IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 25, 2020, accepted April 18, 2020, date of publication May 6, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991257

Reinforcement Learning Over Knowledge Graphs for Explainable Dialogue Intent Mining

KAI YANG[1], XINYU KONG[1], YAFANG WANG[1], JIE ZHANG[1], AND GERARD DE MELO[2]
[1]Ant Financial Services Group, Hangzhou 310099, China
[2]Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA

Corresponding author: Yafang Wang (yafang.wyf@antfin.com)

This work was supported by the National Natural Science Foundation of China under Grant 61503217.

ABSTRACT In light of the millions of households that have adopted intelligent assistant powered devices, multi-turn dialogue has become an important field of inquiry. Most current methods identify the underlying intent in the dialogue using opaque classification techniques that fail to provide any interpretable basis for the classification. To address this, we propose a scheme to interpret the intent in multi-turn dialogue based on specific characteristics of the dialogue text. We rely on policy-guided reinforcement learning to identify paths in a graph to confirm concrete paths of inference that serve as interpretable explanations. The graph is induced based on the multi-turn dialogue user utterances, the intents, i.e., standard queries of the dialogues, and the sub-intents associated with the dialogues. Our reinforcement learning method then discerns the characteristics of the dialogue in chronological order as the basis for multi-turn dialogue path selection. Finally, we consider a wide range of recently proposed knowledge graph-based recommender systems as baselines, mostly based on deep reinforcement learning and our method performs best.

INDEX TERMS Knowledge graph, dialogue intent mining, reinforcement learning.

I. INTRODUCTION

Across the globe, millions of households have adopted intelligent assistant powered devices. In light of this, multi-turn dialogue, in particular, task-oriented multi-turn dialogue which aims to handle with certain questions, has become an important field of inquiry with substantial real-world impact. The system not only needs to identify a user's information need from this dialogue but also locate an appropriate answer from all the knowledge that is accessible to it. Such knowledge can oftentimes be regarded as taking the form of a knowledge graph and locating an answer often corresponds to identifying relevant nodes in the graph [1].

Recent work in this area has exploited advances in neural representation learning to address this task [2], [3]. However, in real-world deployments of such systems, it is not sufficient for a multi-turn dialogue recognition system to merely use latent vector representations for knowledge graph nodes to identify appropriate responses. Rather, the system ought to be able to offer the user clear explanations of how the multi-turn dialogue led to specific intention recognition outcomes. In this paper, we consider a knowledge graph providing information such as user utterances, the sub-intents associated with the dialogues, and the standard queries of the dialogues.

We propose a method called *PGMD* that draws on a neural reinforcement learning network to navigate the knowledge graph in pursuit of the pertinent query nodes in the graph. The reinforcement learning agent starts from a user utterance from the current multi-turn dialogue and searches the knowledge graph iteratively with the goal of obtaining a precise and interpretable path in the graph for intent recognition. As the agent makes its prediction based on specific paths in the graph, we have a highly interpretable model that can easily explain the underlying process of intent recognition [4].

Thus, the goal of our paper is not only to identify the candidate sets of intentions in multi-turn dialogue, but also to provide an interpretable path in the knowledge graph that explains the process of identifying such intentions. This novel strategy yields a means of overcoming the shortcomings of current approaches. We use the intent recognition process as a Markov decision process based on a knowledge graph. Reinforcement learning is invoked for each given multi-turn dialogue, wherein the agent learns to search for the sub-intents associated with the dialogues, and finally search for the standard queries of the dialogues. The search path can

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

serve as an explanation of the dialogue intent prediction process.

The main contributions of this paper are as follows:

1) We use multi-turn dialogue data to construct a knowledge graph and train a node embedding model for this knowledge graph, which mainly includes the following types of nodes: user utterance nodes, sub-intent nodes, and standard query nodes. In light of the sparsity of the textual data, our model draws on the BERT pre-training model [5] to obtain the word representations of the user utterances to train the model.
2) We propose a reinforcement learning method for path selection called PGMD. Since multi-turn dialogue has chronological characteristics, we consider an BiLSTM (Bidirectional Long Short-Term Memory) network with attention mechanism in our reinforcement learning agent to obtain the state characteristics of the path. And we proposed a new reward to compute the macro-averaged matching score between nodes on the path with the query nodes.
3) We have designed multi-turn dialogue tracking path searching algorithms including backward tracking strategy and forward tracking strategy to find different paths as candidate sets for identified intents.

II. RELATED WORK

A. KNOWLEDGE GRAPH-DRIVEN RECOMMENDATION

The primary objective in a recommendation task is to determine the suitability of items for users that they have not yet seen or used. There are two principal ways of incorporating a knowledge graph into a recommendation engine. The first is based on a feature-driven recommendation method, and involves extracting pertinent user and item attributes from the knowledge graph as features, which can then be included into traditional models, such as the FM model, LR model, etc. [6]. The second is the path-based recommendation method. [7] considers the knowledge graph as a heterogeneous information network and then constructs meta-graph based features between items. [8] proposed a new model named KPRN which can generate path based on the semantics of entities and relations. [9] transferred the relation information in knowledge graph in order to figure out the reasons why a user prefers an item. [10] proposed a model named KGCN which can mine the associated attributes between items in knowledge graph. In particular, [11]–[14] proposed different path-based methods to get recommended results for the Linked Open Data(LOD). [12] found out the recommended path in LOD based on variable importance scores. [11] and [13] used DBpedia to extract semantic path-based features to compute the recommended results eventually. [14] made an investigation about the incorporation of graph-based features into LOD path-based systems. One advantage of this second approach is the full and intuitive use of the network structure of the knowledge graph.

In existing work, the recommendation engine is trained based on prior interactions between users and items. However, in our dialogue engine, we need to recommend appropriate query nodes based on the user dialogue, and there may not have been any prior interaction at all with any relevant items. Thus, our setting is quite different from general recommendation systems. However, we compare our algorithm against state-of-the-art recommendation engines.

B. REINFORCEMENT LEARNING

In recent years, a large number of studies in different areas have identified reinforcement learning as a promising artificial intelligence technique. Thus, reinforcement learning is not only used for standard text mining tasks such as text classification [15]. Additionally, it has also been explored for knowledge graphs of the sort mentioned above. For example, in terms of question answering, a knowledge graph may be considered as the environment for an agent. [1] used reinforcement learning whose reward function considers accuracy, diversity and efficiency to find paths in the knowledge graph and [16] proposed a multi-hop knowledge graph to handle Question Answering. [17] proposed a knowledge graph question answering model based on end-to-end learning. [18] proposed a collaborative system which contains two agents. And one agent is used to reason path in knowledge graph, another is used to extract relation from background corpus. More recently, [4] proposed a method called Policy-Guided Path Reasoning which couples recommendation and interpretability by providing actual paths in a knowledge graph. [19] proposed a method which can identify explicit paths from users to items over the knowledge graph as the recommendation results, and experimental results show that not only the method gets a good recommendation results, but also provides explanations. [20] proposed a cooperative system including reasoning agent and information extraction agent to handle with Question Answering problem. The reasoning agent identifies the path over knowledge graph, and the information extraction agent provides shortcut or missing relations for long-distance target entity. [21] proposed a new performance metric for Question Answering agents which improve the results of Question Answering models while not to answer a limited number of questions which have been answered correctly. [22] proposed a new method named CogKR which includes summary module and reasoning module to handle with the one-shot knowledge graph reasoning problem. Besides, reinforcement learning can also be used in automated knowledge base completion and knowledge aware conversation generation. [23] proposed a new framework which reasons the relations between the missing factors and updates the knowledge base to implement the automated knowledge base completion. [24] proposed a new chatting machine which can generate conversation by reasoning over the augmented knowledge graph containing both triples and texts. We compare against such work in our experiments.

Compared with other reinforcement learning path reasoning methods, PGMD use an BiLSTM network with attention mechanism to extract path features which has an important underlying ordered time sequence, and use a new formula which computes the macro-averaged matching score between nodes on the path with the query nodes as the soft reward. Besides, the action trimming method also plays an import role in the algorithm.

C. DIALOGUE CONSTRUCTION

Building an automated conversational agent is a long cherished goal in Artificial Intelligence (AI). At present, there are two common ways to construct a dialogue bot: generation-based methods [25], [26] and retrieval-based methods [2], [3], [27]–[30]. In particular, [2], [27], [28] analyzed different baselines which aim to select the next response on the Ubuntu Dialogue Corpus. [3] formed a fine-grained context representation via formulating previous utterances into context to get a better performance. [29] proposed a model named SMN to address problem i.e., losing relationships among utterances or important contextual information. [30] used Deep Attention Matching Network to select response which takes advantage of attention mechanism to extract information from user utterance and response. Generation-based models generate the best answer under the context. With sufficient data, they can learn various ways to generate diverse responses. However, they may not be sufficiently stable. Retrieval-based chatbots, on the other hand, select a suitable response from a pre-built inventory of potential responses. Their advantage is that the entire system is relatively stable as they only considers a specific narrow domain. However, the set of potential answers is limited by the repository.

Multi-turn retrieval-based dialogues usually compute the matching scores between user utterances and responses and then select the suitable responses from the response inventory. However, our method primarily identifies the standard queries for the multi-turn dialogues. Then the users can get responses according to the standard queries identified by our systems.

D. DIALOGUE INTENT MINING

Generally, the dialogue systems are usually classified into two categories including task-oriented dialogue systems and non-task-oriented dialogue systems. The task-oriented dialogue systems aim to handle certain questions and the non-task-oriented dialogue systems do not have certain targets. The first step of the pipeline for task-oriented dialogue systems is to capture users' intents according to the users' utterances, then the second step is to make actions based on the task policy, and finally the systems select a decent responses to reply to users from the pre-built inventory associated with the actions [31]. The methods using deep learning techniques have made great progress in dialogue intent mining [32]–[34], and convolutional neural networks (CNN) are used to capture the user utterance features to identify standard queries [35]. Moreover, [36] and [37] resembled CNN-based model to get a better performance.

There is no previous work using knowledge graphs to identify suitable query nodes with explainable paths for multi-turn

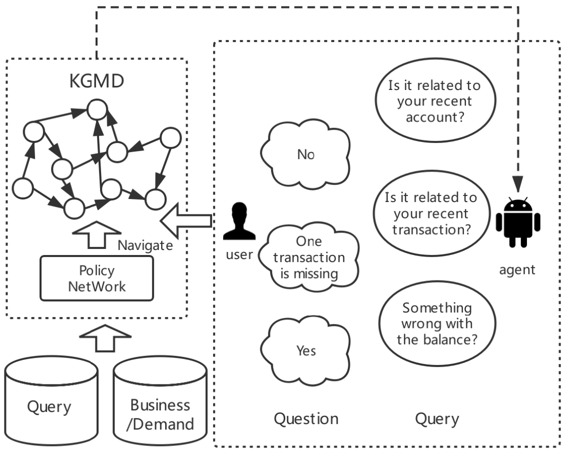

KGMD
Navigate
Policy NetWork
Query
Business /Demand
user
No
One transaction is missing
Yes
Question
Is it related to your recent account?
Is it related to your recent transaction?
Something wrong with the balance?
Query
agent

FIGURE 1. Multi-turn dialogue example.

dialogue. Our method can give clear explanations about how the multi-turn dialogues led to the query nodes.

III. PRELIMINARIES

In a task-oriented dialog system (cf. Figure 1), the system response in each turn of the dialogue is decided by the intention analyzed from the previous turns of user utterances, which plays a significant role in the whole dialogue system. And the goal of our system is exactly the intent mining in the certain task-oriented dialogue system via reinforcement knowledge graph reasoning.

A. INPUT

In our experiments, we consider a dataset coming from a company's real customer service hotline, with 19 predefined standard queries and about 120,000 call dialogues. As input, we consider multi-turn dialogue data including the automated customer service agent from the company that attempts to identify the human caller's intent with regard to an inventory of *standard queries*, i.e., the intents for which the customer service can provide predefined help. The user utterances in the dataset refer to user questions, the sub-intents for dialogues include the demands associated with the dialogues and the relevant business units for the dialogues, and the intents for dialogues refer to standard queries. An example of such a conversation is given in Table 1. The data comprises customer utterance, business unit and demand for each turn of customer questions or utterances, and the standard query IDs (QIDs) for the overall multi-turn dialogue. We assume the connection between business units and their demands is known.

B. PROBLEM FORMULATION

A knowledge graph (KG) $\mathcal{G}$ is a graph $\mathcal{G} = \{(e, r, e') \mid e, e' \in \varepsilon, r \in \mathbb{R}\}$ that captures factual information. A node e represents an entity, class, type, or literal. ε is the set of nodes, and $\mathbb{R}$ is the set of edges between pairs of entities e, e', where two nodes e, e' are connected by a predicate r,

TABLE 1. **Example of a 3-turn dialogue.**

	Customer Utterance	Business Unit	Demand	QID
1	Well, how to restrict the payment of credit cards?	Credit payment	Close	15 (Payment issues)
2	No	Credit payment	Payment method	
3	I want to set a restriction for my receiving code so others cannot pay me with credit cards.	Credit receipt	Open	

TABLE 2. **Data description (counted by turns).**

Entities	Description	Count
Question	User question/utterance for every turn in a dialogue	190,273
Demand	Underlying demand for dialogue	205
Business	Business units for dialogue	41
Query	Standard query, i.e., intent for overall dialogue	19
Predicate	**Description**	**Count**
goesOn	$question \xrightarrow{goesOn} question$	190,273
hasDemand	$question \xrightarrow{hasDemand} Demand$	190,478
hasBusiness	$question \xrightarrow{hasBusiness} Business$	190,314
includes	$Business \xrightarrow{includes} Demand$	246
demandHas	$Demand \xrightarrow{demandHas} query$	224
businessHas	$Business \xrightarrow{businessHas} query$	60
isQuery	$question \xrightarrow{isQuery} query$	190,292

forming a semantic fact (subject e, predicate r, object e'), e.g., (*Berkeley, locatedIn, California*).

KGs are widely used to model such semantics relationships. In this paper, we model the multi-turn dialogue process as an ad hoc knowledge graph $\mathcal{G}_D$, created on the fly to capture relationships between nodes including multi-turn dialogue text utterances T of customers and a subset of standard queries Q, where $T, Q \subseteq \varepsilon$ and $T \cap Q = \emptyset$. The two entities are connected via predicates $r_{t,q}$, where $t \in T, q \in Q$. Overall, in our dataset, there are 4 kinds of entities and 7 types of predicates, as described in Table 2. Given a knowledge graph $\mathcal{G}_D$, the maximum length of searchable paths K and the number of standard queries N, the goal is to learn a model and identify a candidate set $\{(q_n, p_n) \mid 0 \leq n < N\}$ for each customer question $t \in T$, where p_n denotes the probability of query q_n. Thus, for every pair (t, q_n), we need to have a path $p_k(t, q_n)$ with $2 \leq k \leq K$.

C. DEFINITIONS

Definition 1 (k-Hop Path): A k-hop path from entity e_0 to entity e_k is a sequence of $k+1$ entities with k intermediate relationships. This is expressed as $p_k(e_0, e_k) = \{e_0 \overset{r_1}{\longleftrightarrow} e_1 \overset{r_2}{\longleftrightarrow} \ldots \overset{r_k}{\longleftrightarrow} e_k\}$, among them $e_{i-1} \overset{r_i}{\longleftrightarrow} e_i$ indicated as $(e_{i-1}, r_i, e_i) \in \mathcal{G}_D$ or $(e_i, r_i, e_{i-1}) \in \mathcal{G}_D$, $i \in [k]$.

Definition 2 (1-Hop Scoring Function): We define a scoring function f to compute the degree to which the entity e matches the entity e_k, where the relationship r is a relationship along the k-hop path of the entity e to the entity e_k.

$$f(e, e_k) = \langle e + r, e_k \rangle + b_{e_k} \tag{1}$$

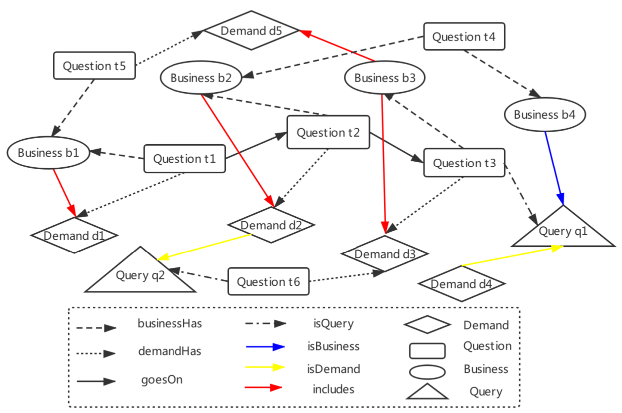


FIGURE 2. **An example of a multi-turn dialogue knowledge graph.**

IV. METHODOLOGY

We construct a knowledge graph for the multi-turn dialogue based on the entities and relations shown in Table 2. All user questions (or utterances), standard queries (which serve as the intents of dialogues), sub-intents of dialogues which include business units and demands can become entity nodes in such a graph. The goal is to find the correct query for the overall dialogue. This entails devising a strategy to pursue paths emanating from the user question node and leading to an appropriate query node. The searching path which begins from the first turn, passes through the following turns and finally reaches the query can be modeled as a Markov Decision Process. Hence, we rely on reinforcement learning for navigation along the graph towards the correct query node.

A. DIALOGUE GRAPH CONSTRUCTION

From the multi-turn dialogue, we induce a knowledge graph (cf. Figure 2) with edges connecting 4 types of entities. For every customer utterance node, there are multiple paths in the graph that can reach the standard query nodes. We assess which standard query has the highest probability based on scores within the knowledge graph.

First, in order to construct a knowledge graph, we begin by extracting triples of the form (e, r, e') from the data and add them to the knowledge graph. For instance, according to the Table 2, "Credit Pay" is a business unit entity (business b2 in Figure 2), while "stolen" is a Demand entity (Demand d2 in Figure 2), and the predicate "includes" is the relationship for the edge linking these two nodes.

1) VECTOR REPRESENTATIONS

While it is possible to apply structured queries on a knowledge graph, to make full use of the rich information that it provides, we additionally learn vector representations. We exploit the elegant TransE method [38] to learn such representations. For the embedding layer, the embeddings for query nodes, business nodes and demand nodes are initialized randomly based on a certain distribution, and then updated in the process of training TransE.

2) BERT QUESTION TEXT EMBEDDINGS

$$E_{\text{demand}}, E_{\text{business}}, E_{\text{query}}, E_{\text{question}} = TransE(E_{\text{demand}}, E_{\text{business}}, E_{\text{query}}, E_{\text{question}}) \quad (2)$$

with embedding matrix initialization

$$\begin{aligned} E_{\text{demand}} &= Embedding_{\text{random}}(V_{\text{demand}}) \\ E_{\text{business}} &= Embedding_{\text{random}}(V_{\text{business}}) \\ E_{\text{query}} &= Embedding_{\text{random}}(V_{\text{query}}) \\ E_{\text{question}} &= Embedding_{\text{BERT}}(V_{\text{question}}) \quad (3) \end{aligned}$$

However, considering only structural information, the embeddings for *question* entities would remain overly coarse-grained and uninformative in light of the rich semantic structure of linguistic utterances. Hence, in order to learn richer embeddings capturing fine-grained semantic nuances, we invoke the pretrained BERT model [5] for embedding initialization, and fine-tune it in the process of training the TransE model. Overall, the process of training TransE can be summarized as Equation 2 and Equation 3.

B. MULTI-TURN DIALOGUE PATH SEARCHING ALGORITHMS

In order to identify the user intent by mapping it to a standard query, we need to find the correct path in the knowledge graph from the user question node to such a query node. We design multi-turn dialogue path searching algorithms including backward tracking strategy and forward tracking strategy. We formalize the forward tracking strategy as Algorithm 1 and propose the algorithmic procedure using forward tracking strategy formalized as Algorithm 2 by considering the specific multi-turn property of the data.

While searching for paths, we rely on reinforcement learning to select the next node in the knowledge graph (cf. blue lines). For a multi-turn dialogue, the shorter the search path, the more reliable its result tends to be. Hence, we define a threshold σ as the maximum number of search steps. When using forward tracking strategy, we start searching the path from the utterance in the first turn of the multi-turn dialogue. For a three-turn dialogue $\{\text{question}_1 \rightarrow \text{question}_2 \rightarrow \text{question}_3\}$, for instance, we set the node question_1 as the starting point when searching for a query node. However, it is possible that the query node is not reached when limiting the number of steps. If the path searching process stops at a certain step, the multi-turn dialogue will fail to return a query. In this case, we forward track to search from question_2.

Inversely, when using backward tracking strategy, we start searching the path from the utterance in the last turn of the multi-turn dialogue. If the query node is not reached, we backward track to search from the previous turn of the multi-turn dialogue.

For this path selection part, the time complexity for each dialogue is $O(T * L * A)$, the T represents the maximum turns of the dialogue, the L represents the maximum length of path when system search query nodes, and the A represents maximum out-degree in the knowledge graph.

Algorithm 1 Multi-Turn Forward Tracking Path Searching

```
Input: The dialogue questions list, $T$;
Input: The query set, $Q$;
Input: The maximum searching steps, $\sigma$;
 1: Initialize $l = length(T)$, $k = 0$
 2: while $k < l$ do
 3:     $current\_node = T[k]$
 4:     $s = 1$                                          ▷ start with step 1
 5:     while $s \leq \sigma$ do
 6:         $current\_node = search\_next\_node(current\_node)$     ▷ find next node in KG
 7:         $s{+}{+}$                                    ▷ next step
 8:         if $current\_node \in Q$ then
 9:             return $current\_node$                   ▷ find the target node
10:         end if
11:     end while
12:     $k{+}{+}$                                        ▷ need forward tracking
13: end while
14: return $\emptyset$
```

C. REINFORCEMENT LEARNING

The goal of our reinforcement learning is to pursue suitable paths in the knowledge graph. Algorithm 2 provides the details of the proposed reinforcement learning empowered PGMD algorithm, which extends the path searching process.

1) POLICY/VALUE NETWORK

At every step, the reinforcement learning model requires the state of the current search path to select the best action to take. An important property for multi-turn dialogue is that the different turns adhere to an underlying ordered time sequence. For instance, the query nodes for two partial dialogues $question1 \rightarrow question2$ versus $question2 \rightarrow question1$ may be entirely different. Thus, the network needs to account for this temporal order property of the data. We use an Actor-Critic algorithm, with the structure of the network as given in Figure 3. An BiLSTM network with attention mechanism is invoked to extract path features. We further concatenate the embedding of the historical nodes with the BiLSTM model's output as a fusion layer, and then pass it through two fully-connected layers. Finally, the probabilities of actions in the action space are emitted by the actor layer. The effect of the network is evaluated by the critic layer.

Algorithm 2 Policy Guided Multi-Turn Path Reasoning (PGMD)

```
Input: The dialogue questions list, $T$;
Input: The query set, $Q$;
Input: The maximum searching steps, $\sigma$;
Output: Reward $r$, path node set $P$
1:  Initialize $l = length(T)$, $k = 0$
2:  while $k < l$ do
3:      $P = \phi$                              ▷ path node set
4:      $i = 0$
5:      $current\_node = T[k]$
6:      while $i \leq k$ do
7:          $P \leftarrow T[i]$                  ▷ save $T[i]$ to $P$
8:      end while
9:      $s = 1$                                  ▷ start with step 1
10:     while $s \leq \sigma$ do
11:         $s = State(P)$                       ▷ get state from path
12:         $A = kg(current\_node)$              ▷ get action space from KG
13:         $\widetilde{A} = pruning(A \mid f)$  ▷ action pruning
14:         $next\_node = PGMD(s, \widetilde{A})$  ▷ determine next action
15:         $current\_node = next\_node$
16:         $P \leftarrow current\_node$         ▷ save $current\_node$ to $P$
17:         $s{+}{+}$                            ▷ next step
18:         if $current\_node \in Q$ then
19:             $r = Reward(current\_node \mid f)$  ▷ calculate reward
20:             return $r, P$
21:         end if
22:     end while
23:     $k{+}{+}$                                ▷ need forward tracking
24: end while
25: return $\emptyset$
```

2) STATES

The state is the input of the policy network and provides information about the current path. To avoid overfitting, we only consider partial paths. Define k as the upper-bound of the historical nodes used to make decision. State s_t at step t is the start node embedding E_q and the path embedding starts from the current node e_t to the previous k nodes e_{t-k+1} including edges: $(E_{e_{t-k+1}}, E_{r_{t-k+2}} \ldots, E_{r_t}, E_{e_t})$ (E_e is the embedding of the entity e, and E_r is the embedding of the predicate r).

3) ACTIONS

For a current node e_t at step t, the complete action space includes all the outgoing connected nodes of e_t (but excluding historical nodes). Some nodes in the graph may have a large out-degree. Owing to efficiency considerations, we propose an action pruning strategy. We compute scores of node e_t with all nodes in the complete action space A according to the 1-hop scoring function $f(e_t, a)$, $a \in A$. Given δ as the upper-bound of the size of the action space, we eliminate low-scoring actions after sorting. The pruned action space $\widetilde{A}$ is defined in Equation 4.

$$\widetilde{A} = \{(e_t, a) \mid |(e_t, a)| \leq \delta, a \in A\} \tag{4}$$

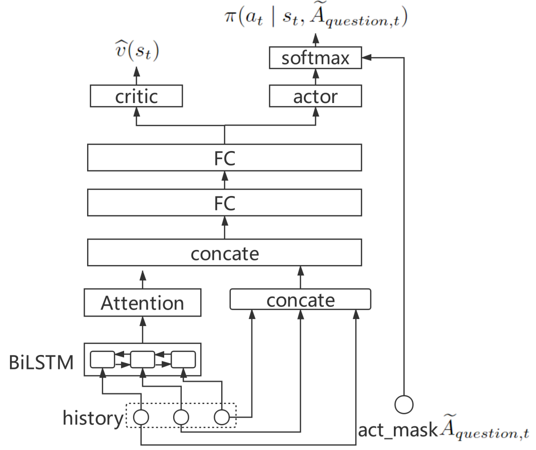

$\pi(a_t \mid s_t, \widetilde{A}_{question,t})$
$\widehat{v}(s_t)$
softmax
critic
actor
FC
FC
concate
Attention
concate
BiLSTM
history
act_mask$\widetilde{A}_{question,t}$

FIGURE 3. The policy network.

4) REWARD

During path searching in the KG, it is not possible to confirm whether the action will ultimately reach the correct target before the final step. Hence, we cannot only use a binary reward to indicate whether the agent has reached the target. Instead we propose a soft reward formula when the agent reach query nodes except the target. As the number of nodes of each type on the path may vary, and we wish for each type of node to play the same role, we consider as the reward the macro-averaged matching score between nodes on the path and the query node. The reward function is defined as Equation 5.

$$r = \begin{cases} \frac{1}{3}[\frac{1}{n_0}\sum_{i=1}^{n_0}\max(0, \frac{f(e_0^i, e_t)}{\max_{q \in Q} f(e_0^i, q)}) \\ \quad +\frac{1}{n_1}\sum_{j=1}^{n_1}\max(0, \frac{f(e_1^j, e_t)}{\max_{q \in Q} f(e_1^j, q)}) \\ \quad +\frac{1}{n_2}\sum_{k=1}^{n_2}\max(0, \frac{f(e_2^k, e_t)}{\max_{q \in Q} f(e_2^k, q)})] & \text{if } e_t \in Q \\ & \text{and } e_t \neq e_r \\ 1 & \text{if } e_t = e_r \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where Q is a set of query entities, B is a set of business entities, D is a set of demand entities, and T is a set of question entities. e_0, e_1 and e_2 represent nodes of searching path, and $e_0 \in D$, $e_1 \in B$, $e_2 \in T$. e_r is the query node corresponding to the multi-turn dialog. n_0 is the number of demand nodes of the path, n_1 is the number of business nodes of the path, n_2 is the number of question nodes of the path.

TABLE 3. Data sources used by baselines.

Model	Question	Business	Demand	Optimizer
BERT-classification	T	T	T	Adam
BPR	T	-	-	SGD
DeepCoNN	T	-	-	RmsProp
KGCN	T	T	T	SGD
KTUP	T	-	-	Adam
JRL	T	T	T	SGD
semhash-classification	T	T	T	Adam
DeepPath	T	T	T	Adam
MultiHopKG	T	T	T	Adam
MINERVA	T	T	T	Adam
PGPR	T	T	T	Adam
PGMD(ours)	T	T	T	SGD

V. EXPERIMENTAL RESULTS

A. SETTINGS

1) DATASET

The details of the dataset were already given in Section III. From the total of $120,000$ call dialogues, we randomly selected one-tenth as the test set, one-tenth as the valid set and the remaining eight tenths for training.

2) DATA PROTECTION STATEMENT

1) The data used in this research does not involve any Personal Identifiable Information (PII).
2) The data used in this research were all processed by data abstraction and data encryption, and the researchers were unable to restore the original data.
3) Sufficient data protection was carried out during the process of experiments to prevent the data leakage and the data was destroyed after the experiments were finished.
4) The data is only used for academic research and sampled from the original data, therefore it does not represent any real business situation in Ant Financial Services Group.

3) EVALUATION METRICS

The experiments target at evaluating whether our algorithm can predict the correct query for a user-provided questions within the dialogue. To this end, we compute macro-Precision (Prec.), macro-Recall (Rec.) and macro-F1 to evaluate the performance of the top-1 result. There are also scenarios in the company requiring multiple queries to be selected. Thus, we additionally computed $Precision@K$, which counts a result as correct when among the top-ranked K result queries, there is at least one match with the ground-truth query.

4) IMPLEMENTATION DETAILS

In our experiments, we relied on a maximum searching step limit $\sigma = 3$, an upper-bound of historical state nodes $k = 3$, and an upper-bound $\delta = 100$ for the action space. We set the dimensionality of the word embeddings to 100. To increase the diversity of paths, we set the dropout rate to 0.5, and use SGD^2 optimizer. We train the model for 10 epochs, setting the learning rate to 0.0001, and adopting a batch size of 64, with the entropy loss weight set to 0.001. Especially, the SGD^2 refers to the SGD optimizer with 0.9 momentum. We have presented the performance when we set different parameters which are the most important parameters including optimizer, maximum searching step limit and upper-bound for action space in the paper Section V-C, Section V-G and Section V-F, and select the parameters corresponding to the best result as the configuration. As for other parameters such as dimensionality of word embedding [1], dropout rate [17], learning rate [4], batch size [4], upper-bound of historical state nodes [4], entropy loss weight [17], are set according to the previous work.

B. BASELINES

We compare the proposed PGMD against both recommender systems and text classification methods. As the baselines are not specifically designed for our problem, they rely on varying subsets of data sources. Details of the data sources used by every baseline could check Table 3.

BERT-Classification: We use the pre-trained BERT vectors of the questions for the task of intent classification.

BPR [39]: The Bayesian Personalized Ranking approach for recommendation, which is one of the state-of-the art ranking-based method for top-N recommendation with numerical ratings. and we use BPR-MF for model learning.

DeepCoNN [40]: The Deep Cooperative Neural Networks model for recommendation, which models users and items jointly using review text for rating prediction.

KGCN [10]: The Knowledge Graph Convolutional Network for recommendation, which mines associated attributes between items on knowledge graph.

KTUP [41]: A Joint Knowledge Graph Recommender, which transfers the relation information in knowledge graph in order to figure out the reason why an user prefers an item.

JRL [42]: A Joint Representation Learning(JRL) framework based on multi-view machine learning, which is capable of incorporating heterogeneous information sources for top-N recommendation by learning user/item representations in a unified space.

Semhash-Classification [43]: We use Semantic Hashing vectors of the questions for the task of intent classification.

DeepPath [1]: A method for knowledge graph reasoning, which includes a reward function that takes the accuracy, diversity and efficiency into consideration.

MINERVA [16]: An reinforcement learning method for knowledge reasoning, which navigates the agent based on the input query to identify predictive paths in the graph.

MultiHopKG [17]: An approach to reason in knowledge graph, which reduces the influence of false negative supervision and weakens the sensitivity to spurious paths of on-policy RL.

PGPR: We adapt PGPR to our problem by removing the BiLSTM and attention mechanism in the policy/value net-

TABLE 4. Statistics of epoch 1, 5 and 10 for backward (Right) and forward (Left) tracking path searching.

			Forward							Backward						
Epoch	Turn	Total	Que1	Que2	Que3	Que4	Que5	Invalid	F1(%)	Que1	Que2	Que3	Que4	Que5	Invalid	F1(%)
1 Epoch	1	6,348	1,324	-	-	-	-	5,024	38.006	1,300	-	-	-	-	5,048	34.991
	2	35,392	7,273	6,548	-	-	-	21,571		5,591	8,252	-	-	-	21,549	
	3	42,175	8,843	6,822	6,473	-	-	20,037		5,283	6,669	10,074	-	-	20,149	
	4	17,764	3,679	3,098	2,330	2,040	-	6,617		1,814	2,295	2,942	4,100	-	6,613	
	5	5,554	1,213	953	795	571	440	1,582		429	569	805	876	1,310	1,565	
	SUM	100,885	22,332	17,421	9,598	2,611	440	54,831		14,417	17,785	13,821	4,976	1,310	54,924	
5 Epoch	1	6,348	1,654	-	-	-	-	4,694	37.844	1,556	-	-	-	-	4,792	34.858
	2	35,392	9,039	8,381	-	-	-	17,972		6,073	11,003	-	-	-	18,316	
	3	42,175	10,829	7,991	7,590	-	-	15,765		5,451	7,359	13,263	-	-	16,102	
	4	17,764	4,687	3,434	2,613	2,174	-	4,861		1,739	2,465	3,192	5,415	-	4,953	
	5	5,554	1,541	1,089	895	530	441	1,058		411	575	761	1,012	1,666	1,129	
	SUM	100,885	27,750	20,895	11,098	2,704	441	44,350		15,230	21,400	17,216	6,427	1,666	45,292	
10 Epoch	1	6,348	1,742	-	-	-	-	4,606	36.493	1,793	-	-	-	-	4,555	34.178
	2	35,392	9,557	9,271	-	-	-	16,564		6,097	12,464	-	-	-	16,831	
	3	42,175	11,685	8,453	8,010	-	-	14,027		5,313	7,470	15,030	-	-	14,362	
	4	17,764	4,994	3,746	2,507	2,263	-	4,254		1,700	2,377	3,388	5,946	-	4,353	
	5	5,554	1,565	1,196	872	541	504	876		359	539	794	1,038	1,881	943	
	SUM	100,885	29,543	22,666	11,389	2,804	504	40,327		15,262	22,850	19,212	6,984	1,881	41,044	

TABLE 5. Evaluation results of different optimizers during training.

optmizer.	Prec.(%)	Rec.(%)	F1(%)	Prec.@2(%)	Prec.@3(%)	time(min)
$Adam$	37.919	38.601	37.836	52.275	59.019	101
SGD^1	38.069	38.573	37.820	50.868	57.211	205
SGD^2	**38.234**	**38.534**	**38.006**	**52.314**	**59.791**	191
$RMSProp$	37.417	38.429	37.479	52.478	59.938	97
$AdaGrad$	37.741	37.769	37.372	51.389	59.679	164

work (cf. Figure 3), but only keeping the concatenation layer of historical nodes.

C. OPTIMIZER

We analyze the performance when using different optimizer during training. Five optimizers are compared in the Table 5 including *Adam*, *SGD*^{1}, *SGD*^{2}, *RMSProp* and *AdaGrad*. Particularly, the *SGD*^{1} refers to the *SGD* optimizer with no momentum, while the *SGD*^{2} represents the *SGD* optimizer with 0.9 momentum. The alpha parameter for *RMSProp* optimizer is set as 0.9. The *time*(*min*) refers to the average cost time for every epoch in the training procedure. By using the *SGD*^{2}, the system performs the best, but costs too much training time. One explanation is that the momentum increases the rate of convergence and helps the optimizer avoid the local optima value. When using *Adam*, the performance is not the best, but less time cost. Thus *Adam* is an optimizer with excellent comprehensive performance in our experiments. The application of adaptive learning rate allows the loss function to converge quickly. However, although *Adam* has an excellent convergence speed at the early stage of training, the final generalization ability of the model is not as good as the model trained with *SGD*.

D. QUANTITATIVE ANALYSIS

In order to compare PGMD against other baseline models exhaustively, we conduct an extensive quantitative analysis of

TABLE 6. Evaluation results of top-1, top-2 and top-3.

Model	Prec.(%)	Rec.(%)	F1(%)	Prec.@2(%)	Prec.@3(%)
BERT-classification	45.377	26.852	30.190	38.360	45.410
BPR	43.281	26.253	29.307	37.790	45.251
DeepCoNN	7.317	6.810	3.880	17.487	24.503
JRL	23.345	20.099	17.180	44.791	52.218
KGCN	6.127	7.621	6.719	14.21	22.219
KTUP	41.160	26.083	28.885	37.715	44.748
semhash-classification	48.533	25.410	28.759	37.044	44.413
DeepPath	20.972	20.350	18.610	28.266	31.614
MultiHopKG	30.637	31.242	30.642	47.774	56.987
MINERVA	23.249	32.138	23.760	25.676	27.027
PGPR	29.829	30.704	30.045	43.104	49.918
PGMD (ours)	**38.234**	**38.534**	**38.006**	**52.314**	**59.791**

these models. First, we train the model for 10 epochs with the default settings mentioned above including *SGD*^{2} optimizer, and observe the results of different models' top-1, top-2, top-3. The best values for every model are reported in Table 6. Overall, PGMD performs better than other baselines in precision, recall, and F1 of the results.

The results of DeepCoNN and KGCN are dismal. This might be because DeepCoNN relies on user reviews and item scores for training. KGCN, meanwhile, builds a knowledge graph around the item and the corresponding attributes. It relies on the user's previous interactions with the item to update the embedding matrix of users and achieve the effect of ''knowing'' a particular user. However, for our dialogue dataset, text utterances are often not repeated. Thus, the user information is limited to that given by the BERT vector representation of the dialogue text. As there is little duplicate text in the data, it is difficult to learn relationships between a text item and a target query node. Thus, every dialogue shows up as introducing a new user at test time, severely hampering the result quality.

TABLE 7. Evaluation results of different action space during training.

Action space	*Adam* F1(%)	*Adam* Prec.@2(%)	*Adam* Prec.@3(%)	SGD^2 F1(%)	SGD^2 Prec.@2(%)	SGD^2 Prec.@3(%)
100	**37.836**	**52.275**	**59.019**	**38.006**	**52.314**	**59.791**
200	35.065	49.888	57.598	35.508	50.190	57.607
300	35.105	50.147	57.779	35.217	49.288	57.094
400	34.254	49.456	57.546	34.806	49.672	57.080
500	33.370	49.102	56.406	33.243	48.006	56.061

TABLE 8. Evaluation results of different max. path search lengths.

Len.	*Adam* F1(%)	*Adam* Prec.@2(%)	*Adam* Prec.@3(%)	SGD^2 F1(%)	SGD^2 Prec.@2(%)	SGD^2 Prec.@3(%)
3	**37.836**	**52.275**	**59.019**	**38.006**	**52.314**	**59.791**
4	34.570	46.999	51.481	36.598	51.770	59.463
5	34.501	46.866	51.718	37.887	51.813	59.316

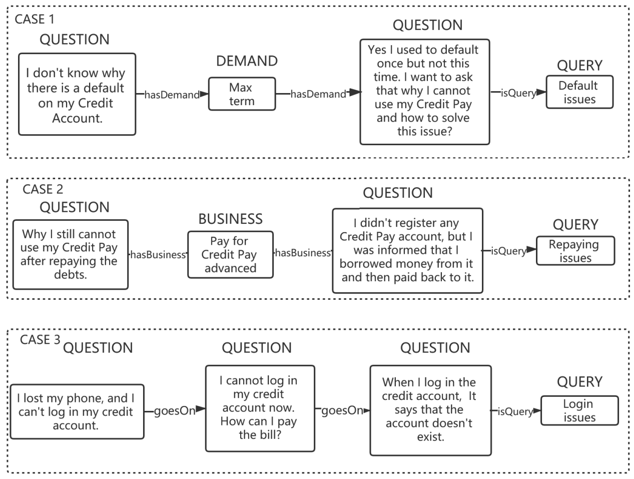

CASE 1
QUESTION
I don't know why there is a default on my Credit Account.
hasDemand
DEMAND
Max term
hasDemand
QUESTION
Yes I used to default once but not this time. I want to ask that why I cannot use my Credit Pay and how to solve this issue?
isQuery
QUERY
Default issues
CASE 2
QUESTION
Why I still cannot use my Credit Pay after repaying the debts.
hasBusiness
BUSINESS
Pay for Credit Pay advanced
hasBusiness
QUESTION
I didn't register any Credit Pay account, but I was informed that I borrowed money from it and then paid back to it.
isQuery
QUERY
Repaying issues
CASE 3
QUESTION
I lost my phone, and I can't log in my credit account.
goesOn
QUESTION
I cannot log in my credit account now. How can I pay the bill?
goesOn
QUESTION
When I log in the credit account, It says that the account doesn't exist.
isQuery
QUERY
Login issues

FIGURE 4. Examples.

E. EVALUATION OF PATH SEARCHING STRATEGIES

For further analysis, we compare our forward tracking strategy during path searching with a backward tracking one. We trained the model with SGD^2 optimizer and other default settings. The total numbers of multi-turn dialogues with respect to different turns of the dialogue is listed in Table 4. The table also provides statistics about which question nodes the algorithm could find query nodes for during the path search. Dialogue for which it is unable to find the query node are referred to as invalid dialogues when reasoning path. "Que" refers to question nodes in the tables. For example, the value 6, 822 in Table 4 signifies that there are 6, 822 3-turn dialogues for which the query node is found at the second turn among all the 42, 175 3-turn dialogues.

The experimental results show that by using the forward path searching strategy the number of searches from the previous question is more than the number of searches from the next question. In contrast, with the backtracking path search strategy, the number of searches from the next question is more than the number of searches from the previous one. The model using the backward tracking strategy attains a lower accuracy than that adopting the forward tracking strategy. One explanation is that for a multi-turn dialogue in our scenario, a new turn usually serve as additional information of the previous turn and doesn't contain complete information. When using forward tracking strategy, it is easier for the system to find the target query node based on the complete information. In contrast, when using backward tracking strategy, it is possible for the system to find wrong query node because of the incomplete information contained of the later turn.

F. INFLUENCE OF ACTION PRUNING STRATEGY

The action space has an important effect on the result. In this experiment, we evaluated the result of PGDM with different sizes of trimming action spaces. When the number of actions is lager than the upper-bound δ, the action space will be adjusted according to the scoring function. Higher score actions are more likely to be saved, the lower score actions will be removed from the action space. We want to explore whether a larger action space, even we keep all the actions, can help the system perform better than smaller action space. Because the *Adam* optimizer is the most widely used optimizer, we analyze the performance of system using different action space with *Adam* optimizer and SGD^2 optimizer. The size of the trimmed action space varies from 100 to 500, with a step size of 100. We can observe from Table 7 that when using a smaller pruning action space, it is more possible to get a better performance. The results indicate that it is effective to apply the action pruning strategy by 1-hop score function. Therefore when using smaller pruning action space, the system trim more noisy nodes, which increases the probability of reaching the correct query nodes.

G. HISTORY REPRESENTATIONS

The maximum length of the path when the system searching query node is an important hyper-parameters. We compared the use of different maximum path searching lengths, considering 3, 4, and 5 as the maximum lengths with *Adam* optimizer and SGD^2 optimizer. The above Table 8 provides the results for different maximum lengths. We find that with 3 as the maximum path length, the system performs best. One explanation is that though longer searching paths directed to more query nodes, they lead to less reliable predictions. Therefore when define the maximum path length, we need to consider the trade off between the reliability of the predictions and the searching range of query nodes.

VI. CASE STUDY ON PATH REASONING

In order to visually understand how our model allows for interpretability, we present a case study based on the previous experimental results. Figure 4 illustrates how to use the predicted path to explain the process of intent recognition through the paths. For the first example, the *question* value "I don't know why there is a default on my Credit Account." has the *Demand* value "Max term", while the *question* value "Yes I used to default once but not this time. I want to ask that why I cannot use my credit pay and how to solve this issue?" also has the *Demand* value "Max term", and the query of the *question* value "Yes I used to default once but not this time. I want to ask that why I cannot use my Credit Pay and how to solve this issue?" is query "Default issues". Thus, we can infer that the query for the *question* "I don't know why there is a default on my credit account." is also query "Default issues".

For the second example, the *question* value "Why I still cannot use my Credit Pay after repaying the debts." has the *Business* value "Pay for Credit Pay advanced", and the *question* value "I didn't register any Credit Pay account, but I was informed that I borrowed money from it and then paid back to it." also has the *Business* value "Pay for Credit Pay advanced", and the query of the *question* value "I didn't register any Credit Pay account, but I was informed that I borrowed money from it and then paid back to it." is query "Repaying issues", then we can think the query of the *question* value "Why I still cannot use my Credit Pay after repaying the debts" is also query "Repaying issues".

For the third example, the *question* value "I lost my phone, and I can't log in my credit account." can go on the dialog *question* value "I cannot log in my credit account now. How can I pay the bill?", while the dialog *question* value "I cannot log in my credit account now. How can I pay the bill?" can go on the dialog *question* value "When I log in the credit account, It says that the account doesn't exist.", and the query of the dialog *question* value "When I log in the credit account, It says that the account doesn't exist." is query "Login issues", then we can think that the query of the *question* value "I lost my phone, and I can't log in my credit account." is also query "Login issues".

VII. CONCLUSION

In this paper, we present a novel explainable approach for intent identification in multi-turn dialogue. Our novel PGMD approach relies on a reinforcement learning neural network to navigate an query-specific ad hoc knowledge graph in pursuit of relevant query nodes, via our order-aware forward tracking path searching algorithm for multi-turn dialogue. We conducted a series of experiments demonstrating that PGMD is a powerful method for multi-turn dialogue intent identification providing intuitive explanations and outperform state-of-the-art related work. We believe our method can be extended to dynamic knowledge graphs to deal with dynamic problems that new knowledge will appear in the future. As new nodes and edges being added in the knowledge graph, some existing nodes and edges are probably removed from the knowledge graph. In order to get the embedding of nodes and edges efficiently once updating, we could use DKGE model [44] which can achieve online embedding learning to get the updated knowledge graph embedding.

REFERENCES

[1] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1–10.

[2] R. Kadlec, M. Schmid, and J. Kleindienst, "Improved deep learning baselines for Ubuntu corpus dialogs," *CoRR*, vol. abs/1510.03753, 2015. [Online]. Available: http://arxiv.org/abs/1510.03753

[3] Z. Zhang, J. Li, P. Zhu, H. Zhao, and G. Liu, "Modeling multi-turn conversation with deep utterance aggregation," 2018, *arXiv:1806.09102*. [Online]. Available: https://arxiv.org/abs/1806.09102

[4] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Paris, France, Jul. 2019, pp. 285–294.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (NAACL-HLT)*, Minneapolis, MN, USA, vol. 1, Jun. 2019, pp. 4171–4186.

[6] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 1–22, May 2012.

[7] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2017, pp. 635–644, doi: 10.1145/3097983.3098063.

[8] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2018, pp. 5329–5336.

[9] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *Proc. World Wide Web Conf. (WWW)*. New York, NY, USA: ACM, 2019, pp. 151–161, doi: 10.1145/3308558.3313705.

[10] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *Proc. World Wide Web Conf. (WWW)*. New York, NY, USA: ACM, 2019, pp. 3307–3313, doi: 10.1145/3308558.3313417.

[11] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio, "SPrank: Semantic path-based ranking for Top- n recommendations using linked open data," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 1, pp. 1–34, Oct. 2016.

[12] T. Wever and F. Frasincar, "A linked open data schema-driven approach for top-N recommendations," in *Proc. Symp. Appl. Comput. (SAC)*. New York, NY, USA: ACM, 2017, pp. 656–663.

[13] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi, "Top-N recommendations from implicit feedback leveraging linked open data," in *Proc. 7th ACM Conf. Recommender Syst. (RecSys)*. New York, NY, USA: ACM, 2013, pp. 85–92.

[14] B. van Rossum and F. Frasincar, "Augmenting LOD-based recommender systems using graph centrality measures," in *Proc. 19th Int. Conf. Web Eng. (ICWE)* (Lecture Notes in Computer Science), vol. 11496, M. Bakaev, F. Frasincar, and I.-Y. Ko, Eds. Springer, 2019, pp. 19–31, doi: 10.1007/978-3-030-19274-7_2.

[15] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Proc. AAAI*, 2018, pp. 6053–6060.

[16] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, and A. Mccallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," 2017, *arXiv:1711.05851*. [Online]. Available: https://arxiv.org/abs/1711.05851

[17] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1–12.

[18] C. Fu, T. Chen, M. Qu, W. Jin, and X. Ren, "Collaborative policy learning for open knowledge graph reasoning," 2019, *arXiv:1909.00230*. [Online]. Available: http://arxiv.org/abs/1909.00230

[19] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, and J. Tang, "Explainable knowledge graph-based recommendation via deep reinforcement learning," 2019, *arXiv:1906.09506*. [Online]. Available: http://arxiv.org/abs/1906.09506

[20] Y. Zhang, X. Cheng, H. Gao, and C. Zhai, "Cooperative reasoning on knowledge graph and corpus: A multi-agentReinforcement learning approach," 2019, *arXiv:1912.02206*. [Online]. Available: http://arxiv.org/abs/1912.02206

[21] F. Godin, A. Kumar, and A. Mittal, "Learning when not to answer: A ternary reward structure for reinforcement learning based question answering," 2019, *arXiv:1902.10236*. [Online]. Available: http://arxiv.org/abs/1902.10236

[22] Z. Du, C. Zhou, M. Ding, H. Yang, and J. Tang, "Cognitive knowledge graph reasoning for one-shot relational learning," 2019, *arXiv:1906.05489*. [Online]. Available: http://arxiv.org/abs/1906.05489

[23] A. Tortay, J. H. Lee, C. H. Lee, and S. W. Lee, "Automated knowledge base completion using collaborative filtering and deep reinforcement learning," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 3069–3074.

[24] Z. Liu, Z. Niu, H. Wu, and H. Wang, "Knowledge aware conversation generation with explainable reasoning on augmented graphs," in *Proc. Artif. Intell.*, 2019, pp. 1–10.

[25] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2015, pp. 1577–1586.

[26] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. AAAI*, vol. 16, 2016, p. 3776–3784.

[27] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," in *Proc. 16th Annu. Meeting Special Interest Group Discourse Dialogue*, 2015, p. 285.

[28] R. T. Lowe, N. Pow, I. V. Serban, L. Charlin, C. W. Liu, and J. Pineau, "Training end-to-end dialogue systems with the ubuntu dialogue corpus," *Dialogue Discourse*, vol. 8, no. 1, pp. 31–65, Jan. 2017.

[29] Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li, "Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, R. Barzilay and M.-Y. Kan, Eds. Association for Computational Linguistics, 2017, pp. 496–505, doi: 10.18653/v1/P17-1046.

[30] X. Zhou, L. Li, D. Dong, Y. Liu, Y. Chen, W. X. Zhao, D. Yu, and H. Wu, "Multi-turn response selection for chatbots with deep attention matching network," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1. Melbourne, VIC, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1118–1127.

[31] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *ACM SIGKDD Explorations Newslett.*, vol. 19, no. 2, pp. 25–35, Nov. 2017.

[32] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2012, pp. 210–215.

[33] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 5045–5048.

[34] D. Yann, G. Tur, D. Hakkani-Tur, and L. Heck, "Zero-shot learning and clustering for semantic utterance classification using deep learning," in *Proc. Int. Conf. Learn. Represent.*, 2014, p. 28.

[35] H. B. Hashemi, A. Asiaee, and R. Kraft, "Query intent detection using convolutional neural networks," in *Proc. Int. Conf. Web Search Data Mining, Workshop Query Understand.*, 2016, pp. 1–5.

[36] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for Web search using clickthrough data," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage. (CIKM)*. New York, NY, USA: ACM, 2013, pp. 2333–2338.

[37] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for Web search," in *Proc. 23rd Int. Conf. World Wide Web (WWW Companion)*. New York, NY, USA: ACM, 2014, pp. 373–374.

[38] C. Moon, P. Jones, and N. F. Samatova, "Learning entity type embeddings for knowledge graph completion," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, Austin, TX, USA, Jun. 2015, pp. 2181–2187.

[39] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell. (UAI)*, Montreal, QC, Canada, Jun. 2009, pp. 452–461.

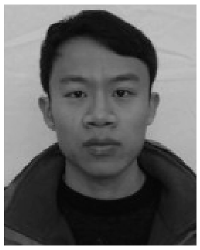
[40] W. X. Zhao, J. Wang, Y. He, J.-R. Wen, E. Y. Chang, and X. Li, "Mining product adopter information from online reviews for improving product recommendation," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 3, pp. 1–23, Feb. 2016.

[41] 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," *CoRR*, vol. abs/1902.06236, 2019. [Online]. Available: http://arxiv.org/abs/1902.06236

[42] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-N recommendation with heterogeneous information sources," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2017, pp. 1449–1458.

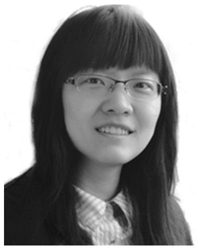
[43] K. Shridhar, A. Dash, A. Sahu, G. G. Pihlgren, P. Alonso, V. Pondenkandath, G. Kovacs, F. Simistira, and M. Liwicki, "Subword semantic hashing for intent classification on small datasets," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–6.

[44] T. Wu, A. Khan, H. Gao, and C. Li, "Efficiently embedding dynamic knowledge graphs," 2019, *arXiv:1910.06708*. [Online]. Available: http://arxiv.org/abs/1910.06708



KAI YANG has been pursuing the master's degree with the School of Software, Huazhong University of Science and Technology, since September 2018. From July 2019 to July 2020, he worked as an Intern with Ant Financial Services Group. His research interests include reinforcement learning and personalized recommendation systems.

XINYU KONG has been pursuing the master's degree with the School of Software, Beijing university of Posts and Telecommunications since September 2017. From May 2019 to July 2020, he worked as an Intern with Ant Financial Services Group. His research interests include reinforcement learning and natural language processing.



YAFANG WANG received the Ph.D. degree from Database and Information Systems Group, Max Planck Institute for Informatics, Germany, in February 2013. She worked as a Postdoctoral Researcher, until September 2013. From October 2013 to March 2017, she worked as an Associate Professor with the School of Computer Science and Technology, Shandong University. Since the end of March 2017, she has been working as a Staff Algorithm Engineer for Ant Financial (Alibaba Group). Her research interests include knowledge harvesting, semantic analytics, reinforcement learning, and dialogue systems.

JIE ZHANG received the master's degree from the Beijing University of Posts and Telecommunications, China, in 2016. She is currently a Senior Algorithm Engineer with Ant Financial Services Group. Her research interests focus on natural language processing, information processing, dialog systems, etc.

GERARD DE MELO received the Ph.D. degree from Saarland University, Saarbrücken, Germany, in 2010. From 2013 to 2016, he was an Assistant Professor heading the Web Mining and Language Technology Group, Tsinghua University. Before that, he spent two years as a Postdoctoral Research Scholar working with the AI/FrameNet Group, International Computer Science Institute (ICSI), UC Berkeley. Prior to that, he was a member of the Databases and Information Systems Group headed by Gerhard Weikum, Max Planck Institute for Informatics. He is currently an Assistant Professor with the Department of Computer Science, Rutgers University. He is also the Director of the Deep Data Laboratory.