Few-Shot Learning with Self-supervised Classifier for Complex Knowledge Base Question Answering

Bo Liu, Lei Liu, and Peiyi Wang(✉)

Lenovo Research, Beijing, China
{liubo22,liulei28,wangpy10}@lenovo.com

Abstract. Complex Question Answering (CQA) over Knowledge Base (KB) involves transferring natural language questions to a sequence of actions, which are utilized to fetch entities and relations for final answer. Typically, meta-learning based models regard question types as standards to divide dataset for pseudo-tasks. However, question type, manually labeled in CQA data set, is indispensable as a filter in the support set retrieving phase, which raises two main problems. First, preset question types could mislead the model to be confined to a non-optimal search space for meta-learning. Second, the annotation dependency makes it difficult to migrate to other datasets. This paper introduces a novel architecture to alleviate above issues by using a co-training scheme featured with self-supervised mechanism for model initialization. Our method utilizes a meta-learning classifier instead of pre-labeled tags to find the optimized search space. Experiments in this paper show that our model achieves state-of-the-art performance on CQA dataset without encoding question type.

Keywords: Meta-learning · Few-shot · Reinforcement learning · Question answering · Knowledge graph

1 Introduction

Knowledge-base question-answering (KBQA) task is defined as using facts stored in a knowledge base (KB) [9] to find the answer to a factoid question, which involves mapping natural-language questions to logical forms (annotations) including action sequences and programs that can be directly executed on a KB to retrieve answers [2,17]. Among the subfields of this topic, complex question answering (CQA) [16], different from multi-hop question answering, requires sophisticated operations such as set union and intersection, counting, min/max to be executed to obtain answers. In particular, this paper will focus on a CQA dataset [16], in which questions are classified into seven types. For instance, questions in the 'Simple' category only need 'select' to yield answers, while questions in the 'Quantitative Reasoning' category demand a sequence of 'select', 'union' and 'count' actions to return the answer.

Supported by Lenovo Research.

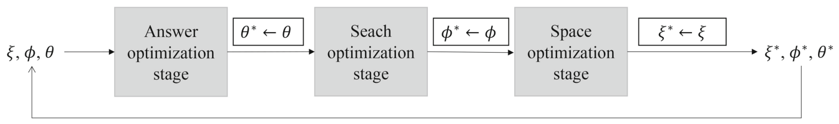

Fig. 1. The architecture of MACL

To solve KBQA problem, one-size-fits-all models-e.g., forward neural network, reinforcement learning, etc.-are regular approaches, which fit the training set with single one model and predict on questions of all types [1,2,4,16,21]. However, if questions have great diversity in logics or sentence structures, tacit knowledge may vary across them, leading to bias in predictions for one-size-fits-all model. To address this challenge, a neural program induction (NPI) approach is taken by Complex Imperative Program Induction from Terminal Rewards (CIPITR) [15]. Nevertheless, CIPITR still trains one one-size-fits-all model for seven question types each on CQA dataset instead of equipping the model with the ability to adapt to a certain form when faced with different kinds of questions.

In 2020, a new learning algorithm named MetA Retrieval Learning (MARL) [6] is proposed, in which a retriever and a programmer is jointly trained in two stages. The programmer is optimized in the stage one by meta-learning with retriever parameter fixed. In the second stage, the programmer will not be updated while the difference between support set used and not used can measure the effectiveness of the retriever, thus the retriever can be advanced by the reinforcement learning with the final reward offered by the programmer. In this way the retriever is encouraged to provide more accurate support set that is beneficial to the meta-task, rather than teacher forcing method in S2A, MARL is featured with this weak supervised training process. Besides, S2A employ multiple-round conversation as context, aiming to answer context-dependent questions. By comparison, questions in MARL are all single-round and KB can directly present the answer.

MARL achieves state-of-the-art performance at the time of publication, but this method cannot be well extended to other question answering datasets except CQA. It requires the question set to have accurate category labels which relies on expensive labor cost. This will also cause another problem that the algorithm limits the search space of the support set by filtering it with question type, whereas question with diverse types may also contribute valuable information.

To alleviate the above issues, this paper proposes a new architecture named MetA Classify Learning (MACL) – a model that utilizes a self-learning classifier instead of pre-labeled tags to find the most reasonable search space for support sets. Considering that the classification results could also participate in parameter updating as a part of the overall model adaptation, we extend the structure of MARL from two-stage joint training to three-stage joint training of retriever, programmer and classifier.

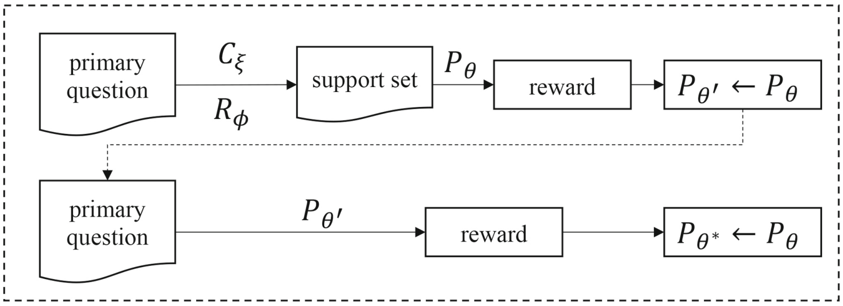

Fig. 2. Answer optimization stage

The classifier is used to predict the label of each question. As mentioned above, in order to reduce the bias introduced by manual labels, we do not use the question type of CQA data set. Instead, we cluster the unlabeled original data to obtain a rough classification as the pseudo-truth. Based on that, we train a classifier as a part of the model, which specifies the search space for meta-learning support set extraction. The parameters of classifier would be updated through reinforcement learning.

The retriever is a question selector that finds similar samples from a group of questions. KB artifacts (i.e., entities, relations, and types) and semantic similarity are used to measure the similarity between main question and support question. In each epoch, the whole data set is split into different groups according to the prediction results of the classifier, then the retriever would search similar samples in the same category for the primary question. The extracted questions together construct the support set for the pseudo task. During the training phase, reinforcement-learning [20] uses rewards from primary question along with pseudo-task to optimize the retriever.

The programmer is an answer generator. In the first step, the programmer encodes the problem into a sequence of actions. In the second step, these sequences are sent to the interpreter and executed to get the final answer in the knowledge base. Finally, the generated answers are evaluated on the ground-truth to produce rewards. Meta-learner trains the programmer to learn fast adaptation to a new primary question that has not been seen before.

The training process is divided into three stages, namely answer optimization stage, search optimization stage and space optimization stage. During these phrases, the programmer, the retriever and the classifier are updated respectively when the parameters of other modules remain fixed. Specially, the programmer is adapted by the meta-learning while the other two are updated by reinforcement-learning.

We evaluated our approach on a large complex QA data set named CQA [16]. MACL does not use manual category labels for dataset, which greatly reduces reliance on human labour. Therefore, this method can now be migrated to other datasets more easily. In terms of performance, our approach is superior to the

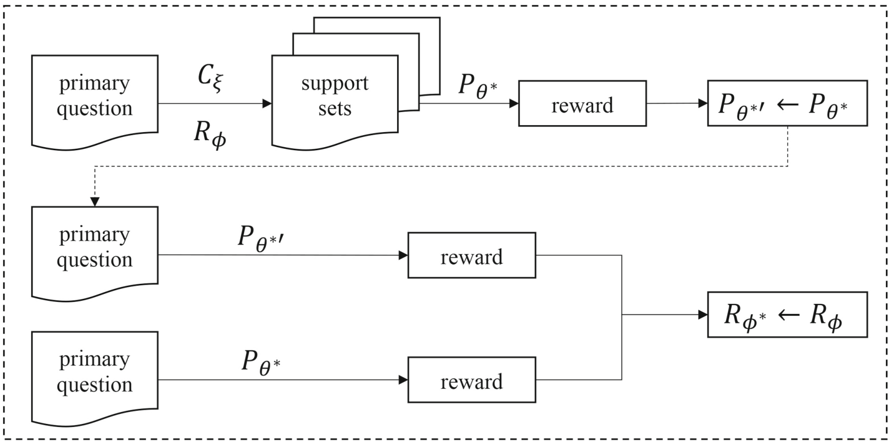

Fig. 3. Search optimization stage

standard imitation learning or RL-based models and achieves state-of-the-art F1 score compared to previous methods on out-of-question-type setting.

2 MACL

The task studied in this paper is Complex Question Answering (CQA) based on Knowledge Base (KB). We see this as a sequence-to-sequence learning task. The question is first transformed into a series of actions, then the triples are extracted from the knowledge base by the actions to obtain the final answer. We tackle this problem with few-shot meta-learning for classifier to decrease the reliance on data annotation and improve the transferability of the model. In this paper, we regard answering a single complex question as a separate task. Our goal is to train a self-learning model that can quickly adapt to new tasks. For this purpose, we leverage a meta-learner to complete each task. Specifically, for each primary question q_{pri}, we first assign a category label to it by classifier. Following that, the most similar N samples with the same category are retrieved to form a support set $S^{q_{pri}}$. Next, Question is decoded by the programmer to obtain the reward signal telling whether the programmer yields the correct answer. In the end, training rewards R_{train} from support set and testing rewards R_{test} from primary question are used to evaluate and update the model. The whole model is iteratively trained in three stages until the global optimal solution is obtained.

2.1 Overview of the Framework

Our architecture for few-shot learning of CQA is illustrated in Fig. 1. Our model adopts a three-stage joint optimization structure. Each stage consists of three working components, classifier, retriever and programmer. In the answer optimization stage, the parameters of BiLSTM-based programmer is updated by

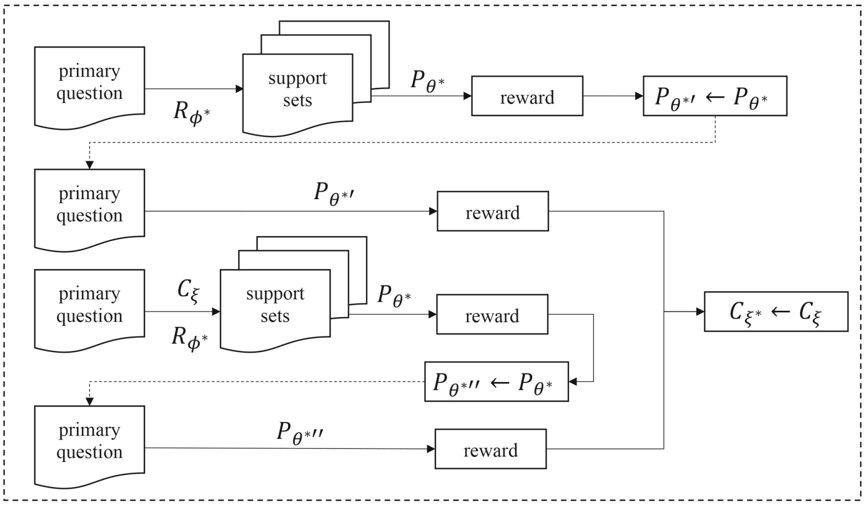

Fig. 4. Space optimization stage

the meta-learning. In the search optimization stage, the retriever based on Deep Structured Semantic Model (DSSM) method [7] is revised by the reinforcement-learning. In the space optimization stage, the parameters of previous modules are fixed and relevant rewards will renew the classifier, which is employed by the LSTM.

2.2 Algorithm

As shown in lines 3 to 9 in Algorithm 1, the programmer is trained by the approach of meta reinforcement learning (meta-RL) in the first stage. Similar to MARL, the gradient-based meta-learning method is employed to solve the meta-RL problem in this procedure. As the meta task is formulated in a RL setting, the vanilla policy gradient (VPG) [18] is performed to retrieve the optimal policy and Monte Carlo integration (MC) [5] strategy is applied for approximation.

For each primary question q_{pri} in training dataset, a support set $S^{q_{pri}}$ consisting of N secondary questions would be obtained after the classifier and the retriever are executed. Next, the primary question q_{pri} and the support set $S^{q_{pri}}$ together are considered as a meta-task. In the meta-training stage, each secondary question will produce K trajectories with probabilities, based on which the expectation of final reward on answering the secondary question can be calculated (Fig. 2). The sum of the rewards is subsequently utilized to adapt θ to θ',

$$L_{S^{q_{pri}}} = \textstyle\sum_{q_i \in S^{q_{pri}}} E_{\tau \sim \pi(\tau|q_i,\theta)}[R(\tau)] \tag{1}$$

$$\theta' \longleftarrow \theta + \eta_1 \nabla_\theta L_{S^{q_{pri}}} \tag{2}$$

During meta-test stage, another K trajectories of the primary question will be generated by programmer with θ'. The expectation of the reward, regarded as the evaluation of adapted parameter θ', is meant to be maximized. Thus, the base θ is further updated by this objective:

$$L_{q_{pri}} = E_{\tau\sim\pi(\tau|q_{pri},\theta')}[R(\tau)] \tag{3}$$

$$\theta \longleftarrow \theta + \eta_2 \nabla_\theta {\textstyle\sum}_{q_{pri}\in Q_{pri}} L_{q_{pri}} \tag{4}$$

To be more specific, the meta-learned policy is achieved by the Reptile meta-learning algorithm [Nichol and Schulman, 2018].

During the second stage, the retriever is updated by a reinforcement learning approach, represented in lines 10–20. First, each primary question directly yields the answer and corresponding reward through the model optimized in the stage 1. Second, following the probability of each secondary question being selected, M support sets are sampled from Q_s. Based on each support set, θ can be adapted to θ'_m, as the adaptation of θ to θ' in stage 1 (Fig. 3).

$$L_{S_m^{q_{pri}}} = {\textstyle\sum}_{q_i\in S_m^{q_{pri}}} E_{\tau\sim\pi(\tau|q_i,\theta)}[R(\tau)] \tag{5}$$

$$\theta'_m \longleftarrow \theta + \eta_1 \nabla_\theta L_{S_m^{q_{pri}}} \tag{6}$$

For each θ'_m, a trajectory corresponding to the primary question can be obtained and then a reward $R({\tau_m^*}')$ can be calculated.

$${\tau_m^*}' \longleftarrow decode(\pi(\tau|q_{pri},\theta'_m)) \tag{7}$$

After that, the difference between $R({\tau_m^*}')$ and $R(\tau^*)$ can be regarded as the evaluation of the significance of support set, which is exploited to update the retriever.

$$L'_{q_{pri}} = E_{S_m^{q_{pri}}\sim P(S^{q_{pri}}|\phi)}[R({\tau_m^*}') - R(\tau^*)] \tag{8}$$

$$\phi \longleftarrow \phi + \eta_3 \nabla_\phi {\textstyle\sum}_{q_{pri}\in Q_{pri}} L'_{q_{pri}} \tag{9}$$

Following that, space optimization stage will be performed. It is noted that support sets will be retrieved under two different circumstances for each primary question. The first type of support sets will be generated by retriever without the information of question type. In other words, the filter module in the retriever will not work in this condition such that all the questions in support dataset will be input into DSSM module. In contrast, for the retrieval of another kind of support set, the classifier C will first categorize all the questions, based on which DSSM will filter out question that has distinct type from the primary question (Fig. 4).

Algorithm 1. The MACL algorithm

```
Input: Support dataset $Q_s$, Training dataset $Q_{train}$, step size $\eta_1, \eta_2, \eta_3, \eta_4$
Parameter: a classifier C (LSTM) with parameter $\xi$, a retriever R (DSSM) with parameter $\phi$, a programmer P (LSTM) with parameter $\theta$
Output: The parameters $\xi^*$, $\phi^*$, $\theta^*$
 1: Initialize $\xi \longleftarrow \xi_0, \phi \longleftarrow \phi_0$, random initialize $\theta$
 2: while not converged do
 3:    for $Q_{pri}$ sampled from $Q_{train}$ do
 4:       for $q_{pri} \in Q_{pri}$ do
 5:          Retrieve $S^{q_{pri}}$ from $Q_s$ with $\xi$ and $\phi$
 6:          $\theta' \longleftarrow \theta + \eta_1 \nabla_\theta \sum_{q_i \in S^{q_{pri}}} E_{\tau \sim \pi(\tau|q_i,\theta)}[R(\tau)]$
 7:          $L_{q_{pri}} = E_{\tau \sim \pi(\tau|q_{pri},\theta')}[R(\tau)]$
 8:       $\theta \longleftarrow \theta + \eta_2 \nabla_\theta \sum_{q_{pri} \in Q_{pri}} L_{q_{pri}}$
 9:    for $Q_{pri}$ sampled from $Q_{train}$ do
10:       for $q_{pri} \in Q_{pri}$ do
11:          $\tau^* \longleftarrow decode(\pi(\tau|q_{pri},\theta))$, Compute $R(\tau^*)$
12:          Sample support sets $\mathcal{M}$ with $\xi$ and $\phi$
13:          for $S_m^{q_{pri}} \in \mathcal{M}$ do
14:             $\theta'_m \longleftarrow \theta + \eta_1 \nabla_\theta \sum_{q_i \in S_m^{q_{pri}}} E_{\tau \sim \pi(\tau|q_i,\theta)}[R(\tau)]$
15:             ${\tau_m^*}' \longleftarrow decode(\pi(\tau|q_{pri},\theta'_m))$, Compute $R({\tau_m^*}')$
16:          $L'_{q_{pri}} = E_{S_m^{q_{pri}} \sim P(S^{q_{pri}}|\phi)}[R({\tau_m^*}') - R(\tau^*)]$
17:       $\phi \longleftarrow \phi + \eta_3 \nabla_\phi \sum_{q_{pri} \in Q_{pri}} L'_{q_{pri}}$
18:    for $Q_{pri}$ sampled from $Q_{train}$ do
19:       for $q_{pri} \in Q_{pri}$ do
20:          Sample support sets $\mathcal{M}_0$ with $\phi$
21:          for $S_{m_0}^{q_{pri}} \in \mathcal{M}_0$ do
22:             $\theta'_{m_0} \longleftarrow \theta + \eta_1 \nabla_\theta \sum_{q_i \in S_{m_0}^{q_{pri}}} E_{\tau \sim \pi(\tau|q_i,\theta)}[R(\tau)]$
23:             ${\tau_{m_0}^*}' \longleftarrow decode(\pi(\tau|q_{pri},\theta'_{m_0}))$, Compute $R({\tau_{m_0}^*}')$
24:          Sample support sets $\mathcal{M}_1$ with $\xi$ and $\phi$
25:          for $S_{m_1}^{q_{pri}} \in \mathcal{M}_1$ do
26:             $\theta'_{m_1} \longleftarrow \theta + \eta_1 \nabla_\theta \sum_{q_i \in S_{m_1}^{q_{pri}}} E_{\tau \sim \pi(\tau|q_i,\theta)}[R(\tau)]$
27:             ${\tau_{m_1}^*}' \longleftarrow decode(\pi(\tau|q_{pri},\theta'_{m_1}))$, Compute $R({\tau_{m_1}^*}')$
28:          $R_0 = E_{S_{m_0}^{q_{pri}} \sim P(S^{q_{pri}}|\phi)} R({\tau_{m_0}^*}')$
29:          $L''_{q_{pri}} = E_{S_{m_1}^{q_{pri}} \sim P(S^{q_{pri}}|\xi,\phi)}[R({\tau_{m_1}^*}') - R_0]$
30:       $\xi \longleftarrow \xi + \eta_4 \nabla_\xi \sum_{q_{pri} \in Q_{pri}} L''_{q_{pri}}$
31: return $\xi, \phi, \theta$
```

After support sets are produced with these two approaches, θ will adapted to θ'_{m_0} and θ'_{m_1} respectively. The rewards of programmer in these two parameters answering primary question – this is $R({\tau_{m_0}^*}')$ and $R({\tau_{m_1}^*}')$– will have a difference, which represents the effect of classifier.

$$R_0 = E_{S_{m_0}^{q_{pri}} \sim P(S^{q_{pri}}|\phi)} R({\tau_{m_0}^*}') \tag{10}$$

$$L''_{q_{pri}} = E_{S_{m_1}^{q_{pri}} \sim P(S^{q_{pri}}|\xi,\phi)}[R({\tau_{m_1}^*}') - R_0] \tag{11}$$

The difference will subsequently be applied to update the parameter of Classifier with reinforcement learning.

$$\xi \longleftarrow \xi + \eta_4 \nabla_\xi \textstyle\sum_{q_{pri} \in Q_{pri}} L''_{q_{pri}} \tag{12}$$

2.3 Objective Function with Reinforcement Learning

Among all procedures, it is worth mentioning that the parameters of DSSM and LSTM for classifier are updated by the final rewards, which are simply values without gradient. Hence, it is vital to design reasonable objective functions to combine rewards and the output of models, in which case the loss can back-propagate and the gradient descent can be operated.

For the DSSM part, the loss function is designed as Eq. (8) such that:

$$\begin{aligned} L'_{q_{pri}} &= E_{S_m^{q_{pri}} \sim P(S^{q_{pri}}|\phi)}[R(\tau_m^{*\prime}) - R(\tau^*)] \\ &= \sum_{m=1}^{M} [R(\tau_m^{*\prime}) - R(\tau^*)] \cdot P_\phi(S_m^{q_{pri}}) \\ &= \sum_{m=1}^{M} [R(\tau_m^{*\prime}) - R(\tau^*)] \cdot \prod_{n=1}^{N} P_\phi(q_n) \end{aligned} \tag{13}$$

In the above result, $P_\phi(q_n)$ is the output of the DSSM. Compared with the original loss function as the Eq. (14), in our model there are no clearly defined positive and negative samples. Instead, we have rewards representing the contribution of the candidates. Hence, the model is encouraged to choose better support sets by maximizing the objective function (13).

$$L_{DSSM} = -\log \prod_{(Q,D^+)} P(D^+|Q) \tag{14}$$

$$\text{where } D^+ \text{ are positive responses}$$

Similar to DSSM, the objective function of LSTM for classification are reasonably adjusted from its original form.

$$\begin{aligned} L''_{q_{pri}} &= E_{S_{m_1}^{q_{pri}} \sim P(S^{q_{pri}}|\xi,\phi)}[R(\tau_{m_1}^{*\prime}) - E_{S_{m_0}^{q_{pri}}} R(\tau_{m_0}^{*\prime})] \\ &= \sum_{m_1=1}^{M} [R(\tau_{m_1}^{*\prime}) - \sum_{m_0=1}^{M} R(\tau_{m_0}^{*\prime}) \cdot P_\phi(S_{m_0}^{q_{pri}})] \cdot P_{\phi,\xi}(S_{m_1}^{q_{pri}}) \\ &\triangleq \sum_{m_1=1}^{M} R^* \cdot \prod_{n=1}^{N} P_\phi(q_n) \cdot P_\xi(q_n) \end{aligned} \tag{15}$$

Except for $P_\xi(q_n)$, the rest part, which is approximately regarded as the label value in the common multi-label classification task, decides the direction $P_\xi(q_n)$ is expected to be optimized.

3 Evaluation

In this section, we describe our experimental settings and results on the large-scale CQA dataset [16]. During the experiments, our model is evaluated against with three baselines. In addition, we conduct ablation experiments to investigate the capability of the classifier and the effect of the cluster.

3.1 CQA Dataset

The CQA dataset is generated from the Wikidata KB [19] through a semi-automated process with manual labour. It contains 944K/100K/156K training/validation/test question-answer pairs respectively. There are seven question categories in CQA dataset which are described as follows:

– **Simple Question** can be answered with one hop in the knowledge graph.
– **Logical Reasoning Question** requires several logical deduction over multiple tuples in the knowledge graph.
– **Quantitative Reasoning Question** requires a large amount of quantitative reasoning which includes max, min, at least/at most/approximately/equal to N, etc.
– **Comparative Reasoning Question** requires a comparison between entities based on certain relations which include sort and more/less operations.
– **Verification (Boolean) Question** needs to judge whether the question is correct and return answer of Boolean data type.
– **Quantitative (Count) Question** requires quite a few quantitative reasoning steps which involve standard aggregation count functions.
– **Comparative (Count) Question** requires a comparison between entities based on certain relations and count operations.

For questions whose types are “Verification”, “Quantitative (Count)” and “Comparative (Count)”, we use “accuracy” as the evaluation metric while “F1 measure” demonstrates the performance of other kinds.

3.2 Comparison Methods

In this paper, several baseline methods on the CQA dataset are applied for comparison, including **HRED+KVmem** [16], **CIPITR** [15] and a variant of MARL [6] named **MARL (no-type)**. MARL proposed a novel meta-learning based algorithm for few-shot CQA task. In MARL, a retrieval model and a programmer model are trained jointly with weak supervision. To evaluate the performance of MARL when the question types are not given, we implemented MARL (no-type) which retrieves similar questions as support data from all question types.

To validate the benefits of each module in MACL, we conducted ablation experiments about the variants of our model. To verify the effect of clustering, we trained a model named **MACL (no-cluster)**. In MACL (no-cluster), each question in CQA dataset is randomly assigned a type rather than being assigned by the clustering result. Similar to MACL, the labeled questions in MACL (no-cluster) are used to pretrain the classifier. Another model **MACL (no-classifier)** is trained for exploring the benefits of the classifier. In this model, question type is annotated based on the clustering result in the beginning and would not be updated in the rest process. Besides, we trained the model named **MACL (random)** to verify the whole effectiveness of cluster and classifier. In MACL (random), we removed the classifier in MACL and randomly initialized the type of questions in CQA dataset. Notably, the number of question types by random initialization is the same as those based on the clustering result.

Table 1. Experimental results on the CQA test dataset. For each category, best result is **bolded** and second-best result is <u>underlined</u>.

Question category	HRED +KVmem	CIPITR	MARL (no-type)	MACL (no-cluster)	MACL (no-classifier)	MACL (random)	MACL
Simple	41.40%	41.62%	<u>85.62%</u>	85.00%	84.94%	85.02%	**85.98%**
Logical reasoning	37.56%	21.31%	<u>79.68%</u>	79.25%	78.57%	78.79%	**80.22%**
Quantitative reasoning	0.89%	5.65%	44.09%	<u>44.97%</u>	44.95%	44.23%	**45.65%**
Verification (Boolean)	27.28%	30.86%	84.99%	85.35%	<u>85.45%</u>	85.32%	**85.84%**
Comparative reasoning	1.63%	1.67%	61.75%	**62.52%**	<u>61.92%</u>	61.61%	61.56%
Quantitative (count)	17.80%	37.23%	62.17%	<u>62.23%</u>	62.21%	62.17%	**62.25%**
Comparative (count)	9.60%	0.36%	39.69%	39.59%	<u>39.71%</u>	39.42%	**40.29%**
Overall macro F1	19.45%	19.82%	65.43%	<u>65.56%</u>	65.39%	65.22%	**65.97%**
Overall micro F1	31.18%	31.52%	<u>76.06%</u>	75.80%	75.66%	75.64%	**76.50%**

3.3 Implementation Details

In MACL, the classifier is pre-trained to speed up the convergence of the full model. To generate the pseudo label for the pre-training of classifier, we adopted K-Means [12] clustering algorithm to cluster questions in CQA dataset. We averaged the embedding representations of each word in a question to obtain the question embeddings and considered it as K-Means input. Specifically, we exploited 50-dimensional GloVe word vectors [14] learned from web crawled data for word embeddings. To determine the value of k in K-means, we calculated the clustering result with SSE, which is defined as the sum of the squared distance between centroid and each member of the cluster. On top of that, elbow method were utilized to select the best k value. Based on the changes in SSE with varying k, we chose $k = 3$ as the number of groups and then annotated the questions into 3 categories.

MACL was implemented in PyTorch. Reptile [13] was employed to simplify the implementation of MAML [3] and avoid the calculation of second-order derivatives which requires magnificent computing expenses. During the stages of retriever and classifier learning, RL-based model optimized the non-differentiable objective by combining the traditional loss function with the rewards, in which the baseline C was imported to reduce the variance and C was set to $1e-3$. We fixed the step size $\eta_1 = 1e-4$ in the meta-training stage for all the learning stages. In contrast, different learning rates were set for the meta-test in varying stages. During the answer optimization stage, we let $\eta_2 = 0.1$ for the gradient descent. For search optimization stage, the initial learning rate η_3 was set to $1e-3$ and AdaBound [11] was applied to update the ϕ with the reward derived from programmer. In the space optimization stage, the Adam optimization algorithm [10] optimized ξ and the learning rate was set as $\eta_4 = 1e-3$. Additionally, the number of N for searching the top-N support questions in meta-training stage was set to 5. In the training of MACL, apart from the natural language questions, the entity, entity type and relation in the question were also encoded as the part of input. The batch size was set to 1 and our model converged on the validation set at round 80 epochs.

3.4 Performance Evaluation

In this part, we compare the proposed method with baseline algorithms described in Sect. 3.2 and offer further analysis. Table 1 presents the performance comparison on the CQA dataset. From the results, we can observe that our model MACL outperforms all the baseline models and achieves the state of the art performance. This is thanks to the fact that the classifier in MACL can be dynamically updated and provide the optimal search space for meta-learning. To understand the impact of each part in MACL on performance, we conducted additional experiments on CQA dataset and the results are shown in Table 2. The results demonstrate that the overall micro F1 of MACL (no-cluster) is 0.70% less than that of MACL, which implies the cluster result could pre-train the classifier and make it reach the optimal state sooner. The MACL (no-classifier) achieves 75.66% F1 score, which is 0.84% less than the micro F1 of MACL. It proves that even though the question type is effectively initialized, classifier still needs updating to group the questions better. The MACL (random) performs the worst among the three variants, illustrating that the cluster operation and the classifier updating are all necessary for better performance.

Table 2. Ablation study on the CQA test dataset.

Models	Overall micro F1
MACL	76.50%
- cluster	−0.70%
- classifier	−0.84%
- cluster&classifier	−0.86%

4 Related Work

CQA. For CQA task, HRED+KVmem [16] is proposed together with CQA dataset by Saha et al. In this model, the current sentence and context is first encoded into a vector by a hierarchical encoder-decoder. Then the most relevant memory is retrieved by a key-value memory network. Finally, the model decode the relevant memory and generate the question answer. CIPITR [15] adopts auxiliary rewards to strengthen training signal which can alleviate extreme reward sparsity. Meanwhile, to generate semantically reasonable programs for a certain question, it designs high-level constraints.

Meta-learning. The Meta-learning approaches is proposed for training a base model on a variety of learning task, making new learning task rapidly adapted with only a small number of training instances [3]. In question-answer domain, a method called PseudoTask MAML (PT-MAML) use the top-K relevant examples to build support task for each question in the training dataset [8]. Furthermore, S2A [4] utilizes a retriever to form support sets and update it and the programmer separately. Aiming to update the retriever better, MARL [6] construct a jointly trained structure with reinforcement learning. However, their methodology heavily relies on the information of question type. To make the model more universal, we propose a three-stage loop-learning structure including a classification module which can be jointly adapted in the process.

5 Conclusion

In this paper, we propose a self-classified meta-reinforcement learning method for complex question answering over KB. Our model is capable of effectively adapting to new questions based on the most similar questions retrieved from a reasonable search space. To obtain the optimal search space boundary, the model constructs a semi-supervised classifier to assign each question a label. Thus, our model addresses two essential challenges – the significant distributional biases presented in the dataset, and the high cost associated with manual labelling of question type. When evaluated on the large-scale complex question answering dataset CQA without question types, our model achieves state-of-the-art performance with overall macro and micro F1 score of 65.97% and 76.50%. In the future, we plan to extend the model to other domains that refers to meta-learning.

References

1. Ansari, GA., Saha, A., Kumar, V., Bhambhani, M., Sankaranarayanan, K., Chakrabarti, S.: Neural program induction for KBQA without gold programs or query annotations. In: IJCAI, pp. 4890–4896 (2019)
2. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1533–1544 (October 2013). https://aclanthology.org/D13-1160
3. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning, pp. 1126–1135. PMLR (2017)

4. Guo, D., Tang, D., Duan, N., Zhou, M., Yin, J.: Dialog-to-action: conversational question answering over a large-scale knowledge base. In: Advances in Neural Information Processing Systems, pp. 2942–2951 (2018)
5. Guu, K., Pasupat, P., Liu, E.Z., Liang, P.: From language to programs: bridging reinforcement learning and maximum marginal likelihood. arXiv preprint arXiv:1704.07926 (2017)
6. Hua, Y., Li, Y.F., Haffari, G., Qi, G., Wu, W.: Retrieve, program, repeat: complex knowledge base question answering via alternate meta-learning. arXiv preprint arXiv:2010.15875 (2020)
7. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 2333–2338 (2013)
8. Huang, P.S., Wang, C., Singh, R., Yih, W., He, X.: Natural language to structured query generation via meta-learning. arXiv preprint arXiv:1803.02400 (2018)
9. Jin, H., Li, C., Zhang, J., Hou, L., Li, J., Zhang, P.: XLORE2: large-scale cross-lingual knowledge graph construction and application. Data Intell. **1**(1), 77–98 (2019)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Luo, L., Xiong, Y., Liu, Y., Sun, X.: Adaptive gradient methods with dynamic bound of learning rate. arXiv preprint arXiv:1902.09843 (2019)
12. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
13. Nichol, A., Schulman, J.: Reptile: a scalable metalearning algorithm. arXiv preprint arXiv:1803.02999 2(3), 4 (2018)
14. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
15. Saha, A., Ansari, G.A., Laddha, A., Sankaranarayanan, K., Chakrabarti, S.: Complex program induction for querying knowledge bases in the absence of gold programs. Trans. Assoc. Comput. Linguist. **7**, 185–200 (2019)
16. Saha, A., Pahuja, V., Khapra, M.M., Sankaranarayanan, K., Chandar, S.: Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: 32nd AAAI Conference on Artificial Intelligence (2018)
17. Shen, T., et al.: Multi-task learning for conversational question answering over a large-scale knowledge base. arXiv preprint arXiv:1910.05069 (2019)
18. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063 (2000)
19. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
20. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3), 229–256 (1992)
21. Yih, W., He, X., Meek, C.: Semantic parsing for single-relation question answering. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 643–648 (2014)