**-1 relations, $\mathbb{R} = \{\mathcal{R}_{UP}, \mathcal{R}_{UP}^{-1}, \mathcal{R}_{UT}, \mathcal{R}_{UT}^{-1}, \mathcal{R}_{TP}, \mathcal{R}_{TP}^{-1}\}$. The example recommendation task is to predict whether user u_1 is interested in an unobserved item p_3. From the perspective of graph analysis, it is to predict whether there is a **hop**-$n(n > 1)$ link between node u_1 and p_3. The *rating based user profile* of u_1 is $\vec{u}_1 = \{(p_1, 1), (p_2, 1), (p_3, 0), (p_4, 0), (p_5, 0), (p_6, 0)\}$.

3.2 Reinforcement User Profiling Framework

For a target user u_i and an item p_j in information network G, the **hop**-$n(n > 1)$ paths finding problem is considered as a sequential decision making process in this paper. This process starts from node u_i, then the decision making model decides which relation to follow and transit to another node (i.e., intermediate neighbour node), repeat this process until finding the target item p_j or exceeding maximum allowed steps. If the decision making model is effective, we can find and recommend correct potential interesting items for each target user u_i.

In this paper, user profiling is defined as a process of learning a decision making model for each target user u_i. Based on a user profile, we learn a decision making model. The difference between user profiling and recommendation process is that the former is the model learning process while the latter is the model application process. More specifically, we define user profiling process as a sequential decision making problem. It can be solved with a Reinforcement Learning (RL) agent.

The proposed reinforcement learning framework includes the external environment and the RL agent $\mathcal{A}$. The external environment is the heterogenous information network G defined under $\mathbb{C}$. The interaction between the RL agent and the environment can be modelled as a Markov Decision Process (MDP). Every state and action pair receives an immediate reward at the state transitions. To solve an RL problem is to find the best policy for it. In a MDP, a policy describes the general rules of which actions for an agent to perform at each time step, given a history of state and action pairs since time $\tau = 0$.

The MDP aims to optimise the long term overall rewards for the entire process. The MDP is defined as $< S, A, \mathcal{P}, \Re >$, where $S = \{s_1, s_2, ..., s_n\}$ indicates states, $A = \{a_1, a_2, ..., a_l\}$ indicates actions, $\mathcal{P}$ indicates the transition probability from one state s_τ to the next state $s_{\tau+1}$ after taking action a_τ at time τ, $\mathcal{P} = \mathscr{P}(s_{\tau+1}|s_\tau, a_\tau)$, and $\Re$ indicates the immediate rewards when transitioning from state s_τ to $s_{\tau+1}$ by taking action a_τ at time τ. The details of these components are discussed as below.

Actions: The action space is defined as the set of relation types $\mathbb{R} \in \mathbb{C}$. For a given user u_i and each observed item p_j in a user profile, the agent is expected to learn which relation types and nodes to follow to find item p_j. In other words, the agent is expected to find the most informative or predictive paths linking these two nodes. After taking an action $a \in A$, the transition probability $\mathcal{P}$ will decide which state the agent will move to. Starting from u_i, the sequence of nodes and relation types that reaches p_j is an success episode of u_i and p_j. The sequence of nodes and relation types forms an episode path, which is defined as below.

Episode Path. *An episode path is a sequence of nodes in an information network G and a sequence of relation types in an network schema* $\mathbb{C}$. For a given pair of nodes v_i and v_l, and a given meta-path $\mathcal{M} = (ij...kl)$ defined in schema $\mathbb{C}$, an episode path E is defined as $E =< v_i, \mathcal{R}_{ij}, v_j, ..., v_k, \mathcal{R}_{kl}, v_l >$. Let e_{ij} denote the edge between node v_i and v_j with relation type $\mathcal{R}_{ij}$, episode path E also can be defined as a set of nodes and edges in G, $E = v_i \overset{e_{ij}}{\rightarrow} v_j...v_k \overset{e_{kl}}{\rightarrow} v_l$.

States: The external environment is information network G and its schema $\mathbb{C}$. Each state captures the agent's position in the information network G. The nodes and types of edges (i.e., relations) in G are naturally discrete atomic symbols. To capture the latent semantic information of these symbols, translation-based embeddings such as TransE [28] and TransH [29] are used to learn the latent representations of the nodes and relation types. Each node and relation type is represented by a k-dimensional continuous vector. After taking an action, the agent will move from one node to another. Embedding the goal (e.g., the target node) in the training process is a common practice in reinforcement learning [31] to reduce the time needed to train the agent. As how far the agent is from the target node can help to identify the position of the agent, both the current node and the distance between the target and current node are considered [27]. Given an node pair (v_s, v_t), let $\mathbf{S}(v_\tau)$ denote the state of current node v_τ, the state $\mathbf{S}(v_\tau)$ is defined as the concatenation of the latent vectors of current node v_τ and the distance between target node v_t and current node v_τ.

$$s_\tau = \mathbf{S}(v_\tau) = (\mathbf{v}_\tau, \mathbf{v}_t - \mathbf{v}_\tau) \tag{1}$$

Where $\mathbf{v}_\tau$ denote the latent vector of the current node v_τ and $\mathbf{v}_t$ denote that of the target node v_t. Note, this setting is for offline training environment when a (user, item) pair is given. In online training environment, the state is defined as the current state $s_\tau = \mathbf{v}_\tau$.

Rewards: To encourage the agent to find predictive episode paths, the proposed reward functions considers both the global reward r_g and the path efficiency reward r_e. For a given user u_i and an observed item p_j, starting with u_i, if the agent reaches p_j , then the agent will be given a positive global reward $r_g = +1$ for this success episode path. If the agent fails to reach p_j, then the agent will be given a negative global reward $r_g = -1$ for this failed episode path. As short paths tend to provide more reliable reasoning evidence than longer paths [27], the efficiency reward is defined as $r_e = \frac{1}{|E|}$, where $|E|$ is the length of an episode path E. The reward is the linear combination of these two parts. It is defined as below:

$$\Re = \alpha * r_g + (1 - \alpha) * r_e \tag{2}$$

Where α is parameter, $0 \le \alpha \le 1$

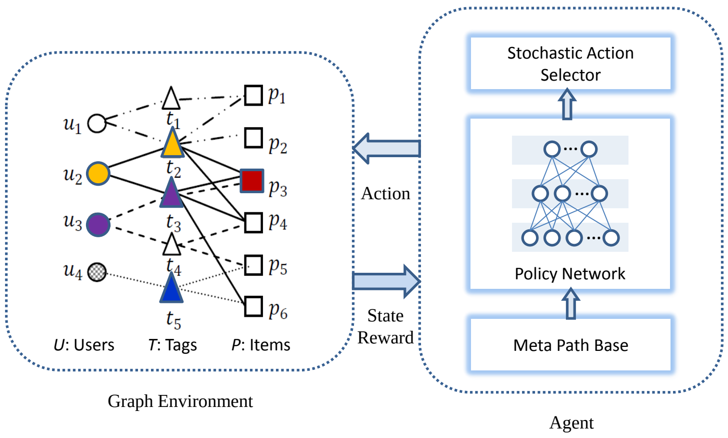


Fig. 1: The framework of the proposed approach

Agent: The agent starts with u_i node and interacts with the environment and learns a decision making model to decide which action to take (i.e., which relation type to follow) to find important intermediate transition nodes that lead to the target item p_j. Random search is popularly used for RL agent to find paths in environment [27]. This strategy will find data-specific paths. On the other hand, meta-paths contain semantic meanings and expert knowledge for finding useful nodes [13]. We can search based on meta-paths provided by experts. To take the advantages of both approaches, a Meta-Path Base $\mathbb{M}$ is introduced to store meta-paths given by experts and learned from data. We use a policy network to represent a decision making model. The agent $\mathcal{A}$ is mainly represented by a policy network [7], a Meta-Path Base $\mathbb{M}$, and a stochastic action selector. Let θ denote a set of parameters, the policy network π_θ is a probability function that maps states S to actions A. For a given state and action pair $(s,a), s \in S, a \in A$, $\pi_\theta(s,a) = \mathscr{P}(a|s;\theta)$, where $\mathscr{P}$ denote the probability distribution.

In this paper, a fully-connected neural network is used as the policy function. The input layer has the same number of dimensions with a state $s \in S$, the output layer is normalised using a softmax function over the action space (i.e., the relation type $\mathbb{R}$ space). The parameter θ is randomly initialised and will be learned during the user profiling process through the interaction between the agent and the environment.

Example 2: Figure 1 shows the proposed reinforcement user profiling framework. It contains the graph environment and the RL agent. The RL agent has Meta-Path Base $\mathbb{M}$, policy network π_θ and a stochastic action selector. The agent interacts with the graph environment, take actions and get rewards. The action space $A = \{\mathcal{R}_{UP}, \mathcal{R}_{UP}^{-1}, \mathcal{R}_{UT}, \mathcal{R}_{UT}^{-1}, \mathcal{R}_{TP}, \mathcal{R}_{TP}^{-1}\}$, Assume meta-path $\mathcal{M} = (UTP)$, $E =< u_1, \mathcal{R}_{UT}, t_2, \mathcal{R}_{TP}, p_1 >$ is one episode path guided by the meta-path $\mathcal{M}$.

3.3 Training Process

To train an effective policy network, usually a large amount of training data or episodes is needed. It usually will take a very long time to converge, if we directly train the RL agent by trial and errors. The training process will be guided by both expert knowledge and data-specific knowledge. The RL agent will conduct both random search and meta-path guided search to find potential interesting item node in external information network environment.

Lifelong Machine Learning or Lifelong Learning is an advanced machine learning paradigm that learns continuously, accumulates the knowledge learned in the past, and uses/adapts it to help future learning and problem solving [32]. A Meta-Path Base is introduced to store meta-paths given by experts and learned from data. To make use of the learned meta-paths knowledge and support life-long learning [33], this paper proposes a multi-iteration training process to train the RL agent. The generated meta-paths at each iteration will be stored in Meta-Path Base. The updated Meta-Path knowledge base will be used to train the RL agent at the next iteration until no new knowledge or performance gain achieved. Note Multi-iteration training is different with multi-epoch training. The former one is to update the Meta-Path Base while the latter one is to learn a policy network with a given Meta-Path Base.

The Meta-Path Base $\mathbb{M}$ is initialised to a set of meta-paths given by experts. At each iteration, similar to AlphaGo [7] and deepPath [27], we first train a supervised policy network based on experts moves. Then we use reward functions to retrain the supervised policy network. At the end of each iteration, the prediction accuracy of the training process will be evaluated. The learned popular new meta-paths are merged with existing meta-paths in $\mathbb{M}$. The training process continues until it reaches the predefined maximum iteration number or the prediction accuracy stops to improve. Firstly, the Pre-training and Reinforcement Learning at one iteration will be discussed, then follows the multi-iteration training process.

3.3.1 *Pre-training*

For a given user u_i, we firstly get a positive sample data $D_p = \{< u_i, p_j >\}$ from this user's profile. For each user-item pair $< u_i, p_j >$, we need to get a set of success episode paths to guide the training of the policy network. Each success episode E starts with u_i and ends with p_j. A commonly used approach to find a path in a graph is to use randomised breadth-first search(BFS) [27]. However, in heterogenous information networks, meta-paths that given by experts usually reflect the semantics of information networks. To incorporate the semantics of an information network from the viewpoint of experts, this paper proposes to find success episodes following meta-paths pre-defined by experts, as well as by randomised breadth-first search.

For an success episode path $E =< u_i, \mathcal{R}_{ul}, v_l, ..., v_k, \mathcal{R}_{kj}, p_j >$, the state of an agent is initialised to the state of the starting node u_i, $s_0 = \mathbf{S}(u_i)$. We can get the sequence of agent states $S_E =< s_0 = \mathbf{S}(u_i), s_1 = \mathbf{S}(v_l), ..., s_{|E|} = \mathbf{S}(p_j) >$, and actions $A_E =< a_0 = \mathcal{R}_{ul}, ..., a_{|E|-1} = \mathcal{R}_{kj} >$ based on each success episode path E. At each time step τ, $\tau = 0, 1, ..., |E| - 1$, the policy network that parameterised with θ will predict the action a_τ based on s_τ and get the prediction probability $\pi(a_\tau = \mathcal{R}_\tau | s_\tau; \theta)$. We maximise the expected cumulative reward using Monte-Carlo Policy Gradient [34]:

$$J(\theta) = \mathbb{E}_{a_\tau \sim \pi(a_\tau|s_\tau;\theta)}(\sum_\tau \Re_{s_\tau, a_\tau}) = \sum_\tau \sum_{a_\tau \in A} \pi(a_\tau|s_\tau;\theta) \Re_{s_\tau, a_\tau} \tag{3}$$

where $J(\theta)$ is the expected total rewards for one episode, $\tau = 0, 1, ..., |E| - 1$, $\Re_{s_\tau, a_\tau}$ is the reward after taking action a_τ. For supervised learning, we give a positive reward of $+1$ for each step of a successful episode E, $\Re_{s_\tau, a_\tau} = r_g = +1$.

The approximated gradient used to update the parameter θ of policy network is shown below:

$$\nabla_\theta J(\theta) = \sum_\tau \sum_{a_\tau \in A} \pi(a_\tau|s_\tau;\theta) \nabla_\theta \log \pi(a_\tau|s_\tau;\theta) \approx \nabla_\theta \sum_{\tau=0}^{|E|-1} \log \pi(a_\tau = \mathcal{R}_\tau|s_\tau;\theta) \Re_{s_\tau, a_\tau} \tag{4}$$

The parameter θ is batch updated after each success episode path. The detail of the pre-training process is shown in Algorithm 1.

3.3.2 Reinforcement Learning

After the pre-training process, we use the reward function to retrain the policy network and further update parameter θ in reinforcement learning process. For a user u_i, the agent will start with initial state s_0, and predict the probability distribution of each action $a \in A$ with $\pi(a_\tau|s_\tau;\theta)$ at each time step τ. Different with pre-training process, the agent employs a stochastic selection policy to randomly select an action based on the predicted probability distribution over all actions.

The sequence of actions may form a success episode path, or it may fail. The agent will receive negative rewards for those failed steps. We penalise the failed actions with negative rewards $\Re_{s_\tau, a_\tau} = r_g = -1$. As the agent selects actions based on a stochastic policy, the agent will not get stuck by repeating an incorrect step.

To improve the training efficiency, an upper bound n_{max} is set to limit the episode path length. The episode ends if the agent fails to reach the target node within n_{max} steps. After each success episode, the policy network will be updated using Equation 4 with $\Re_{s_\tau, a_\tau} = \alpha * r_g + (1 - \alpha) * r_e$. The detail of the reinforcement training process is shown in Algorithm 2.

Algorithm 1: Pre-training Procedure

```
Input:
- Graph $G$, Graph schema $\mathbb{C}$, Meta-Path Base $\mathbb{M}$
- Maximum number of path search $m_{max}$
- Latent vectors of nodes $\mathcal{V}$ and relations $\mathcal{E}$
- Positive data samples $D_p = \{(v_s, v_t)|v_s, v_t \in \mathcal{V}, v_s \neq v_t\}$
Output:
- policy network parameter $\theta$

1:Random initialise policy network parameter $\theta$ // Initialization
2:For each node pair $(v_s, v_t) \in D_p$:
3:    $\mathcal{E} \leftarrow \{\}$ // Initialise successful episode set
4:    For each $\mathcal{M} \in \mathbb{M}$:
5:        Search $G$ based on $\mathcal{M}$
6:        Get successful episodes $E_M$, $\mathcal{E} \leftarrow \mathcal{E} \cup E_M$
7:    For $i = 1$ to $m_{max} - |\mathbb{M}|$
7:        Search $G$ with random walk
8:        Get successful episodes $E_W$, $\mathcal{E} \leftarrow \mathcal{E} \cup E_W$
9:    For each path $E =< v_s, \mathcal{R}_0, v_1, ..., \mathcal{R}_{|E|-1}, v_t >\in \mathcal{E}$
10:       Get states $S_E =< s_0, s_1, ..., s_{|E|} >$ // $s_0 = \mathbf{S}(v_s)$
11:       Get actions $A_E =< a_0, a_1, ..., a_{|E|-1} >$
12:    For $\tau = 0, ..., |E| - 1$:
13:       Get prediction probability $\pi(a_\tau = \mathcal{R}_\tau|s_\tau;\theta)$
14:    //Batch update after each successful episode
15:    Update $\theta$ with gradient $g \propto \nabla_\theta \sum_\tau \log \pi(a_\tau = \mathcal{R}_\tau|s_\tau;\theta) * (+1)$
```

Algorithm 2: Reinforce Training Procedure

```
Input:
- Graph $G$, Graph schema $\mathbb{C}$
- Latent vectors of nodes $\mathcal{V}$ and relations $\mathcal{E}$
- Data samples $D = \{(v_s, v_t)\}$
- Policy network parameter $\theta$
- Maximum length of an episode $n_{max}$
Output:
- Policy network parameter $\theta$

1:For each node pair $(v_s, v_t) \in D$://
2:    //Initialisation
3:    State vector $s_0 = \mathbf{S}(v_s)$
4:    Negative instance path $E_- \leftarrow \{\}$, $Success \leftarrow False$
5:    Episode $E \leftarrow \{< v_s >\}$, episode step $\tau = 0$
6:    //Reinforce exploration
7:    While $\tau < n_{max}$ do:
8:        Get prediction probability $\mathcal{P} = \pi(a_\tau|s_\tau;\theta)$
9:        //Stochastic policy selection
10:       Randomly sample $a_\tau \sim A$ based on $\mathcal{P}$
11:       Observe reward $\Re_\tau$ and next state $s_\tau$
12:       $E \leftarrow E \cup \{< \mathcal{R}_\tau, v_\tau >\}$
13:       If $\Re_\tau = -1$ then: // Failed
14:           $E_- \leftarrow E_- \cup < v_\tau, \mathcal{R}_\tau, v_{\tau+1} >$
15:       If $\Re_\tau = 1$ then: // Success
16:           $Success \leftarrow True$
17:           break
18:       $\tau \leftarrow \tau + 1$
19:    If $E_- \neq \{\}$ then: //Penalise negative steps $E_-$
20:       Update $\theta$, $g \propto \nabla_\theta \sum_{\tau=0}^{|E_-|-1} \log \pi(a_\tau = \mathcal{R}_\tau|s_\tau;\theta) * (-1)$
21:    If $Success$ then: //Update $\theta$ after each successful episode
22:       $\Re = \alpha_1 * r_g + \alpha_2 * r_e$
23:       Update $\theta$, $g \propto \nabla_\theta \sum_{\tau=0}^{|E|-1} \log \pi(a_\tau = \mathcal{R}_\tau|s_\tau;\theta) * \Re$
```

3.3.3 Multi-iteration Training

To support life-long learning [32] and making use of the generated meta-paths, the Meta-Path Base $\mathbb{M}$ is introduced to store all the predefined or learned meta-paths. At iteration $i = 0$, $\mathbb{M}$ is initialised to the meta-paths given by experts. For a given success episode path $E =< u_i, \mathcal{R}_{il}, v_l, ..., v_k, \mathcal{R}_{kj}, p_j >$, if we keep the relations in order, we can get a meta-path $\mathcal{M} = (il...kj)$. At each iteration, we can get a set of generated meta-paths. We update the Meta-Path Base $\mathbb{M}$ at the end of each iteration. Let $\mathcal{M}_{DR}$ denote the meta-path set that generated at iteration I. As short meta-paths are computationally more efficient than long meta-paths, we rank the meta-paths of $\mathcal{M}_{DR}$ based on their length and frequency. For a meta-path $\mathcal{M} \in \mathcal{M}_{DR}$, the rank is calculated by the Equation below:

$$rank = \beta \frac{1}{|\mathcal{M}|} + (1 - \beta) f_M \tag{5}$$

Where β is a parameter, $0 \leq \beta \leq 1$, $|\mathcal{M}|$ denote the length of $\mathcal{M}$ and f_M denote the frequency of $\mathcal{M}$ in $\mathcal{M}_{DR}$. A sub set of frequent short meta-paths are selected to update the Meta-Path Base $\mathbb{M}$. At the end of each iteration, we can measure the prediction accuracy of the training process. The training process continues until it reaches the pre-defined maximum

Algorithm 3: Multi-iteration Training

```
Input:
- Meta-Path Base $\mathbb{M}$
- Maximum number of iteration $i_{max}$
Output:
- Meta-Path Base $\mathbb{M}$

1:Initialise Meta-Path Base $\mathbb{M}$ // $\mathbb{M} \leftarrow \{\}$ or $\mathbb{M}$ is provided by experts
2:Initialise iteration $I = 0$
3:**While** $I < i_{max}$ do:
4:    $\mathcal{M}_{DR} \leftarrow \{\}$ //Initialise meta-path set $\mathcal{M}_{DR}$ at iteration $I$
5:    Conduct pre-training with Algorithm 1
6:    Conduct reinforcement training with Algorithm 2.
7:    **If** accuracy at iteration $I$ improves:
8:        Get the meta-path set $\mathcal{M}_{DR}$ generated by policy network.
9:        Rank all the meta-paths of $\mathcal{M}_{DR}$ based on Equation 5
10:       //Add the selected meta-paths $\mathcal{M}^s_{DR}$ of $\mathcal{M}_{DR}$ to Meta-Path Base.
11:       $\mathbb{M} \leftarrow \mathbb{M} \cup \mathcal{M}^s_{DR}$
12:   **Else**:
13:       **break**
14:   $I = I + 1$
```

iteration number i_{max} or the accuracy performance stops to improve. The detailed multi-iteration training process is shown in Algorithm 3.

3.3.4 Time complexity Analysis

This subsection discusses the time complexity of the proposed approach. From Algorithm 3, we can see that the multi-iteration training process is controlled by the maximum iteration i_{max} and an early stopping mechanism based on the accuracy performance. Each iteration of training includes pre-training and RL-training.

In the stage of pre-training, the time complexity is mainly dependent of the two search functions: meta-path based search and random walk based search. For a given user-item pair, let n be the average number of out neighbour nodes per node, the time complexity of finding a path between the given user-item pair for meta-path based search can be calculated by the multiplication of the adjacency vectors of each relation along a given meta-path $\mathcal{M}$. The longer the meta-path is, the more time is needed. Let l be the averaged meta-path length of Meta-Path Base $\mathbb{M}$, the total time cost of searching the graph environment based on $\mathbb{M}$ is $|\mathbb{M}| \times n^l$.

For random walk based search, the time complexity in the worst case is $O(|\mathcal{V}| + |\mathcal{E}|)$ for graph environment G. Let w denote the averaged length of path that generated by random walk based search, the time complicity of random walk based search is n^w. Let m_{max} be the total number of search attempts between the given user-item pair. As the proposed approach has both meta-path based search and random search, the total time cost is $(m_{max} - |\mathbb{M}|) \times n^w + |\mathbb{M}| \times n^l$. In the implementation, we can decrease the time complexity by using bi-directional search. As long paths are usually computationally expensive and less explainable, we can set up a maximum path length to avoid generating very long paths.

In the stage of RL-training, the time complexity is mainly dependent of the time cost of 1) conducting policy based search and 2) penalising negative steps. The former one is controlled by the maximum step n_{max} and the time cost is close to a meta-path based search. The latter one is dependent of the size of negative path set E_{-}. Usually the size of E_{-} is big at the beginning of RL training. With the increasing of each iteration, the size of $\mathbb{M}$ increases while the size of negative path set E_{-} usually decreases, after a better policy network is trained. Thus, the time complexity of each iteration is not always the same. For the test stage, the time complexity is the same with meta-path based search following meta-path Base $\mathbb{M}$.

3.4 Recommendation Generation

After the training process, we can predict ratings. Based on the generated meta-paths, we can make Top-N recommendations. This section discusses how to make recommendations based on the proposed reinforcement user profiling model. Both rating prediction and Top-N recommendation will be discussed.

3.4.1 Rating prediction

As the user profiling process learned the predictive path between a given user and all the observed items, we can directly apply the policy network to predict whether a user is interested in an unobserved item (i.e., whether there is a link between them). Let v_* be the last node of an episode $< u_i, \pi(a_1|s_1;\theta), ..., v_* >$ decided by policy network π_θ, let $\mathcal{I}$ be a function, if $v_* = p_j, \mathcal{I}(v_*, p_j) = 1$; otherwise, $\mathcal{I}(v_*, p_j) = 0$. Let $u_i \in U$ be a target user, $\mathcal{O}_i$ be the item set that the user u_i already has, the prediction between a target user $u_i \in U$ and an unobserved item $p_j \in P - \mathcal{O}_i$ can be calculated with the equation below.

$$\mathcal{L}_d(u_i, p_j) = \mathcal{I}(v_*, p_j) \tag{6}$$

3.4.2 Top-N recommendation

The *neighborhood* based collaborative filtering approaches are popularly used recommendation approaches. Comparing with matrix factorisation and deep learning models, they are simple, explainable, and easy to implement. In this paper, we discuss how to use *neighborhood* based approaches to make recommendations based on the generated meta-paths. Let p_j be a candidate item of target user $u_i \in U$, $p_j \in (P - \mathcal{O}_i)$ is one candidate items for user u_i. Let $\mathcal{L}_u(u_i, p_j)$ be the predicted score of how much the user u_i would be interested in the item p_k, the problem of Top-N recommendation is to generate a set of (i.e., N numbers of) ordered items p_l, ..., $p_m \in P - \mathcal{O}_i$ to the use u_i, where $\mathcal{L}_u(u_i, p_l) \geq ... \geq \mathcal{L}_u(u_i, p_m)$.

Neighbourhood formation is the process of generating like-minded peers (i.e., the k nearest neighbours, "k-NN") for a target user $u_i \in U$. The similarity of two users u_i and u_k can be measured by the similarity of their user profiles. The more accurate a user profile is, the higher quality the user's neighbourhood. In heterogeneous information network, the prediction score of a user u_i and item p_j can be estimated by the probability of walking from the user node to item node.

Meta-path based random walk can capture the complex semantics in heterogeneous information networks [30]. We can uses the generated meta-paths in the training process to estimate a user's preference weight to items. This paper assumes each connection(i.e., edge) between any two nodes are equally important. Let $\mathbb{M}$ denote the updated meta-paths knowledge base after the multi-iteration training process.

For a given meta-path $\mathcal{M} \in \mathbb{M}$, $\mathcal{M} = (il...kj) = \mathcal{T}_i \overset{\mathcal{R}_{il}}{\rightarrow} \mathcal{T}_l...\mathcal{T}_k \overset{\mathcal{R}_{kj}}{\rightarrow} \mathcal{T}_j$. Let $\mathscr{T}(v_i)$ be the function of obtaining the type of node v_i, the walk transition probability at time τ from one node $w_\tau = v_i$ to another $w_{\tau+1} = v_j$ is generated based on the following distribution:

$$\mathscr{P}(v_i, v_l) = \mathscr{P}(w_{\tau+1} = v_l | w_\tau = v_i, \mathcal{M})$$
$$= \begin{cases} \frac{1}{|L^{\mathcal{T}_{\tau+1}(v_i)}|} & \text{if } \mathscr{T}(v_i) = \mathcal{T}_i \text{ and } \mathscr{T}(v_l) = \mathcal{T}_l \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Where $|L^{\mathcal{T}_{\tau+1}}(v_i)|$ is the number of nodes that node v_i has linked based on relationship $\mathcal{R}_{il}$ of meta-path $\mathcal{M}$. Following a meta-path $\mathcal{M} \in \mathbb{M}$, the transition probability from start node v_i to end node v_j can be calculated by the following equation:

$$\mathscr{P}(v_i, v_j|\mathcal{M}) = \mathscr{P}(v_i, v_l) * ... * \mathscr{P}(v_k, v_j) \quad (8)$$

To calculate the preference weight of user $u_i \in U$ to item $p_j \in P$, we take the frequency of a meta-path in $\mathcal{M}$ into consideration. Let f_M be the frequency of the meta-path $\mathcal{M} \in \mathbb{M}$, $\mathcal{W}(u_i, p_j|\mathbb{M})$ be the preference weight of user u_i to item p_j, $\mathcal{W}(u_i, p_j|\mathbb{M})$ can be calculated by the weighted summation of the random walk probability from node u_i to p_j, following each meta-path $\mathcal{M}$ of $\mathbb{M}$:

$$\mathcal{W}(u_i, p_j|\mathbb{M}) = \sum_{\mathcal{M} \in \mathbb{M}} \mathscr{P}(u_i, p_j|\mathcal{M}) * f_M \quad (9)$$

Based on Equation 9, we can make recommendations. To further improve the recommendation accuracy, we can apply the popularly used user based collaborative filtering approach [17]. For this approach, each user is profiled with an extended profile that generated by meta-paths with Equation 9. We calculate the similarity of each user with other users, those items that are popularly used by nearest neighbour users will be selected as recommended items. Specifically, each user is represented by a vector of items with their preference weights, which is called *extended rating based user profile* and reflects the combination of a set of **hop**-$n(n > 0)$ User-Item relationship. Let $\tilde{u_i}$ denote the *path based user profile* based on $\mathbb{M}$, $\tilde{u_i} = \{(p_0, \mathcal{W}(u_i, p_0|\mathbb{M}), (p_1, \mathcal{W}(u_i, p_1|\mathbb{M}), ..., (p_z, \mathcal{W}(u_i, p_z|\mathbb{M}))\}$.

The distance or similarity measure can be calculated through various kinds of proximity computing approaches such as cosine similarity or Pearson's correlation. Cosine is used to measure the similarity of two users in this paper. The similarity of any two users $u_i \in U$ and $u_j \in U$ can be calculated as:

$$sim(u_i, u_j) = cosine(\tilde{u_i}, \tilde{u_j}) \quad (10)$$

The neighborhood of user u_i is denoted as $\mathcal{N}(u_i) = \{u_j|u_j \in maxK_{u_j \in U}\{sim(u_i, u_j)\}\}, u_j \in U$, where $maxK\{\}$ returns the top-k most similar users to u_i. For each target user u_i, the prediction score of how much u_i will be interested in an unobserved candidate item $p_j \in P - \mathcal{O}_i$ is calculated by considering the similarities between user u_i and those users who are neighbors of u_i and have rated item p_j [1]:

$$\mathcal{L}_u(u_i, p_j) = \sum_{u_k \in (\mathcal{N}_{u_i} \cap L_j)} sim(u_i, u_k) \quad (11)$$

Where L_j denotes the user nodes that item node p_j has linked based on **hop**-1 relation $\mathcal{R}_{UP}{}^{-1}$ (i.e., those users that has rated item p_j). The Top N items with high prediction scores will be recommended to the target user u_i. This is a user-based approach.

4 Experiments

4.1 Datasets

To evaluate the effectiveness of the proposed approaches, this work conducted rating prediction and Top-N ($N =$ [5,10,15,20,25,30,40,50,60,70,80,90,100]) recommendation experiments on the following three datasets.

D1: MovieLens-Tagging Dataset. The MovieLens 10M Dataset is a movie rating and social tagging dataset. As this paper only considers binary relations, only the tagging assignment dataset (i.e., tags.dat file, represents meta-path (UTP), or triplets $\mathcal{R}_{UTP}$) were used in the experiments. The recommended items are movies. [1] The action space $A = \{R_{UP}, R_{UP}^{-1}, R_{UT}, R_{UT}^{-1}, R_{TP}, R_{TP}^{-1}\}$.

D2: HetRec2011-MovieLens Dataset. This is an extension of MovieLens 10M dataset. It has enriched with various kinds of affiliated information about movies. This includes the following data files or relations: the tagging assignment user-taggedmovies.dat that represents meta-path (UTP), the movie-genres.dat that represents relation $\mathcal{R}_{PG}$, movie-directors.dat that represents relation $\mathcal{R}_{PD}$, movie-actors.dat that represents relation $\mathcal{R}_{PA}$, movie-countries.dat that represents relation $\mathcal{R}_{PC}$. The recommended items are movies. [2] The action space $A = \{R_{UP}, R_{UP}^{-1}, R_{UT}, R_{UT}^{-1}, R_{TP}, R_{TP}^{-1}, R_{PG}, R_{PG}^{-1}, R_{PD}, R_{PD}^{-1}, R_{PC}, R_{PC}^{-1}, R_{PA}, R_{PA}^{-1}\}$.

D3: HetRec2011-LastFM Dataset. This dataset contains social networking, tagging, and user music artist listening information from Last.fm online music system. The user-taggedartists.dat that represents meta-path (UTP), user-friends.dat that represents relation $\mathcal{R}_{UF}$ were used in the experiments. The recommended items are musical artists. [3] The action space $A = \{R_{UP}, R_{UP}^{-1}, R_{UT}, R_{UT}^{-1}, R_{TP}, R_{TP}^{-1}, R_{UF}, R_{UF}^{-1}\}$.

To reduce the sparseness in the dataset, this work filtered out those entities nodes (e.g., users, items, tags, genres, directors, actors) that have been occurred only once in the data. For each test user, an RL agent was trained. As training an RL agent is time consuming, we randomly selected 100 users as the test user set. To reduce the randomness caused by very few training samples, each test user has tagged at least 100 items. As the task is to predict or recommend items, we only keep User-Item $\mathcal{R}_{UP}$ relation in the Training Set and Test Set. For a target user, we randomly selected 40% of User-Item relation as Training set and 20% of User-Item relation as Test Set, the rest including 40% of User-Item relation and the affiliated information forms the graph Environment Set.

The statistics of the three datasets after preprocessing and the networks are shown in Table 1. D1 only contains tagging information, D2 has various kinds of affiliated information about items, and D3 has affiliated relation about users. On average, each node of D1 has 6.16 edges, while D2 has 2.88 edges, D3 has 13.15 edges.

4.2 Experimental Setup and Results

Top-N recommendation task is popularly used for implicit or binary ratings [1]. The *Precision* and *Recall* are used to measure the performance of the Top-N item recommendation task. The averaged values of all target users are used to measure the overall performance of recommendation approaches. The *HitRatio* is used to measure the accuracy of rating prediction task. It is defined as the ratio of correct rating

1. https://grouplens.org/datasets/movielens/
2. https://grouplens.org/datasets/hetrec-2011/
3. http://www.last.fm,https://grouplens.org/datasets/hetrec-2011/

TABLE 1: The basic statistics of datasets

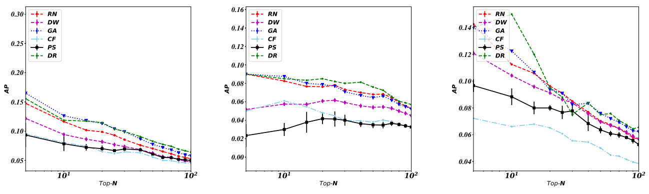
Nodes	Number	Relations	Number
D1: MovieLens-Tagging			
Users U	3,706		
Movies P	6,145		
Tags T	7,272	Triplets $\mathcal{R}_{UTP}$	85,021
Nodes $\mathcal{V}$	23,597	Edges $\mathcal{E}$	145,297
D2: HetRec2011-MovieLens			
Users U	1,887		
Movies P	4,442		
Tags T	3,994	Triplets $\mathcal{R}_{UTP}$	41,656
Movie Genres G	19	$\mathcal{R}_{PG}$	10,212
Directors D	795	$\mathcal{R}_{PD}$	3,272
Actors A	18,919	$\mathcal{R}_{PA}$	71,790
Countries C	39	$\mathcal{R}_{PC}$	4,428
Nodes $\mathcal{V}$	55,865	Edges $\mathcal{E}$	161,349
D3: HetRec2011-LastFM			
Users U	1,832		
Artists P	10,753	Triplets $\mathcal{R}_{UTP}$	179,501
Tags T	4,373	$\mathcal{R}_{UF}$	24,162
Nodes $\mathcal{V}$	17,123	Edges $\mathcal{E}$	225,234

TABLE 2: The statistics of generated meta-paths

	Length			Example MetaPaths
	Avg.	Med.	Max	
				MovieLens-Tagging
$\mathcal{M}_{DP}$	4.57	3	13	$(UPUP), (UPUTPTUPUTP)$
$\mathcal{M}^1_{DR}$	6.2	3	16	$(UTPUP)$
$\mathcal{M}^2_{DR}$	3.0	3	5	$(UPTPUP), (UTPUP)$
$\mathcal{M}^3_{DR}$	4.5	3	12	(UPUPUP)
				HetRec2011-Movielens
$\mathcal{M}_{DP}$	7.8	6	16	$(UPGPUP), (UTP), (UPGPTPUP)$
$\mathcal{M}^1_{DR}$	4.6	3.5	11	$(UTUPAP), (UPGPUP)$
$\mathcal{M}^2_{DR}$	2.4	3	3	$(UPAPGP), (UPDPAP)$
$\mathcal{M}^3_{DR}$	4.4	3	9	$(UPGPUP), (UTUPAP)$
				HetRec2011-Last.fm
$\mathcal{M}_{DP}$	5.0	4	11	$(UTUTUTP), (UPUPUFP)$
$\mathcal{M}^1_{DR}$	5.0	5	9	$(FUP), (UPTUFP)$
$\mathcal{M}^2_{DR}$	6.8	4	19	$(UFPUP), (UFTP)$
$\mathcal{M}^3_{DR}$	5.3	4	9	$(UPTUFP)$

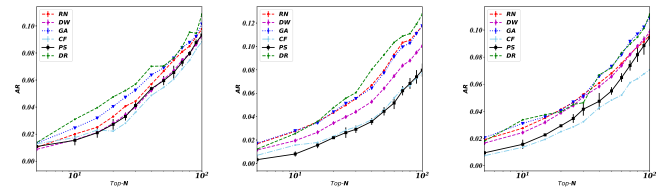
TABLE 3: Synthetic dataset

Name	Hop-1	Hop-2	Hop-3	Scale	Name	Hop-1	Hop-2	Hop-3	Scale
SD1	7,354	28,858	63,788	10^5	SD4	132,972	6,696,908	33,363,528	10^8
SD2	22,032	209,968	768,020	10^6	SD5	297,796	33,363,528	966,332,912	10^9
SD3	57,180	1,257,822	8,685,000	10^7	SD6	545,806	111,987,626	5,798,026,166	6×10^9



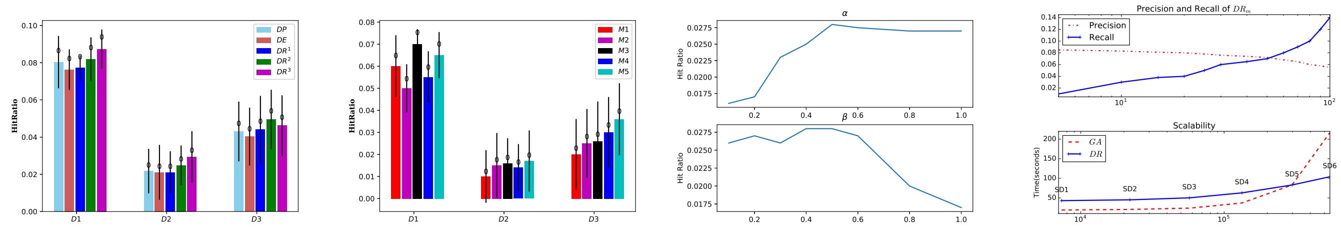
(a) D1 (b) D2 (c) D3

Fig. 2: Top-N Precision results



(a) D1 (b) D2 (c) D3

Fig. 3: Top-N Recall results



(a) Comparison with different models (b) Results of different single meta-paths (c) Influence of parameter α and β (d) Precision and Recall of DR_m, and Scalability

Fig. 4: Detailed Analysis and Scalability Results

prediction over the total number of test items of a target user. Let n_r denotes the number of correct rating predictions, n_t denotes the total number of test items of a target user, $HitRatio = \frac{n_r}{n_t}$.

In the experiments, TransE [28] approach is used to train the embeddings of each node and each relation of the graph, the embeddings were trained for 1,000 epochs. The settings of parameters are: embedding dimension $k = 20$, batch size $= 128$, maximum length of an episode $n_{max}= 20$, $\alpha = 0.5$, $\beta = 0.5$, the maximum number of iterations of the training process i_{max} was set to 3, the maximum number of search attempt m_{max} was set to 20. As discussed before, short paths are more efficient and have higher explainability than long paths. The policy network is a fully connected neural network with 4 layers in total, the setting of the number of nodes of the two hidden layers are 512 and 1024. The RL agents were trained on a desktop server with 16 GB Memory and 12 Intel Core i7-8700 CPU with 3.20GHz.

4.2.1 *Recommendation Results*

This set of experiments compared the effectiveness of the proposed approach with the-state-of-the-art Top-N recommendation based-line models. We compared the Top N recommendation of the following approaches:

- ***DR***: the proposed recommendation approach.
- ***CF***: the traditional user-based CF approach [1].
- ***PS***: the state-of-the-art recommendation approach based on meta-path based similarity [3]. For fair com-

parison, it was based on the same set of pre-selected meta-paths with $\boldsymbol{DR}$.

- $\boldsymbol{DW}$: a Top-N recommendation approach based on Deep&Wide neural network [20]. The existing work is based on explicit ratings. To make $\boldsymbol{DW}$ work for implicit ratings with only positive samples, this paper adopted the same process of Bayesian Personalized Ranking [35].
- $\boldsymbol{RN}$: the RippleNet approach [21], a recommendation approach based on a knowledge graph constructed by items and their feature entities. The input graph is an item-item hop-n graph that generated by item-feature relations. The same with [21], $n = 2$.
- $\boldsymbol{GA}$: the state-of-the-art recommendation approach based on knowledge graph attention network [22]. The input graph is an expanded graph that generated by the combination of various types of hop-n item-feature relations. The same with [22], $n = [1, 2, 3]$.

The parameter of the compared approaches are set to the most optimised values. To decrease the degree of the randomness of these approaches, the proposed approach was run 3 times. the average and standard deviation of the prediction accuracy values were used. The Top-N *Precision* and *Recall* results of these compared approaches on the three datasets are shown in Figure 2 and Figure 3. We can see that the proposed approach $\boldsymbol{DR}$ performed better than $\boldsymbol{PS}$ in both *Precision* and *Recall*. This demonstrates that the proposed recommendation approach can effectively find similar users than the path based similarity regulation based approach $\boldsymbol{PS}$.

$\boldsymbol{CF}$ performed the worst among all the compared models. This can be explained that $\boldsymbol{CF}$ only considered the typical user-item relation while the other relations that can help to find potential interesting items were ignored. $\boldsymbol{DW}$ performed better than $\boldsymbol{CF}$ but not better than the other compared models in *Recall*. $\boldsymbol{DW}$ only considered the features of items or users, but ignored the multiple relationship among users, items, and features. $\boldsymbol{GA}$ performed better than $\boldsymbol{RN}$ and $\boldsymbol{DW}$ in *Recall*. This can be explained that $\boldsymbol{GA}$ based on the combination of hop-1, hop-2, hop-3 relations, while $\boldsymbol{RN}$ is only based on hop-2 relations. Overall, the proposed approach $\boldsymbol{DR}$ performed the best. This shows that the proposed reinforcement learning based approach can effectively profile users and make recommendations.

4.2.2 *Detailed analysis of the proposed approach*

This subsection discusses the detailed analysis of the proposed approach. The proposed approach $\boldsymbol{DR}$ introduces Meta-Path Base $\mathbb{M}$ to store the pre-selected or generated meta-path and supports multi-iteration training. It considers both expert knowledge and data-specific knowledge in the training process. The first set of experiments analysed the effectiveness of the setting of Meta-path Base $\mathbb{M}$ and the effectiveness of the proposed approach from the aspect of generating meta-paths. The *HitRatio* value of $\boldsymbol{DR}$ were compared with the the following two approaches:

- $\boldsymbol{DE}$: the version of the proposed approach that only based on experts pre-selected meta-paths. Different with $\boldsymbol{DR}$, it only follows the pre-selected meta-paths. It does not support multi-iteration training.
- $\boldsymbol{DP}$: the version of the proposed approach that only based on random search. This is inspired by DeepPath [27], a reinforcement learning approach for knowledge graph relation reasoning. It does not support meta-path based search or multi-iteration training.

To decrease the degree of the randomness of these approaches, each model was run 3 times. $\mathbb{M}_I$ denotes the Meta-Path Base after the I-th iteration of training. Notation $\boldsymbol{DR^I}$ is used to denote the proposed approach after training iteration I. $\mathbb{M}_{DP}$ denotes the generated meta-path sets.

$\mathbb{M}_0$ represents the pre-selected meta-path set. Each meta-path starts with a user node and ends with an item node. The pre-selected meta-path set includes those short paths that are popularly used in literature and easy to be interpreted in terms of semantic meanings. Let $\mathbb{M}_* = \{(UTP), (UTUP), (UPTP), (UTPUP), (UPTUP), (UTUTP), (UPUTP), (UTPTP), (UP), (UPUP)\}$. This set of meta-paths are based on the tagging graph. Among them, some meta-paths can be interpreted as collaborative filtering based approaches such as $(UPUP)$, while some are content based approaches such as (UTP), $(UPTP)$, and $(UTPTP)$, or hybrid approaches such as $(UTPUP)$ and $(UPUTP)$. For Dataset D1, $\mathbb{M}_0 = \mathbb{M}_*$. For Dataset D2, a set of popular paths that contain more item feature related entity nodes were added. $\mathbb{M}_0 = \mathbb{M}_* \cup \{ (UPGP), (UPDP), (UPAP), (UPCP), (UPTP), (UPTPGP), (UPTPDP)\}$. For Dataset D3, a set of meta-paths that contain user-friends relations were added. $\mathbb{M}_0 = \mathbb{M}_* \cup \{(UFUP), (UTUFP), (UPUFP)\}$.

Table 2 shows the statistics of the generated meta-paths of $\boldsymbol{DP}$ and $\boldsymbol{DR}$ as well as some example generated meta-paths. We can see that the average lengths of generated meta-paths of $\boldsymbol{DP}$ are usually longer than $\boldsymbol{DR}$. $\boldsymbol{DP}$ can generate interesting meta-paths from the data. It generated popular meta-paths that are overlapped with expert pre-selected meta-paths $\mathbb{M}_0$, for example, $(UPUP)$ for D1 and (UTP) for D2. However, $\boldsymbol{DP}$ failed to generate some important meta-paths. For example, $(UPTUP)$, (UFP), $(UPAP)$ were not included in the generated meta-paths set of $\boldsymbol{DP}$. Instead, $\boldsymbol{DP}$ generated very long meta-paths with low interpretability such as $(UPTPUTPUTPTUP)$ for D1. The paths generated by $\boldsymbol{DR}$ are relatively short, interesting, and creative. It can find interesting data-specific meta-paths that are missed by experts. For example, $(UTUPAP)$ and $(UPGPUP)$ for Dataset D2 and (FUP) for dataset D3.

The *HitRatio* results of the 3 compared models on dataset D2 are shown in Figure 4 (a). We can see that $\boldsymbol{DE}$ sometimes performed better than $\boldsymbol{DP}$, while sometimes performed worse than $\boldsymbol{DP}$. It can be explained that if the expert-based approach $\boldsymbol{DE}$ equipped with the meta-paths that reflect the characteristics of the data, then it can achieve performances similar to or even better than $\boldsymbol{DP}$ that has random explorations. $\boldsymbol{DR}$ achieved the best performance at iteration $I = 2$ or $I = 3$. This can be explained that after updating the knowledge base or meta-paths Base with those novel meta-paths generated at previous iterations, $\boldsymbol{DR}$ can better capture the successful meta-paths that lead to those items that a user might be interested.

For dataset D1, the Top 5 most frequent meta-paths include $\mathcal{M}_1 = (UP)$, $\mathcal{M}_2 = (UTP)$, $\mathcal{M}_3 = (UPUP)$, $\mathcal{M}_4 = (UPTP)$, and $\mathcal{M}_5 = (UPUTP)$. For dataset D2, the Top 5 frequent meta-paths include $\mathcal{M}_1 = (UP)$, $\mathcal{M}_2 = (UPDP)$, $\mathcal{M}_3 = (UPUP)$, $\mathcal{M}_4 = (UPGP)$, and $\mathcal{M}_5 = (UPGPUP)$. For dataset D3, the Top 5 most frequent meta-paths include $\mathcal{M}_1 = (UP)$, $\mathcal{M}_2 = (FUP)$, $\mathcal{M}_3 = (UPUP)$, $\mathcal{M}_4 = (UFUP)$, and $\mathcal{M}_5 = (UPUFP)$. The HitRatio of these meta-path on dataset D2 are shown in Figure 4 (b). The $HitRatio$ of the proposed approach with different parameter settings of α and β for dataset D2 are shown in Figure 4 (c). The proposed recommendation approach based on Equation 9 is denoted as DR_m. The Precision and Recall of DR_m based on D2 are shown in the upper chart of Figure 4 (d). The accuracy of DR_m is slightly lower than DR. This can be explained that the user based approach DR can help to find more potentially interested items from those similar users with extended user profiles which are generated by meta-path based approach.

4.2.3 Efficiency Discussion

We compared the training and test time of $\boldsymbol{DR}$ with two recent models $\boldsymbol{RN}$ and $\boldsymbol{GA}$ on dataset D2. The batch size is set to 1024. The experiments were conducted with the same hardware and software environment. The averaged training time of 1 epoch is 19.83 seconds for $\boldsymbol{RN}$ and 426.34 seconds for $\boldsymbol{GA}$. As they require different number of epochs to converge, the total training time for $\boldsymbol{GA}$ is 22 hours and that for $\boldsymbol{RN}$ is 2 hours. On average, it took about 3 hours to train one RL agent on one core. As it is easy and natural to train multiple RL agents for different users at the same time, the training for $\boldsymbol{DR}$ was completed in 46 hours.

The averaged testing time for one test user for $\boldsymbol{RN}$, $\boldsymbol{GA}$, and $\boldsymbol{DR}$ is 0.94 seconds, 0.01 seconds, and 0.85 seconds respectively. $\boldsymbol{DR}$ has the longest training time but had good performance in the test stage. This shows that the RL agent can make quick decisions after equipped with the trained policy network. $\boldsymbol{GA}$ had very long training time as it considered multi-hop relations in the training. After the network is trained, $\boldsymbol{GA}$ can make predictions quickly with no need to explore the graph along meta-paths or propagate the graph like $\boldsymbol{DR}$ and $\boldsymbol{RN}$ did. It performed the best in test stage.

To compare the scalability of $\boldsymbol{DR}$ and $\boldsymbol{GA}$, a set of synthetic datasets is generated with various input graph size (i.e., hop-1 relations). The set of synthetic datasets is shown in Table 3. It has 6 sub datasets: SD1 to SD6. Each sub dataset contains different number of hop-1 relations, and generated hop-2 and hop-3 relations based on the hop-1 relations. The size (i.e., total number of relations) of each sub dataset is at different scale. The average training time for 1 epoch of these two approaches for different sub datasets are shown in the lower chart of Figure 4 (d). We can see that $\boldsymbol{GA}$ performed better than $\boldsymbol{DR}$ when the data size is relatively small. However, the training time of $\boldsymbol{GA}$ scales up significantly for SD6, when large number of hop-2 and hop-3 relations are generated with an increased number of hop-1 relations. The scalability of $\boldsymbol{DR}$ is better than $\boldsymbol{GA}$ for large datasets. [4]

5 Conclusion

This paper proposed a deep reinforcement user profiling approach for recommender systems in heterogenous information networks. This paper defines user profiling in heterogenous information networks as a process of learning a decision making model for each target user. A multi-iteration training process is proposed to train an RL agent to interact with the external heterogenous information network environment and receive rewards. It learns a decision making model to decide which actions to take to find potential interesting item nodes. The reward functions consider both global accuracy and efficiency to further improve the training efficiency. At each iteration, the training process combines both expert knowledge and data-specific knowledge to learn a policy network. It firstly train the policy network in a supervised way based on experts moves. Then re-train the policy network with reinforcement learning. A Meta-Path Base is introduced to store the learned knowledge (i.e., the generated meta-paths) at each iteration. The meta-paths of the updated Meta-Path Base are used to train the RL agent at the next iteration. The multi-iteration training process makes use of learned knowledge about effective meta-paths and supports life-long learning [33]. This paper contributes to a novel user profiling approach and a new application area of reinforcement learning.

Moreover, the proposed training process and RL agent can be used to generate quality meta-paths in heterogenous information networks. Although meta-paths based approaches have been popular to make recommendations in heterogenous information networks, many existing approaches fully rely on experts to provide meta-paths. This paper bridges this gap through combining expert knowledge and data-specific knowledge to generate meta-paths. It contributes to a novel meta-paths generation approach. This work proposed neighbourhood based collaborative filtering recommendation approach based on the generated meta-paths and user profiles. This paper conducted experiments on three datasets. The proposed approaches were compared with baseline models on both rating prediction task and Top-N recommendation task. This paper also analysed and discussed the quality of the generated meta-paths.

This work has some limitations. The learning process is trained and tested on offline environment, not in a real-time online mode that has dynamic environment. The future work will extend it to real-time dynamic environment and will improve the reward function, consider explicit rating, and combine the text content of nodes such as reviews with relations. Moreover, the future work will explore the following directions: 1) extend the Markov hypothesis and explore model-based RL approach; 2) further improve the explainability of user profiling agent; 3) explore item-based RL approach; 4) explore RL approaches for warm-start and cold-start problems.

4. The author would like to thank PhD student Thanet Markchom for his help in the scalability experiment.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[2] H. Liang, Y. Xu, D. Tjondronegoro, and P. Christen, "Time-aware topic recommendation based on micro-blogs," in *CIKM '12*, 2012, pp. 1657–1661.

[3] J. Zheng, J. Liu, C. Shi, F. Zhuang, J. Li, and B. Wu, “Recommendation in heterogeneous information network via dual similarity regularization,” *I. J. Data Science and Analytics*, vol. 3, no. 1, pp. 35–48, 2017.

[4] Y. Koren, R. Bell, and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[5] X. Li and H. Chen, “Recommendation as link prediction in bipartite graphs,” *Decis. Support Syst.*, vol. 54, no. 2, pp. 880–890, Jan. 2013.

[6] Z. Huang, X. Li, and H. Chen, “Link prediction approach to collaborative filtering,” in *JCDL ’05*. ACM, 2005, pp. 141–142.

[7] V. Mnih, K. Kavukcuoglu, and et. al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 02 2015.

[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[9] G. Shani, D. Heckerman, and R. I. Brafman, “An mdp-based recommender system,” *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, Dec. 2005.

[10] A. H. Nabizadeh, A. M. Jorge, and J. P. Leal, “Long term goal oriented recommender systems,” in *Proceedings of the 11th International Conference on Web Information Systems and Technologies*, 2015, pp. 552–557.

[11] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, “Reinforcement learning to optimize long-term user engagement in recommender systems,” in *KDD ’19*. ACM, 2019, pp. 2810–2818.

[12] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, “Generative adversarial user model for reinforcement learning based recommendation system,” in *ICML’19*, 2019, pp. 1052–1061.

[13] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

[14] Y. Dong, N. V. Chawla, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in *KDD ’17*. ACM, 2017, pp. 135–144.

[15] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, “Heterogeneous information network embedding for recommendation,” *CoRR*, vol. abs/1711.10730, 2017.

[16] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang, “Discovering meta-paths in large heterogeneous information networks,” in *WWW ’15*, 2015, pp. 754–764.

[17] H. Liang, Y. Xu, Y. Li, R. Nayak, and X. Tao, “Connecting users and items with weighted tags for personalized item recommendations,” in *HT’10*, 2010, pp. 51–60.

[18] H. Wang, N. Wang, and D.-Y. Yeung, “Collaborative deep learning for recommender systems,” in *KDD ’15*. ACM, 2015, pp. 1235–1244.

[19] H. Liang and T. Baldwin, “A probabilistic rating auto-encoder for personalized recommender systems,” in *CIKM ’15*. ACM, 2015, pp. 1863–1866.

[20] H.-T. Cheng, L. Koc, and et. al., “Wide & deep learning for recommender systems,” in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, ser. DLRS 2016. ACM, 2016, pp. 7–10.

[21] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, “Ripplenet: Propagating user preferences on the knowledge graph for recommender systems,” in *CIKM’18*. ACM, 2018, pp. 417–426.

[22] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, “Kgat: Knowledge graph attention network for recommendation,” in *KDD ’19*. ACM, 2019, pp. 950–958.

[23] N. Taghipour, A. Kardan, and S. S. Ghidary, “Usage-based web recommendations: A reinforcement learning approach,” in *RecSys ’07*. ACM, 2007, pp. 113–120.

[24] 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 *WWW ’18*. ACM, 2018, pp. 167–176.

[25] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, “Recommendations with negative feedback via pairwise deep reinforcement learning,” in *KDD ’18*. ACM, 2018, pp. 1040–1048.

[26] G. Theocharous, P. S. Thomas, and M. Ghavamzadeh, “Personalized ad recommendation systems for life-time value optimization with guarantees,” in *IJCAI ’15*. AAAI Press, 2015, pp. 1806–1812.

[27] W. Xiong, T. Hoang, and W. Y. Wang, “Deeppath: A reinforcement learning method for knowledge graph reasoning,” in *EMNLP ’17*. ACL, September 2017.

[28] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *NIPS ’13*, 2013, pp. 2787–2795.

[29] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *AAAI ’14*. AAAI Press, 2014, pp. 1112–1119.

[30] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, “Heterogeneous network embedding via deep architectures,” in *KDD ’15*. ACM, 2015, pp. 119–128.

[31] S. Nasiriany, V. H. Pong, S. Lin, and S. Levine, “Planning with goal-conditioned policies,” in *NeurIPS ’19*, 2019.

[32] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi, *Lifelong Machine Learning*, 2nd ed. Morgan and Claypool Publishers, 2018.

[33] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, Jr., and T. M. Mitchell, “Toward an architecture for never-ending language learning,” in *AAAI’10*. AAAI Press, 2010, pp. 1306–1313.

[34] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[35] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *UAI ’09*. AUAI Press, 2009, pp. 452–461.

Huizhi Liang Lecturer of Department of Computer Science, University of Reading, United Kingdom. Huizhi received her PhD degree from the Queensland University of Technology, Australia. Before joining University of Reading in 2017, she worked at LIP6, Pierre et Marie Curie University and French National Center for Scientific Research (CNRS), University of Melbourne, and Australian National University. Her research interests include recommender systems, data mining, and machine learning.