Conversational Question Answering with Language Models Generated Reformulations over Knowledge Graph

Lihui Liu[*], Blaine Hill[*], Boxin Du[†], Fei Wang[†], Hanghang Tong[*]
University of Illinois Urbana-Champaign
Computer Science Department
Amazon
[*]{lihuil2, blaineh2, htong}@illinois.edu
[†]{boxin, feiww}@amazon.com

Abstract

Conversational question answering (ConvQA) over knowledge graphs (KGs) involves answering multi-turn natural language questions about information contained in a KG. State-of-the-art methods of ConvQA often struggle with inexplicit question-answer pairs. These inputs are easy for human beings to understand given a conversation history, but hard for a machine to interpret, which can degrade ConvQA performance. To address this issue, we propose a reinforcement learning (RL) based model, CORNNET, which utilizes question reformulations generated by large language models (LLMs) to improve ConvQA performance. CORNNET adopts a teacher-student architecture where a teacher model learns question representations using human writing reformulations, and a student model to mimic the teacher model's output via reformulations generated by LLMs. The learned question representation is then used by a RL model to locate the correct answer in a KG. Extensive experimental results show that CORNNET outperforms state-of-the-art ConvQA models.

1 Introduction

Knowledge graphs (KGs) are collections of nouns represented as nodes (representing real-world entities, events, and objects) and edges (denoting relationships between nodes). Knowledge graph question answering (KGQA) has long been a focus of study, with the goal of answering queries using information from a KG. However, traditional KGQA approaches often only consider single-shot questions (Liu et al., 2022), rather than the iterative nature of real-world conversation. Conversational question answering (ConvQA) addresses this gap by allowing users to interact with a QA system conversationally. ConvQA systems have had much success, as seen by Google's Lambda (Thoppilan et al., 2022), Amazon's Alexa and OpenAI's ChatGPT.

Conversational question answering (ConvQA) involves a multi-turn process consisting of users iteratively asking natural language questions, a system deciphering both the conversation context and underlying queries, and the system returning natural language answers. Some models will create rich, human-like responses (Acharya and Adhikari, 2021; Zhou et al., 2020; Brown et al.), these methods are known as 'dialogue' conversation models. While for ConvQA over KGs, a corresponding entity in the KG is sufficient to answer the input question, we call it 'non-dialogue' conversation models. In this paper, we focus on the non-dialogue ConvQA task as shown in Example 1.

Example 1:
q_1: Who is the author that wrote the book Moby-Dick?

Reformulation1: Author of the book?
Reformulation2: Who wrote Moby Dick?

a^1: Herman Melville
q_2: When was he born?

Reformulation1: His birthdate is?
Reformulation2: When was Herman Melville born?

a^2: 1 August 1819
q_3: And where is he from originally?

Reformulation1: His place of birth?
Reformulation2: Where did he grow up?

a^3: Manhattan
q_4:How about his wife?

Reformulation1: Where is Herman Melville's wife from?
Reformulation2: Herman Melville's wife's place of birth?

a^4: Boston
q_5: Did they make a movie based on the book?
a^5: yes

In general, a conversation is typically initiated with a well-formed question (i.e., q_1) followed by inexplicit follow-up questions (e.g., q_2 - q_5). The initial question (q_1) often includes a central **topic entity** of interest ("Moby-Dick"), while the topic entities of follow-up questions (q_2 - q_5) are not

Findings of the Association for Computational Linguistics ACL 2024, pages 839–850
August 11-16, 2024 ©2024 Association for Computational Linguistics

explicitly given. Additionally, the topic entity of the conversation may shift over time (e.g., inquiring about the birth time of Herman Melville in q_2).

To operate ConvQA over KG, different methods have previously been proposed. For instance, Magdalena et al. in (Kaiser et al., 2021) use named entity recognition (NER) methods to detect potential KG topic entities in the conversation and employ multi-agent reinforcement learning starting from these entities to find answers; the performance of this method is largely dependent on the quality of the detected entities. Philipp et al. in (Christmann et al., 2019) propose finding a conversation-related subgraph and using heuristic-based methods to identify the answer within the subgraph. The subgraph is expanded as new questions are asked. Endri et al. in (Kacupaj et al., 2022) use contrastive learning to separate correct answers from incorrect answers.

Despite the above progress, inexplicit input data hinders a ConvQA system's ability to find correct answers (Kaiser et al., 2021; Vakulenko et al.; Buck and Bulian, 2017). Two common linguistic phenomena which undermine the semantic completeness of a query in the conversation are: **anaphora** and **ellipsis** (Vakulenko et al.). **Anaphora** refers to the phenomenon of an expression that depends on an expression in the previous context. In Example 1, the word “he" in q_2 refers to a^1. Meanwhile, **ellipsis** refers to the phenomenon of the omission of expressions in the previous context. For example, the complete form q_4 should be "Where is Herman Melville's wife from?". To address this issue, several methods aim to learn a **reformulation** of the input query, *rewriting the original question* in a more meaningful way. Then, one can search for the answer using this new reformulation with existing techniques (Buck and Bulian, 2017; Vakulenko et al.; Nogueira and Cho, 2017). Although many of the existing question rewriting models have shown potential to enhance ConvQA performance, as demonstrated by prior research (Ishii et al., 2022), their generated reformulations fall short compared to human-generated reformulations (Vakulenko et al.).

In this paper, we present **CORNNET**, a new reinforcement learning (RL) model for non-dialogue conversational question answering (ConvQA) with large language model (LLM) generated reformulations. First, we fine-tune existing LLMs, GPT2 (Radford et al., 2018) and Bart (Lewis et al., 2019), to generate high quality reformulations, using human writing reformulations as the ground truth. Second, to further increase the convQA performance, we propose a teacher-student architecture to achieve near human-level performance.[1] Specifically, CORNNET (1) **directly trains** a teacher model with human writing reformulations in the training data, and (2) **indirectly trains** a student model with LLM-generated reformations to mimic the teacher model's output so that it can approach human-level performance. Note that the human writing reformulations only exist in the training and validation data. Lastly, to locate an answer, a RL model walks over the KG, sampling actions from a policy network to guide the direction of the walk and identify candidate answers. Our experiments demonstrate the effectiveness of CORNNET and its superiority over the state-of-the-art conversational question answering baselines.

The main contributions of this paper are: (1) **Analysis.** We demonstrate that although LLMs are good question reformulators, their performance lags behind human-level performance. (2) **Method.** We propose a RL based model CORNNET which utilizes the question reformulations to improve the QA performance. The proposed teacher-student model can help us achieve near human-level performance with LLM-generated reformulations. (3) **Evaluation.** The experimental results on several real-world datasets demonstrate that the proposed CORNNET consistently achieves state-of-the-art performance.

2 Related work

2.1 Conversational Question Answering

Various approaches have been used to develop ConvQA systems. For instance, in (Buck and Bulian, 2017), the authors employed RL to train an agent that reformulates input questions to aid the system's understanding. In (Guo et al., 2018), an encoder-decoder model is used to transform natural language questions into logical queries for finding answers. In (Kacupaj et al.), a Transformer model is used to generate logical forms and graph attention is introduced to identify entities in the query context. Other systems, such as Google's Lambda (Thoppilan et al., 2022), Amazon Alexa (Acharya and Adhikari, 2021), Apple's Siri, and

[1] Human-level performance refers to the ability to find answers based on real human writing reformulations during testing.

OpenAI's ChatGPT, are also pursuing this task.

2.2 Knowledge Graph Reasoning and Question Answering

Knowledge graph reasoning aims to infer or discovery new knowledge according to existing information in the knowledge graph. It has been used in many applications, such as knowledge graph completion (Wang et al., 2022), entity alignment (Yan et al., 2021a,b), temporal reasoning (Wang et al., 2024, 2023) and so on. While knowledge graph question answering, a special reasoning task, has been researched for some time, many of the existing methods primarily focus on answering single-turn questions (Liu et al., 2023; Bordes and N, 2013) or complex logical questions (Wang et al., 2021; Liu et al., 2021, 2024). For example, Zhang et al. (Zhang et al., 2022) use a KG as the environment and propose a RL-based agent model to navigate the KG in order to find answers to input questions. Similarly, in (Das et al., 2017; Lin and Socher; Xiong et al., 2017; Kingma and Ba, 2017), authors use RL models to find paths in the KG for answering input queries. Other studies, such as (Misu et al., 2012; Zhou et al., 2020; Acharya and Adhikari, 2021; Radford et al., 2018; Brown et al.), integrate RL with other methods to create more human-like systems.

2.3 Question Rewriting

Question rewriting aims to reformulate an input question into a more salient representation. This can improve the accuracy of search engine results or make a question more understandable for a natural language processing (NLP) system. In (Vakulenko et al.), an unidirectional Transformer decoder is proposed to automatically rewrite a user's input question to improve the performance of a conversational question answering system. In (Elgohary et al., 2019), authors propose a Seq2Seq model to rewrite the current question according to the conversational history, and also introduce a new dataset named CANARD. In (Fader et al., 2014), query rewriting rules are mined from a background KG and a query rewriting operator is used to generate a new question. Unlike the previous techniques, CORNNET trains teacher-student model with both human written reformulations and LLM-generated reformulations. This approach helps to avoid the negative impact from the generated low quality reformulations.

3 Problem Definition

A KG can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{L})$ where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes/entities, $\mathcal{R} = \{r_1, r_2, ..., r_m\}$ is the set of relations and $\mathcal{L}$ is the list of triples. Each triple in the KG can be denoted as (h, r, t) where $h \in \mathcal{V}$ is the head (i.e., subject) of the triple, $t \in \mathcal{V}$ is the tail (i.e., object) of the triple and $r \in \mathcal{R}$ is the edge (i.e., relation, predicate) of the triple which connects the head h to the tail t. The embedding of a node or relation type is represented by bold lowercase letters, e.g., $\mathbf{e}_i$, $\mathbf{r}_i$. Each triple/edge (h, r, t) in the KG has a unique edge embedding which is denoted as $\mathbf{u}_r$.

Conversational question answering over a KG aims to iteratively answer multiple related questions from the users. Unlike dialog question answering which wants the chatbot to imitate the response of a human, ConvQA over KG only requires the model to return entities in the knowledge graph. We formally define the key terminologies used in this paper as follows.

Conversation. A conversation C with T turns is made up of a sequence of questions $q_1, q_2, ..., q_T$ and their corresponding answers Ans = { $a^1, a^2, ..., a^T$ }, such that $C = \langle(q_1, a^1), (q_2, a^2), ..., (q_T, a^T)\rangle$. Example 1 in Introduction contains $T = 5$ turns. We assume that q_1 is well-formed, and all other q_t are inexplicit.

Question. Each question q_t is a sequence of words $q_t = (w_1^t, ..., w_{\Omega_t}^t)$, where Ω_t is the number of words in q_t. We assume each question can be mapped to a relation r_{q_t} in the KG and make no assumptions on the grammatical correctness of q_t.

Topic Entity. We assume that each q_t has a topic/central entity v_{q_t} which the user wants to ask about. We assume that the topic entity of q_1 is given in the training data, while the topic entities for other questions $q_2, ..., q_T$ are not given. For example, for the five questions in Example 1, their topic entities are `Moby Dick`, `Herman Melville`, `Herman Melville`, `Moby Dick` and `Moby Dick`, respectively. The topic entity of q_1 is presumed the main topic entity which is denoted as v_{q_1}.

Answer. Each answer a^t to question q_t is a (possibly multiple, single, or null-valued) set of entities in the KG. We assume that all the answer entities exist in the KG, except true or false questions.

Reformulation. A reformulation is a sentence which expresses the same information as the input question, but in a different way. We assume in the training data, each question has multiple refor-

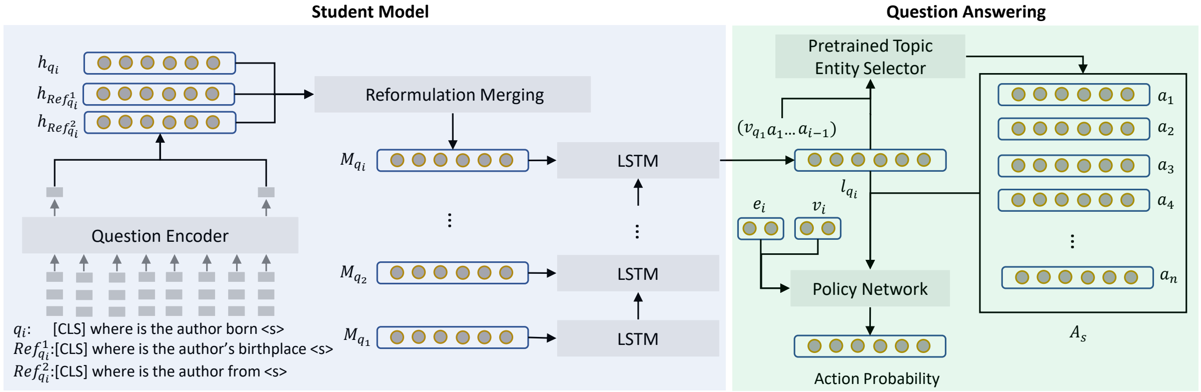

Figure 1: Training Process of CORNNET. The light gray part denotes the architecture of the student model. The light green part shows the framework of RL-based question answering model. Both are trained end-to-end.

mulations.

Turn. Each question in C, including its reformulations and corresponding answers, constitutes a turn t_i. Each turn t_i contains a question q_i, the answer a^i and reformulations of q_i.

Based on the above, we formally define the problem of ConvQA over KG as:

Definition 1. ***Given:*** *(1) A knowledge graph G, (2) the training set of conversations where each question contains multiple human written reformulations, (3) the test set of conversations where no question reformulation is provided;* ***Output:*** *(1) The trained model, (2) the answer for each question in each conversation of the test set.*

4 Proposed Method

Due to anaphora and ellipsis, current ConvQA methods often rewrite input queries to generate more understandable reformulations. In this paper, we follow this general idea by fine-tuning two existing LLMs, GPT2 and Bart, to generate reformulations. Although GPT2 and Bart are good reformulation generators, their performance still lags behind human-level performance. To further improve the performance, we propose a teacher-student architecture. The teacher model learns the question representation by using human written reformulations, while the student model takes reformulations generated by LLMs as input, and tries to mimic the output of the teacher model, so that it can achieve the same performance as the teacher model despite using the LLM-generated reformulations.

In each iteration, our model uses the conversation history and the current query to identify the current topic entity, and a RL agent travels the KG starting from the topic entity to find the answer. This process is repeated for a number of turns until the conversation is completed. Figure 1 illustrates the framework of the proposed CORNNET. We will describe the details of each component in the following subsections.

4.1 Student Model: LLMs Reformulation Encoder

A - Context Encoder. Given a question $q_i = (w^i_1, w^i_2, ..., w^i_{\Omega_t})$, we first add two indicator tokens ([CLS] and <s>) to the beginning and end of the question context to signify its boundary. Then, we pass the processed question context through a pre-trained BERT (Devlin et al., 2019) to extract contextual embeddings for each token:

$$[\mathbf{h}_{CLS}, \mathbf{w}_1, .., \mathbf{w}_{\Omega_t}, \mathbf{h_s}] = \text{BERT}([CLS], w_1, .., w_{\Omega_t}, < s >) \quad (1)$$

where $\mathbf{h}_{CLS}$ is the embedding of the [CLS] token and $\mathbf{h_s}$ is the embedding of the <s> token. The context question embedding is obtained from the transformation of $\mathbf{h}_{CLS}$ and $\mathbf{h_s}$, where FFN is a feed forward neural network.

$$\mathbf{h}_{q_i} = \text{FFN}(\mathbf{h}_{CLS} || \mathbf{h_s}) \quad (2)$$

B - Context Fusion. During the training, each input question has multiple corresponding reformulations generated. For each reformulation, we use the Context Encoder to obtain its context embedding. To merge the reformulation information, we stack the embeddings of the N reformulations and the original question context embedding to create a sequence with N+1 embeddings. We treat this sequence as the embedding of a language sentence and pass it through a Transformer Encoder (Vaswani et al., 2017) to merge them together.

$$\mathbf{M_{q_i}} = \text{TRANSFORMER}([\mathbf{h}_{q_i} | \mathbf{h}_{\text{Ref}^1_{q_i}} | ... | \mathbf{h}_{\text{Ref}^n_{q_i}}])[0]$$

where $\mathbf{M_{q_i}}$ is the query embedding after merging the reformulations and getting the first output.

C - Integrating Conversational History. Another problem in ConvQA is that the user's inputs are often ambiguous, hampering a system's ability to give accurate answers. This is illustrated in Example 1 q_3 "And where is he from originally?". It is impossible to identify the antecedent to the pronoun 'he' without any conversational history. Consequently, *conversational history is vital* to the success of CORNNET. We use an LSTM to encode all the conversational history which is given below.

$$\mathbf{l_{q_i}} = \text{LSTM}(\mathbf{M_{q_i}}) \tag{3}$$

the output of the LSTM $\mathbf{l_{q_i}}$ will be treated as the query embedding and be used by other components. Note that the reformulations used here are generated by LLMs.

4.2 Teacher Model: Human Written Reformulation Encoder

Reformulations have been used by various methods to improve the performance of QA systems by creating more understandable queries. For instance, in (Buck and Bulian, 2017), Christian et al. use a Seq2Seq-based reinforcement learning agent to transform input questions into machine-readable reformulations. In (Vakulenko et al.), Svitlana et al. propose a Transformer Decoder-based model for question rewriting. According to the study in (Ishii et al., 2022), most question reformulation methods only improve the performance about 2-3%. Despite that LLMs have exploded in popularity for all sorts of natural language tasks, the ConvQA performance based on LLM-generated reformulations is still upper bounded by human reformulations (Vakulenko et al.; Ishii et al., 2022).

To further improve the student model's performance, we propose a *teacher-student approach* where a teacher network is trained on human written reformulations. The teacher network has the same network structure as the student model, but uses human written reformulations as the input. An example is given in Figure 2. During the training process, our goal is to make the output question embedding of the student model as close as possible to the output of the teacher model in the embedding space. The distance between the student's and teacher's output is measured using the L2 distance:

$$L = \sum_{q_i \in C} [d(\Upsilon_{q_i}, \mathbf{l}_{q_i})], \tag{4}$$

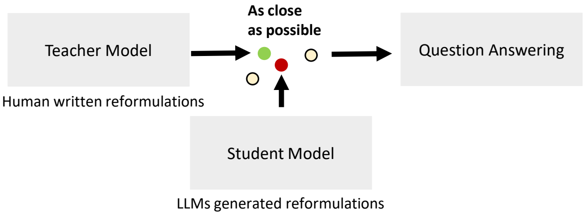

Figure 2: Reformulation imitator.

where Υ_{q_i} is the output of the teacher network for input question q_i, and $\mathbf{l_{q_i}}$ is the output of the student network for the same input with reformulations. By minimizing this distance, we can ensure that the student network produces the output that is similar to that of the teacher model, even when it only has access to the synthetic reformulations. Note that the teacher model is pretrained and fixed when we train the student model. During the testing phase, given a question, we first use LLMs to generate multiple reformulations for it, then the student model is used to encode the input question with LLM-generated reformulations. The performance of directly using the teacher model on the test data is slightly inferior to our model due to the different data distribution of human written reformulations compared to the reformulations generated with LLMs.

4.3 Inferring the Topic Entity

During a conversation, the topic entity may change over time. To accurately answer questions, we determine the current topic entity based on the conversation history and the current question. We use a multi-layer perception (MLP) to determine whether the topic entity remains unchanged. If the classifier predicts that the topic entity of the current question is not the answer to the previous question, we set the topic entity to the main topic entity v_{q_1}.

The classifier consists of a feed forward neural network (FFN) with ReLU activation functions and a classification layer. The classification layer uses softmax on a 2D output to calculate the cross-entropy loss. Here we use the index format of PyTorch to show that the probability of $v_{q_i} = a^{i-1}$ is equal to the second element of the 2D output.

$$\Pr(v_{q_i} = a^{i-1}) = \text{MLP\_classifier}(\text{FNN}(\mathbf{l}_{q_i}))[1]$$

A - Pretrain Topic Entity Selector. The goal of the Topic Entity Selector is to identify the correct topic entity within the conversation, which is an input to the RL model. In order to stabilize the training process for the RL model, we pre-train the parameters of the classifier using binary cross-entropy loss

$$\mathcal{L}_1 = -\left[y \log(\Pr(v_{q_i} = a^{i-1})) + (1-y)\log(1 - \Pr(v_{q_i} = a^{i-1}))\right)]$$

4.4 Question Answering

After obtaining the topic entity, the next step is to find the correct entity to answer the user. We formulate this problem as a Markov decision process (MDP) which is defined by a 5-tuple (S, A, R, P, γ), where S is the state space, A is the action space, P is the state transition function and R denotes the reward function.

States. Intuitively, we want a state to encode the question, the current position of the agent in the KG, and the search history information. At the ith step, the state $s_i \in S$ is defined as a triple $s_t = (n_i, \mathbf{l}_q, \mathbf{g}_i)$, n_i is the current entity where the agent is at; $\mathbf{l}_q$ is the question embedding generated by the student network; and $\mathbf{g}_i$ refers to the search history information. $(n_i, \mathbf{g}_i)$ can be viewed as state-dependent information while $(\mathbf{l}_q)$ is the global context shared by all states.

Actions. The set of possible actions A_s from a state $s_t = (n_t, \mathbf{l}_q, \mathbf{g}_t)$ consists of all outgoing edges of the vertex n_t in the KG. Formally, $A_s = \{(\mathbf{r}_i, \mathbf{u}_{r_i}, \mathbf{e}_{e'}) | (n_t, r_i, e') \in G\}$. This means an agent at each state has the option to select which outgoing edge it wishes to take having knowledge of the label of the edge r_i and destination vertex e'. Note that different from most of the existing methods (Lin and Socher; Kaiser et al., 2021) which only use $\mathbf{A}_s = (\mathbf{r}_i, \mathbf{e}_{e'})$, we also use the unique edge embedding $\mathbf{u}_{r_i}$. To allow the agent to have the option of ending a search, a self-loop edge is added to every entity. In addition, we also include the inverse relationship of a triple in the graph.

Transition. The transition function is defined as $\delta : S \times A \longrightarrow S$, which represents the probability distribution of the next states $\delta(s_{t+1}|s_t, a_t)$. In the current state s_t, the agent aims to choose proper actions a_t and then reach the next state $s_{t+1} = (n_{t+1}, \mathbf{l}_q, \mathbf{g}_{t+1})$. Both n_t and g_t are updated, while the query and answer remains the same.

Rewards. The model will receive the reward of $R_b(s_t) = 1$ if the current location is the correct answer and 0 otherwise. We set $\gamma = 1$ during the experiments.

4.5 Policy Network

The search policy is parameterized using state information and global context, including the search history. Specifically, every entity and relation in $\mathcal{G}$ is assigned a dense vector embedding $\mathbf{e} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ respectively. The action $a_t = (\mathbf{r}_{r_i}, \mathbf{u}_{r_i}, \mathbf{e}_{e'}) \in A_t$ is represented as the concatenation of the relation embedding, the unique edge embedding and the end node embedding.

The search history $(n_1 = v_{q_i}, r_1, n_2, ..., n_t) \in H$ consists of the sequence of observations and actions taken up to step t, and can be encoded using an LSTM (Hochreiter and Schmidhuber, 1997):

$$\mathbf{g}_0 = \text{LSTM}(0, [\mathbf{e}_{v_{q_i}} || \mathbf{l}_{q_i}])$$
$$\mathbf{g}_t = \text{LSTM}(\mathbf{g}_{t-1}, \mathbf{a}_{t-1}), t > 0$$

where $\mathbf{l}_{q_i}$ is the question embedding to form a start action with $\mathbf{e}_{v_{q_i}}$. The action space is encoded by stacking the embeddings of all actions in A_t: $\mathbf{A}_t \in \mathbb{R}^{|A_t| \times 3d}$. And the policy network π is defined as:

$$\pi_\theta(a_t|s_t) = \Psi(\mathbf{A}_t \times \mathbf{W_2}\text{ReLU}(\mathbf{W}_1[\mathbf{n}_t||\mathbf{l}_{q_i}||\mathbf{g}_t]))$$

where Ψ is the softmax operator.

4.6 Knowledge-Based Soft Reward

Due to the weak supervision in ConvQA, the agent will receive a positive reward until it arrives at the target entity. Such delayed and sparse rewards significantly slow the convergence. To address the issue of weak supervision and sparsity of rewards in ConvQA, we assign a soft reward to entities other than the target answer to measure the similarity between them. This helps speed up convergence and mitigate incompleteness in the KG. Specifically, the soft reward is used to measure the similarity between the current entity n_t identified by our model and the ground truth answer a^t. We use ComplEx (Trouillon et al., 2016) to learn the initial entity embedding and relation embedding for all nodes and edges in the knowledge graph. The probability that n_t is the correct answer is calculated by

$$Pr(n_t|\mathbf{l}_{q_i}, v_{q_i}, \mathcal{G}) = Re(< \mathbf{l}_{q_i}, \mathbf{e}_{n_t}, \bar{\mathbf{e}}_{v_{q_i}} >) \quad (5)$$

We propose the following soft reward calculation strategy

$$R(s_t) = R_b(s_t) + (1 - R_b(s_t))Pr(n_t|\mathbf{l}_{q_i}, v_{q_i}, \mathcal{G})$$

Namely, if the destination n_t is a correct answer according to $\mathcal{G}$, the agent receives reward 1. Otherwise the agent receives a fact score weighted by a pretrained distribution: $Pr(n_t|\mathbf{l}_{q_i}, v_{q_i}, \mathcal{G})$.

4.7 Training

Given a set of conversations, we want to return the best possible answers a^*, maximizing a reward $a^* = \text{argmax}_a \sum_C \sum_T R(a^i|q_i)$. The reward is computed with respect to the question q_i while the answer is provided in the train dataset. The goal is to maximize the expected reward of the answer returned under the policy $E_{a_1,...,a_T \sim \pi_\theta}[R(s_t)]$. Since it is difficult to compute the expectation, we use Monte Carlo sampling to obtain an unbiased estimate:

$$E_{a_1,...,a_T \sim \pi_\theta}[R(s_t)] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} R(s_t) \pi_\theta(a_t|s_t)$$

In the experiment, we approximate the expected reward by running multiple rollouts for each training example. The number of rollouts is fixed, We set this number to 20. We use REINFORCE (Williams, 1992) to compute gradients for training.

$$\nabla_\theta E_{a_1,...,a_T \sim \pi_\theta}[R(s_t)] = \sum_{i=1}^{T} \nabla_\theta \pi_\theta(a_t|s_t) R(s_t)$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} R(s_t) \nabla_\theta \log(\pi_\theta(a_t|s_t))$$

Additionally, to encourage diversity in the paths sampled by the policy during training, we add an entropy regularization term to our cost function, as proposed in (Kaiser et al., 2021).

$$H_{\pi_\theta}(.,s) = -\sum_{a \in A_s} \pi_\theta(a|s) \text{log} \pi_\theta(a|s)$$

$H_{\pi,\theta}$ is added to the cost function to ensure better exploration and prevent the agent from getting stuck in local optima. This final objective is:

$$E_{a_1,...,a_T \sim \pi_\theta}[R(s_t)] + \lambda H_{\pi_\theta}(.,s)$$

After training, in the testing phase, given a query, we rank all the entities in the KG based on their probabilities of being the correct answer. We let the policy network keep top-k most likely paths according to beam search, and we rank them according to their possibilities. For all the other entities which are not in the top-k candidates, we use ComplEx (Trouillon et al., 2016) to rank them according to Eq. (5).

5 Experiments

We use two datasets in the experiments: ConvQuestions (Christmann et al., 2019) and ConvRef (Kaiser et al., 2021). ConvQuestions contains a total of 6,720 conversations, each with 5 turns. ConvRef contains a total of 6,720 conversations, each with 5 turns. The code and datasets will be made publically available upon the acceptance of the paper. The details of these datasets can be found in Appendix.

Both ConvQuestions and ConvRef use Wikidata[2] as their background KG. However, the full Wikidata KG is extremely large, containing approximately 2 billion triples. Therefore, in the experiment, we sample a subset of triples from Wikidata. We first take the overlapped entities between the Wikidata and the QA datasets, and then we further obtain all the one-hop neighbours of these overlapped entities. The one-hop neighbors are retrieved from both the original data dump and also the entities' corresponding online Wikidata websites.

We compare the performance of our method, CORNNET, with four baselines: **Convex** (Christmann et al., 2019), **Conqer** (Kaiser et al., 2021), **OAT** and **Focal Entity** (Lan and Jiang, 2021). The details of all baselines can be found in Appendix.

Two LLMs are used to generate reformulations for the input query, which are GPT2 (Radford et al., 2018) and Bart (Lewis et al., 2019). For each input question, we generate multiple reformulations and use attention mechanism to aggregate them inside the model. We adopt the following ranking metrics which are also employed by the previous baselines: (1) Precision at the top rank (P@1); (2) Mean Reciprocal Rank (MRR); (3) Hit ratio at k (H@k/Hit@k). The details of all the datasets and experiment environment can be found in Appendix.

5.1 Main Results

In this subsection, we test CORNNET on conversational question answering tasks and compare it with other baseline methods.

A - Overall performance on ConvQA datasets. Table 2 compares the results of CORNNET with baselines on the ConvQuestions and ConvRef datasets. As we can see, CORNNET outperforms the baselines in both H@5 and MRR metrics On ConvQA. For H@5, CORNNET performs 4.5% bet-

[2] https://www.wikidata.org/wiki/Wikidata:Database_download

Table 1: Performance on different domain datasets.

Domain	Movies			TV Series			Music			Soccer		
Metric	H@3	H@5	H@8	H@3	H@5	H@8	H@3	H@5	H@8	H@3	H@5	H@8
Dataset	ConvQA											
Conv	0.345	0.345	0.345	0.303	0.308	0.309	0.213	0.217	0.221	0.2019	0.2019	0.2019
Conquer	0.336	0.343	0.354	0.375	0.396	0.421	0.279	0.282	0.289	0.325	0.343	0.350
CornNet	0.385	0.456	0.514	0.437	0.482	0.587	0.290	0.356	0.392	0.288	0.418	0.550
Dataset	ConvRef											
Conv	0.345	0.345	0.345	0.303	0.308	0.308	0.213	0.217	0.221	0.202	0.202	0.202
Conquer	0.389	0.404	0.429	0.435	0.442	0.485	0.371	0.398	0.413	0.371	0.384	0.393
CornNet	0.393	0.461	0.521	0.436	0.533	0.564	0.377	0.447	0.484	0.353	0.385	0.425

Table 2: Performance on ConvQA and ConvRef.

Dataset	ConvQA		ConvRef	
Model	Hit@5	MRR	Hit@5	MRR
CONVEX	0.219	0.200	0.257	0.241
CONQUER	0.372	0.327	0.427	0.382
OAT	-	0.260	-	-
Focal Entity	-	0.248	-	-
CornNet	0.417	0.337	0.477	0.353

ter than CONQUER and 20% better than CONVEX. In terms of MRR, CONVEX has the lowest performance, which is 13.7% worse than CornNet. CONQUER has the second highest performance, but it is also 1% lower than CornNet. For OAT, because its source code is not available, we directly adopt its results from (Marion et al., 2021). We can find that its MRR is 7.3% lower than that of CornNet. For Focal Entity, it has the third highest MRR. On the ConvRef dataset, CornNet also has similar performance. It achieves the highest Hit@5, which is 5% better than that of CONQUER. CornNet also has the second highest MRR compared with other baselines. Due to the unavailability of the OAT source code and the failure to run Focal Entity on the ConvRef dataset, we are unable to include their results in our analysis on the ConvRef dataset.

B - Performance on different domains. We further investigate the ranking performance of CornNet across different domains for both datasets. Table 1 illustrates detailed ranking results for H@3, H@5 and H@8. As the results show, CornNet outperforms baselines on most domains on the ConvQuestions dataset. On average, it achieves a 13.7% improvement in H@8 and 6.6% improvement in H@5 compared to the second highest baseline CONQUER. Similar results are also observed on the ConvRef dataset.

5.2 Ablation Studies and Efficiency Results

A - The effectiveness of the reformulations. In this subsection, we demonstrate the effectiveness of using different reformulations in the model. Two large language models are used to generate reformulations: GPT2 and Bart. When not using reformulations, the output of the Question Encoder is treated as the input of the LSTM directly. Table 3 shows the experiment results. As we can see, using reformulations can indeed increase the ConvQA performance most of the time. The performance of using GPT2 reformulations is very similar to that of using Bart reformulations. Using human writing reformulations has the best performance.

Table 3: The effectiveness of the reformulations.

Model	P@1	Hit@3	Hit@5	Hit@8
No Reformulation	0.231	0.373	0.445	0.474
GPT2 Reformulation	0.212	0.367	0.451	0.504
Bart Reformulation	0.221	0.376	0.441	0.494
Human Reformulation	0.257	0.395	0.463	0.518

B - The effectiveness of the teacher-student model. We further test the effectiveness of the proposed teacher-student model, shown in Table 4. If we only use the reformulations generated by LLMs, the performance is about 3% lower than that of teacher-student model. If we train the model on human written reformulations while tests on generated reformulations, the performance is about 1.3% lower than CornNet(the last row of Table 4).

Table 4: The effectiveness of the teacher-student model.

Model	P@1	Hit@3	Hit@5	Hit@8
GPT2 Reformulation	0.212	0.367	0.451	0.504
CornNet (GPT2)	0.265	0.404	0.477	0.526
Bart Reformulation	0.221	0.376	0.441	0.494
CornNet (Bart)	0.244	0.413	0.491	0.523
Train test data shift	0.237	0.389	0.472	0.507

6 Conclusion

In this paper, a model (CornNet) that creatively combines the question reformulation and reinforcement learning is proposed on a knowledge graph (KG) to attain accurate multi-turn conversational question answering. CornNet utilizes a teacher-student model and reinforcement learning agent to find answers from a KG. Experimental results demonstrate that CornNet surpasses existing methods on various benchmark datasets on conversational question answering.

7 Ethical Considerations

We have thoroughly evaluated potential risks associated with our work and do not anticipate any significant issues. Our conversational question answering framework is intentionally designed to prioritize usability and ease of implementation, thereby reducing barriers for adoption and minimizing operational complexities. Furthermore, it's essential to highlight that our research builds upon an open-source dataset. This ensures transparency, fosters collaboration, and helps address ethical considerations by providing accessibility to the underlying data.

8 Limitation

Our dataset exhibits several limitations that warrant consideration. Firstly, our training data is constrained by its limited scope, primarily focusing on specific domains rather than providing comprehensive coverage across diverse topics. This restriction may affect the model's generalizability and performance in addressing queries outside of these predefined domains. Moreover, while our knowledge graph serves as a valuable resource for contextual information, it is essential to acknowledge its incompleteness. Despite its vast size, certain areas within the knowledge graph may lack sufficient data or connections, potentially leading to gaps in the model's understanding and inference capabilities.

9 Acknowledgements

LL, BH and HT are partially supported by NSF (1947135,), DARPA (HR001121C0165), and NIFA (2020-67021-32799).

References

A Acharya and S Adhikari. 2021. Alexa conversations: An extensible data-driven approach for building task-oriented dialogue systems.

A Bordes and Usunier N. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*.

T Brown, B Mann, and N Ryder. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

C Buck and J Bulian. 2017. Ask the right questions: Active question reformulation with reinforcement learning.

P Christmann, Saha R, A Abujabal, J Singh, and G Weikum. 2019. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. CIKM '19.

R Das, S Dhuliawala, and M Zaheer. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

A Elgohary, D Peskov, and J Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. Association for Computational Linguistics.

A Fader, L Zettlemoyer, and O Etzioni. 2014. Open question answering over curated and extracted knowledge bases. KDD '14. Association for Computing Machinery.

D Guo, D Tang, N Duan, M Zhou, and J Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Etsuko Ishii, Yan Xu, Samuel Cahyawijaya, and Bryan Wilie. 2022. Can question rewriting help conversational question answering?

E Kacupaj, J Plepi, K Singh, and H Thakkar. Conversational question answering over knowledge graphs with transformer and graph attention networks. Association for Computational Linguistics.

E Kacupaj, K Singh, M Maleshkova, and J Lehmann. 2022. Contrastive representation learning for conversational question answering over knowledge graphs.

M Kaiser, R Saha Roy, and G Weikum. 2021. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. SIGIR '21. Association for Computing Machinery.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Y Lan and J Jiang. 2021. Modeling transitions of focal entities for conversational knowledge base question answering. Association for Computational Linguistics.

M Lewis, Y Liu, and N Goyal. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

X Lin and R Socher. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP 2018*.

Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge graph question answering with ambiguous query. In *Proceedings of the ACM Web Conference 2023*.

Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. 2021. Neural-answering logical queries on knowledge graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 1087–1097, New York, NY, USA. Association for Computing Machinery.

Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Lihui Liu, Zihao Wang, Ruizhong Qiu, Yikun Ban, Eunice Chan, Yangqiu Song, Jingrui He, and Hanghang Tong. 2024. Logic query of thoughts: Guiding large language models to answer complex logic queries with knowledge graphs.

P Marion, K Nowak, and F Piccinno. 2021. Structured context and high-coverage grammar for conversational question answering over knowledge graphs.

T Misu, K Georgila, A Leuski, and D Traum. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. Association for Computational Linguistics.

R Nogueira and K Cho. 2017. Task-oriented query reformulation with reinforcement learning.

A Radford, J Wu, and R Child. 2018. Language models are unsupervised multitask learners.

Romal Thoppilan, Daniel De Freitas, and Jamie Hall. 2022. Lamda: Language models for dialog applications. arXiv.

T Trouillon, J Welbl, and S Riedel. 2016. Complex embeddings for simple link prediction. ICML'16. JMLR.org.

S Vakulenko, S Longpre, Z Tu, and R Anantha. Question rewriting for conversational question answering. WSDM '21. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Ruijie Wang, Zheng Li, Dachun Sun, Shengzhong Liu, Jinning Li, Bing Yin, and Tarek Abdelzaher. 2022. Learning to sample and aggregate: few-shot reasoning over temporal knowledge graphs. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22.

Ruijie Wang, Zheng Li, Jingfeng Yang, Tianyu Cao, Chao Zhang, Bing Yin, and Tarek Abdelzaher. 2023. Mutually-paced knowledge distillation for cross-lingual temporal knowledge graph reasoning. In *Proceedings of the ACM Web Conference 2023*, WWW '23.

Ruijie Wang, Yutong Zhang, Jinyang Li, Shengzhong Liu, Dachun Sun, Tianchen Wang, Tianshi Wang, Yizhuo Chen, Denizhan Kara, and Tarek Abdelzaher. 2024. Metahkg: Meta hyperbolic learning for few-shot temporal reasoning. In *Proceedings of the 47th International Conference on Research and Development in Information Retrieval*, SIGIR '24.

Zihao Wang, Hang Yin, and Yangqiu Song. 2021. Benchmarking the combinatorial generalizability of complex query answering on knowledge graphs. *Proceedings of the 35th International Conference on Neural Information Processing Systems*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning.

Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, page 2485–2496, New York, NY, USA. Association for Computing Machinery.

Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2023. Reconciling competing sampling strategies of network embedding. In *Advances in Neural Information Processing Systems*, volume 36, pages 6844–6861. Curran Associates, Inc.

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021a. Dynamic knowledge graph alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4564–4572.

Yuchen Yan, Si Zhang, and Hanghang Tong. 2021b. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*, WWW '21, page 3907–3917, New York, NY, USA. Association for Computing Machinery.

Qixuan Zhang, Xinyi Weng, Guangyou Zhou, Yi Zhang, and Jimmy Xiangji Huang. 2022. Arl: An adaptive reinforcement learning framework for complex question answering over knowledge base. *Information Processing and Management*, 59(3):102933.

Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. *Computational Linguistics*, 46(1):53–93.

A Appendix

A.1 Dataset

We use two datasets in the experiments: ConvQuestions (Christmann et al., 2019) and ConvRef (Kaiser et al., 2021). ConvQuestions contains a total of 6,720 conversations, each with 5 turns, but no human writing reformulations are provided in this dataset. ConvRef contains a total of 6,720 conversations, each with 5 turns, and each question has several human writing reformulations. The statistics of these datasets are shown in Table 5.

Table 5: Summary of datasets.

Dataset	Train	Valid	Test
ConvQA	6,720	2,240	2,240
ConRef	6,720	2,240	2,240

A.2 Experiment Setting

All experiments are conducted on a machine with an Intel(R) Xeon(R) Gold 6240R CPU, 1510 GB memory, and an NVIDIA-SMI Tesla V100-SXM2 GPU. Network embedding has been studied for a long time (Yan et al., 2023, 2024), in this paper, we use ComplEx to learn the KG embeddings, we use a hidden dimension of 200 and train for 100 epochs with a batch size of 256. For CornNet, we set the hidden dimension to 200, the batch size to 12, the learning rate to 0.00002, and train for 20 epochs, storing checkpoints along the way. The Adam optimizer (Kingma and Ba, 2017) is used with fixed weight decay of 1.0. For the pre-trained BERT model, we set the maximum sentence size to 64 tokens, padding if necessary. The token dimension is 128.

A.3 Baselines

We compare the performance of our method, CornNet, with four baselines:

- **Convex** (Christmann et al., 2019): This method detects answers to conversational utterances over KGs in a two-stage process based on graph expansion. It first detects so-called frontier nodes that define the context at a given turn and then finds high-scoring candidate answers in the vicinity of the frontier nodes.

- **Conqer** (Kaiser et al., 2021): This is the current state-of-the-art baseline. It uses RL with reformulations to find answers in the KG. It employs multiple agents to find multiple answers for each question and ranks these answers in the end.

- **OAT** (Marion et al., 2021): This Transformer-based model takes a JSON-like structure as input to generate a Logical Form (LF) grammar that can model a wide range of queries on the graph. It finds answers by applying the LF.

- **Focal Entity** (Lan and Jiang, 2021): This is a novel graph-based model that captures transitions of focal entities and applies a graph neural network to derive a probability distribution of focal entities for each question. The probability distribution is then combined with a standard KBQA module to rank answers.

Note that Convex and Conqer utilize named entity recognition (NER) to identify topic entities in a question. Since we assume that the topic entity of q_1 is given, in order to ensure a fair comparison, we revise the NER method in Convex and Conqer so that it returns the main topic entity of q_1 directly.

A.4 The effectiveness of the unique relation embedding

We also test the effectiveness of using a unique edge embedding in the action space with reinforcement learning. 4 reformulations are used in the experiments, and the results are shown in Table 6. In this table, " CornNet -" denotes CornNet without using a unique edge embedding. As we can see, using a unique edge embedding improves the question answering performance by an average of 2.2%.

Table 6: The effectiveness of the unique edge embedding.

Model	P@1	Hit@3	Hit@5	Hit@8	MRR
CornNet -	0.227	0.396	0.469	0.507	0.330
CornNet	0.265	0.404	0.477	0.526	0.353

A.5 The effectiveness of the topic entity selector.

The topic entity selector is pre-trained using the training data After pre-training, the results of the accuracy for the classifier is 83.2%. We find that the classifier has a relatively high accuracy.

A.6 Efficiency.

Figure 3 shows the training time and test time of different methods on ConvRef dataset. As we can see, CORNNET has the shortest training and test time. While Conv has the longest training and test time. Despite the short training time of CORNNET, it still achieves better or comparable ConvQA performance compared to baselines.

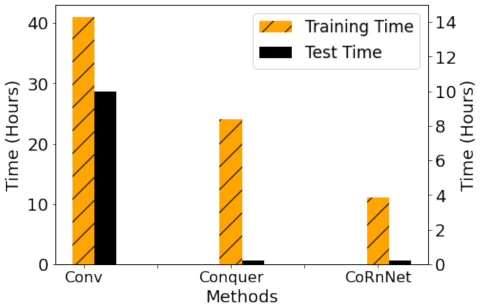

Figure 3: CORNNET Training and Test Time.

A.7 More details of CornNet

Figure 4 gives an example of reinforcement learning based KGQA method. In this example, we want to predict the relationship between "Steven Spielberg" and "Tom Hanks". Despite the KG is incomplete, the RL agent can correctly find the answer.

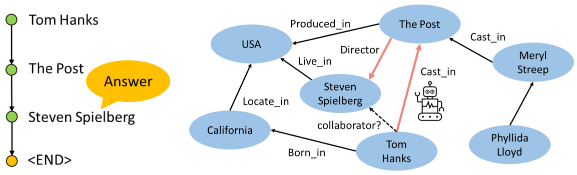

Figure 4: Reinforcement learning example.

Figure 5 provides a illustrative example of a reinforcement learning action space. In reinforcement learning, the action space refers to the set of all possible actions that an agent can take in a given environment. This figure demonstrates how the action space might look in a specific reinforcement learning scenario, highlighting the various options available to the agent at each step. Understanding the action space is crucial for designing effective reinforcement learning algorithms, as it directly impacts the agent's ability to learn and make decisions.

On the other hand, Figure 6 presents the framework of the teacher-student model, a popular approach in machine learning for knowledge distillation and transfer learning. In this framework, the teacher model serves as a knowledge source, providing high-quality predictions or representations

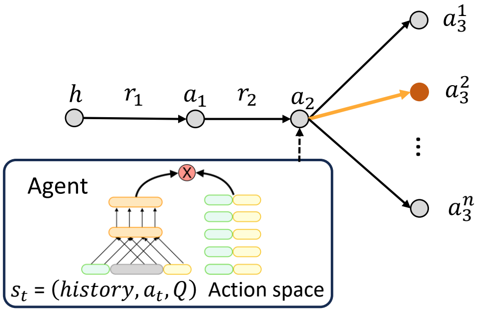

Figure 5: Reinforcement action space.

that the student model aims to learn from. The key distinction between the teacher and student models lies in their inputs: the student model receives reformulations generated by a large language model (LLM) as input, while the teacher model relies on human-written reformulations. This setup allows the student model to learn from both the teacher's expertise and the diversity of reformulations provided by the LLM, potentially leading to improved performance and generalization.

The combination of reinforcement learning and the teacher-student framework offers exciting possibilities for advancing machine learning algorithms. By leveraging the strengths of both approaches, we can design more robust and efficient systems.

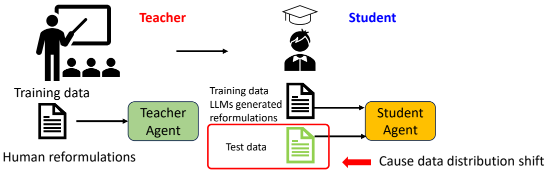

Figure 6: Teacher student model example.