ATAP: Automatic Template-Augmented Commonsense Knowledge Graph Completion via Pre-Trained Language Models

Fu Zhang[†], **Yifan Ding**[†], **Jingwei Cheng**[*]
School of Computer Science and Engineering, Northeastern University, China
{zhangfu,chengjingwei}@mail.neu.edu.cn; dingyifan@stumail.neu.edu.cn

Abstract

The mission of commonsense knowledge graph completion (CKGC) is to infer missing facts from known commonsense knowledge. CKGC methods can be roughly divided into two categories: triple-based methods and text-based methods. Due to the imbalanced distribution of entities and limited structural information, triple-based methods struggle with long-tail entities. Text-based methods alleviate this issue, but require extensive training and fine-tuning of language models, which reduces efficiency. To alleviate these problems, we propose ATAP, the first CKGC framework that utilizes automatically generated continuous prompt templates combined with pre-trained language models (PLMs). Moreover, ATAP uses a carefully designed new prompt template training strategy, guiding PLMs to generate optimal prompt templates for CKGC tasks. Combining the rich knowledge of PLMs with the template automatic augmentation strategy, ATAP effectively mitigates the long-tail problem and enhances CKGC performance. Results on benchmark datasets show that ATAP achieves state-of-the-art performance overall.[1]

1 Introduction

Commonsense knowledge is information that humans typically have that helps them make sense of everyday situations (Storks et al., 2019). Commosense knowledge graphs (CKGs) are powerful representation of real-world commonsense knowledge. CKGs such as ConceptNet (Speer et al., 2017), ATOMIC (Sap et al., 2019), and Dense-ATOMIC (Shen et al., 2023) provide a structured way to represent commonsense concepts based on triples, consisting of head nodes, tail nodes and relationship edges (i.e., <head entity, relation, tail entity>). Unlike traditional knowledge graphs (e.g., Dbpedia (Lehmann et al., 2015) and Wikidata (Vrandečić and Krötzsch, 2014)), CKGs consist of nodes that are represented by *free-form text*. Such CKGs have been widely used to build commonsense-grounded AI applications, such as search engines (Lu et al., 2020), question answering (Yasunaga et al., 2021), and recommendation systems (Zhang et al., 2016).

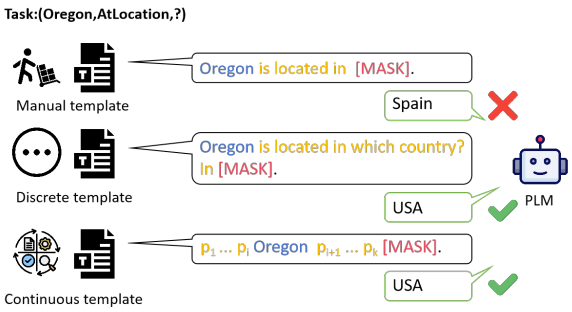

Figure 1: The entity, prompt and [MASK] are spliced together to obtain a template, which is input into a PLM to complete the prediction task of [MASK]. *Manual* template are manually constructed from relations (e.g., AtLocation). *Discrete* templates is the optimal template for completing the task, which are selected through a large number of experiments on different templates. *Continuous* template p_i $(i = 1, \ldots, k)$, unlike before, does not represent a specific word. It is a continuous, vectorized, trainable pseudo-label that is input into the PLM in a special form.

However, existing CKGs are typically constructed through manual or semi-automated methods, resulting in most CKGs are far from complete, which affects the performance. Accordingly, *commonsense knowledge graph completion* (CKGC) is proposed to solve the inherent incompleteness problem of CKGs by predicting missing entities in incomplete triples (also called link prediction).

Existing CKGC work mainly focuses on triple-based methods (e.g., SGBC (Malaviya et al., 2020) and InductivE (Wang et al., 2021a)), which utilize the structure of CKGs as the only source of infor-

[1]The code and datasets are available at https://github.com/neu-dyf/ATAP_code.
[†]Equal contribution. [*]Corresponding Author.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 16456–16472
November 12-16, 2024

mation. But CKGs are usually sparse because there are a large number of entities with in-degree 1 in triples of the CKGs (Wang et al., 2021a). These existing methods struggle with long-tail entities (Wang et al., 2022) due to limited structural information (information scarcity) and imbalanced entity distributions (long-tail entities are prevalent).

Pre-trained language models (PLMs) (e.g., BERT (Kenton and Toutanova, 2019), GPT-2 (Radford et al., 2019)) have extensive knowledge bases. These huge training text corpus can be used as additional knowledge for many natural language processing tasks (AlKhamissi et al., 2022; Petroni et al., 2019; Zhang et al., 2020a), and may also be used to alleviate information scarcity for long-tail entities. In particular, PLMs have proven to be very powerful in predicting [MASK] in masked language model (MLM) learning (Kenton and Toutanova, 2019). The work (Lv et al., 2022) provides a reliable evaluation demonstrating the potential of PLMs to handle traditional knowledge graph completion tasks. They achieve this by converting the traditional knowledge graph completion task into an MLM task, and manually constructing prompt templates to guide the PLM to complete the MLM task. Therefore, a straightforward approach applied to CKGC might be to incorporate PLMs with manually constructed prompt templates.

However, *manual prompt templates* have stochasticity in the performance of PLMs. As shown in Figure 1, the answer to the $(Oregon, AtLocation, ?)$ using a manual template is incorrect. Manually building a top-performing prompt template can be difficult, time-consuming and costly. The recent method in open-world knowledge graph completion domain (Jiang et al., 2023) proposes to automatically generate *discrete prompt templates* to complete the prediction of [MASK]. However, neural networks are continuous, using discrete prompt templates may be not optimal (Liu et al., 2023). Instead of manual or discrete prompt templates used in traditional close/open-world knowledge graph completion tasks, we propose an approach of automatically generating *continuous prompt templates* as depicted in Figure 1, and leverage it to the commonsense knowledge graph completion.

In our study, for the first time, we propose a framework that enhances commonsense knowledge graph completion by automatically generating continuous prompt templates. This framework innovatively extends prompt learning (a strategy of in-context learning (Liu et al., 2023)) into CKGC to convert the triple form of commonsense knowledge in CKGs into the text form that PLMs are good at processing. Compared with the existing triple-based methods, our framework leverages the knowledge in CKGs and PLMs simultaneously, effectively alleviating the information scarcity of long-tail entities.

The main contributions can be summarized as follows:

- We propose a new **A**utomatic **T**emplate-**A**ugmented commonsense knowledge graph completion framework via **P**re-trained language models (ATAP). ATAP is the first model to combine template generation with PLMs for CKGC.
- We propose a method to automatically generate continuous prompt templates to replace traditional manual and discrete templates, which improves the performance of CKGC and reduces the cost of manual labor.
- Moreover, we propose a new prompt template training strategy specifically designed for relations in CKGC, aiming to guide the model to generate an optimal prompt template corresponding to each relation in the CKGC task.
- Extensive experiments on CKGC datasets (ConceptNet-100k and ATOMIC) show that ATAP achieves state-of-the-art performance overall. More extensive experiments on other traditional KGC datasets also verify that the continuous template we proposed performs better than manual and discrete templates and has strong robustness.

2 Related Work

2.1 Knowledge Graph Completion

Traditional KGC methods. Traditional methods can be mainly divided into *three categories*: translation-based methods, tensor decomposition-based methods and neural network-based methods. Translation-based methods operate by mapping entities and relations in knowledge graphs into vector space, such as TransE (Bordes et al., 2013), TransR (Lin et al., 2015), and RotatE (Sun et al., 2018). Tensor decomposition-based methods, including RESCAL (Nickel et al., 2011), DisMult (Schlichtkrull et al., 2018), ComplEx (Trouillon et al., 2016), Tucker (Kolda and Bader, 2009), learn

representations of entities and relations by representing knowledge graphs as tensors and decomposing the tensors. Neural network-based methods utilize the powerful representation capabilities of neural networks to learn the embedding of entities and relations in knowledge graphs, including ConvE (Dettmers et al., 2018), R-GCN (Schlichtkrull et al., 2018). Moreover, KG-BERT (Yao et al., 2019) is a model based on the PLM's BERT. It converts the triple (head, relation, tail) into text sequences to solve KGC task.

Commonsense KGC methods. SGBC (Malaviya et al., 2020) uses graph neural networks to learn CKG structural information and uses PLMs for entity embedding to enhance the representation of entities. InductivE (Wang et al., 2021a) further solves the prediction problem of unseen entities through a novel densification process based on SGBC. Both of them solve the long tail problem by adding similar edges, but adding a large number of similar edges may also introduce noise. CNPC-S and CNPC-I (Wu et al., 2023) propose to contrastive pre-training and node clustering. Contrastive pre-training builds positive and negative node pairs and uses contrastive learning to obtain better semantic representations of nodes, while node clustering aggregates nodes with the same concept to alleviate the sparsity of CKG. Bi-CoRPe (Pan et al., 2024) builds contextual sentences by designing templates. It captures the first-order logic reasoning path and leverages context to enhance the representation of PLMs.

2.2 Language Model Prompting

As mentioned in Section 2.1, researchers have applied PLMs to the task of CKGC, but mainly used PLMs to directly encode entities (Malaviya et al., 2020; Wang et al., 2021a) or used PLMs to fine-tune knowledge graphs (Yao et al., 2019). None of these works leverage PLM's huge training text corpus to alleviate the information scarcity problem of long-tail entities that is widely present in CKGC tasks.

Recent studies have solved this problem by constructing prompt templates. Jiang et al. (2020) *manually* construct prompt templates for traditional KG completion, but this method depends on the quality of the manual prompt templates. Jiang et al. (2023) propose to automatically construct *discrete* prompt templates for open-domain KG completion. However, due to the continuity of neural networks, it is very difficult and time-consuming to find the optimal prompt template in discrete space.

Currently, there is no work on solving commonsense knowledge graph completion by automatically generating templates. In this paper, we propose a novel method to automatically construct continuous prompts in combination with a pre-trained language model to solve CKGC tasks.

3 ATAP

This section first introduces the definition of CKGC, and then we propose our ATAP framework, consisting of two key modules: Continuous Prompt Automatic Generation and Prediction, as depicted in Figure 2 and described in detail later.

3.1 Problem Formulation

The goal of CKGC is to predict missing triples that objectively exist to commonsense knowledge graphs. A CKG is typically represented as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}$ is the set of all entities, $\mathcal{R}$ is the set of all relations, and $\mathcal{T}$ is the set of all factual triples. $\mathcal{T} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where h and t represent head and tail entities, and r represents a relation. CKGC includes two types of tasks. One type of task is *triple classification*, which is to determine whether the triple (h, r, t) to be predicted belongs to $\mathcal{T}$. Another type of task is *link prediction*. For a given missing triple $(h, r, ?)$ or $(?, r, t)$, predict the entity represented by $?$ that is more consistent with the facts. Our model is used to solve the link prediction task.

3.2 Continuous Prompt Automatic Generation

3.2.1 CKG Triple Processing

Triple Classification. Given multiple triples with a same relation r, instead of generating different prompt templates for these triples, we generate a unified prompt template based on the relation r. All triples with this relation r will use this template, which greatly reduces the number of templates.

To achieve this, we first classify the set of triples $\mathcal{T}$ in a CKG according to the relations r_i ($r_i \in \mathcal{R}, 1 \leq i \leq N$). Then, we stack the triples with the same relation r_i together to form the corresponding $\mathcal{T}_i$ ($1 \leq i \leq N$), and finally classify the $\mathcal{T}$ as follows:

$$\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\} = \texttt{classify}(\mathcal{T}), \quad (1)$$

where $\mathcal{T}_i$ is the set of triples of the relation r_i, and N represents the number of relations. During the

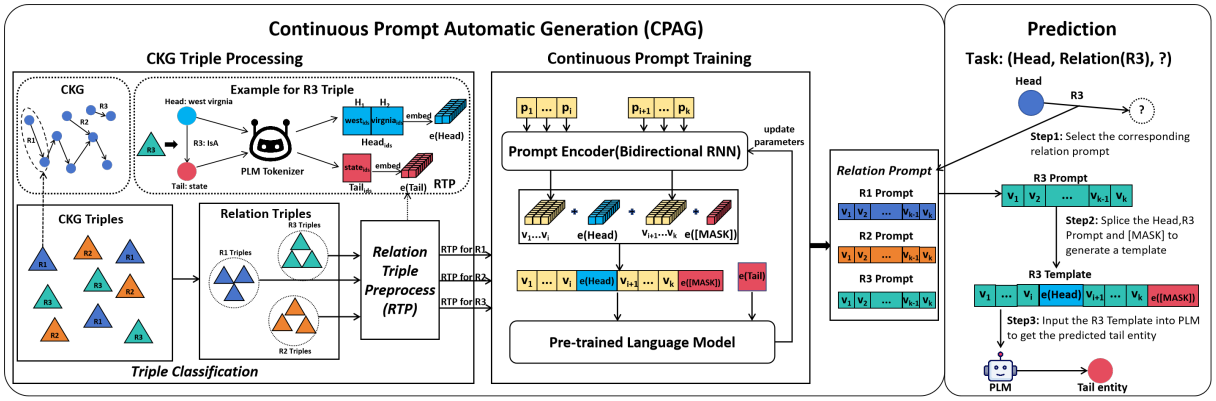

Figure 2: Our model ATAP consists of two modules: **Continuous Prompt Automatic Generation** (**CPAG**) and **Prediction**. In the CPAG module, the CKG Triples are first classified according to relations to obtain triples under each relation. Then, the relation triples are preprocessed to obtain the embeddings e(Head) and e(Tail) of the head and tail entities. Next, the pseudo-label p_i $(i = 1, \ldots, k)$ is encoded to obtain its pseudo-mark v_i $(i = 1, \ldots, k)$. The pseudo-marks, the head entity embedding e(Head), and [MASK] are concatenated into a prompt template and input into the PLM for training. Finally, the optimal prompt under each relation is obtained in this way. In the Prediction module, the CKGC task is completed with the help of the prompt of each relation following three steps.

training phase, we generate an identical continuous prompt template for all triples in $\mathcal{T}_i$.

Relation Triple Preprocess (RTP). The purpose of RTP is to further obtain the embeddings of the head and tail entities of each triple in $\mathcal{T}_i$. We use PLMs' tokenizer to segment the entities:

$$h_{\text{ids}} = \texttt{PLM\_Tokenizer}(h), \tag{2}$$

$$t_{\text{ids}} = \texttt{PLM\_Tokenizer}(t). \tag{3}$$

Further, the corresponding embeddings of the head and tail entities can be obtained as follow:

$$\mathbf{E}_h = \texttt{PLM\_Model}(h_{\text{ids}}), \tag{4}$$

$$\mathbf{E}_t = \texttt{PLM\_Model}(t_{\text{ids}}), \tag{5}$$

where the embedding of each token (after the segmentation of the head entity h and the head entity t) is concatenated to obtain the final embedding $\mathbf{E}_h$ of h and $\mathbf{E}_t$ of t.

3.2.2 Continuous Prompt Training

Given a link prediction task $(h, r, ?)$, it can be converted into an MLM task. Traditional methods (Jiang et al., 2020, 2023) manually construct prompt templates by introducing entity descriptions. As we have pointed out in Section 2.2, these methods are not only time-consuming, but also unstable or random in performance. Instead, inspired by (Liu et al., 2023), we propose to train continuous prompts to make our model more expressive and robust. Continuous prompts are composed of pseudo-label p_i $(1 \le i \le k)$. Unlike manual prompt templates, the pseudo-label p_i is not an exact word in form, but a prompt label. Its representation is randomly initialized and can be inserted anywhere in the prompt template.

The pseudo-label p_i $(1 \le i \le k)$ generates the pseudo-mark v_i $(1 \le i \le k)$ through the Prompt Encoder layer. The pseudo-mark v_i is a continuous and trainable parameter. We splice the pseudo-marks, the embedding of the header entity $\mathbf{E}_\text{h}$ and the embedding of [MASK] into a prompt template $\texttt{prompt}_T(h, r)$:

$$\begin{aligned}\texttt{prompt}_T(h, r) = \{&v_1, \ldots, v_i, \mathbf{E}_h, \\ &v_{i+1}, \ldots, v_k, \mathbf{E}([\text{MASK}])\},\end{aligned} \tag{6}$$

where k is the number of pseudo-marks. Intuitively, we believe that these pseudo-marks v_i are dependent on each other and have a sequential relation. Therefore, we employ bidirectional RNN (Grossberg, 2013) as the Prompt Encoder to learn the association among the pseudo-marks v_i. Moreover, considering that the processed pseudo-mark v_i is highly discrete and easily falls into the local optimal solution. In order to solve this problem, we use multilayer perceptron (MLP) for optimization, which is shaped like:

$$\begin{aligned}v_i &= \texttt{MLP}([\overrightarrow{v_i} : \overleftarrow{v_i}]) \\ &= \texttt{MLP}([\texttt{RNN}(v_{1:i}) : \texttt{RNN}(v_{i:k})]).\end{aligned} \tag{7}$$

During the training process, given a commonsense triple (h, r, t), we use the special mark [MASK] to replace the tail entity t. Our goal is to input $\texttt{prompt}_T(h, r)$ into the PLM to predict the

correct tail entity t. Using the pre-trained language model, the prediction logits $\mathbf{P}_t$ can be obtained:

$$\mathbf{P}_t = \texttt{PLM\_Model}(\texttt{prompt}_T(h, r)). \quad (8)$$

Then, we compute the cross-entropy loss between the prediction logits $\mathbf{P}_t$ and the embedding $\mathbf{E}_t$ corresponding to the tail entity t to be predicted as follows:

$$\mathcal{L}_t = \texttt{CrossEntropy}(\mathbf{P}_t, \mathbf{E}_t), \quad (9)$$

where $\texttt{CrossEntropy}(\mathbf{x}, \mathbf{y}) = -\sum_j \mathbf{x}_j \log \mathbf{y}_j$. Then $\mathcal{L}_t$ is used as the loss of the prompt template $\texttt{prompt}_T(h, r)$ to optimize the pseudo-marks v_i as:

$$\hat{v}_{0:k} = \arg\min_v \mathcal{L}_t. \quad (10)$$

The training process is presented in Appendix A. In this way, a specific prompt of the relation r_i $(1 \leq i \leq N)$ in the CKG can be obtained respectively.

3.3 Prediction

For a given commonsense knowledge graph completion task $(Head, Relation, ?)$, the goal is to predict the missing tail entity through the head entity $Head$ and the relation $Relation$. As shown in Figure 2, first find the corresponding prompt R_Prompt according to the relation $Relation$. Then concatenate the head entity, the specific prompt R_Prompt and [MASK] together to generate a template. Finally, input the template into PLM to predict the missing tail entity.

4 Experiment

In this section, we evaluate the ability of ATAP on the CKG completion task: link prediction. We compare ATAP with traditional and commonsense completion models. In addition, we conduct ablation experiments on the ATAP model. Finally, we further analyze each module of ATAP.

4.1 Experimental Setup

Datasets. We use two standard evaluation datasets CN-100K and ATOMIC for our experiments. CN-100K (Li et al., 2016) is a dataset that encompasses general commonsense knowledge. This version contains the Open Mind Common Sense (OMCS) entries from ConceptNet (Speer and Havasi, 2013). ATOMIC (Sap et al., 2019) is an atlas of everyday commonsense reasoning and primarily focuses on event-level commonsense knowledge in the form of if-then relations. The detailed information of these two datasets is in Table 14 in the Appendix.

Baselines. We compare ATAP with the state-of-the-art baselines as detailed in Section 2, including:

- **KGC methods:** DisMult (Schlichtkrull et al., 2018), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2018), ConvE (Dettmers et al., 2018), ConvTransE (Shang et al., 2019), COMET (Bosselut et al., 2019).
- **CKGC methods:** RGAT (Wang et al., 2020), SGBC (Malaviya et al., 2020), InductivE (Wang et al., 2021a), CNPC-S and CNPC-I (Wu et al., 2023), Bi-CoRPe (Pan et al., 2024).

Evaluation Metrics. For evaluating the performance of ATAP, we use standard evaluation metrics for the CKGC task, which are also adopted by previous baselines. These metrics include: (*i*) Hits@n $(n = 1, 3, 10)$ measures whether the model ranks the correct entity in the top n positions for each query. (*ii*) MRR is the average of the reciprocal rankings of all queries and is used to measure the performance of the model in the ranking task. The larger these metrics are, the better the performance of the model.

Implementation Details. During the training process, we choose the Adam optimizer for optimization, and in order to better converge to the global optimal solution, we set an exponential decay learning rate scheduler to 0.98 and the learning rate to 1e-5. We used a batch size of 64 and trained on CN-100K and ATOMIC datasets for 200 epochs. We evaluate the MRR of the validation set for each epoch. We selected the model checkpoint that achieved the highest MRR on the validation set for testing.

4.2 Main Result

The results are shown in Table 1. As can be seen, the proposed ATAP achieve state-of-the-art performance on both CN-100K and ATOMIC in most metrics.

Specifically, on the CN-100K and ATOMIC datasets, ATAP achieves state-of-the-art performance on all metrics except Hits@1. In particular, on CN-100K, Hits@3 is improved by 1.79% and on ATOMIC, Hits@10 is improved by 0.88%. The results show that our proposed framework combining pre-trained language model (PLM) and automatically generated continuous prompt templates can further improve the overall performance of CKGC.

Models	CN-100K				ATOMIC			
	MRR	$Hits@1$	$Hits@3$	$Hits@10$	MRR	$Hits@1$	$Hits@3$	$Hits@10$
DistMult	8.97	4.51	9.76	17.44	12.39	9.24	15.18	18.30
ComplEx	11.40	7.42	12.45	19.01	14.24	13.27	14.13	15.96
ConvE	20.88	13.97	22.91	34.02	10.07	8.24	10.29	13.37
RotatE	24.72	-	28.20	45.41	11.16	-	11.54	15.60
ConvTransE	18.68	7.87	23.87	28.95	12.94	12.92	12.95	12.98
COMET	6.07	0.08	2.92	21.17	3.36	0.00	2.15	15.75
RGAT+BERT$_{large}$	41.03	27.90	47.74	67.21	-	-	-	-
RGAT+SIM+BERT$_{large}$	43.97	30.75	51.54	69.34	-	-	-	-
SGBC$_{\text{BERT+ConvTransE}}$	49.56	38.12	55.5	71.54	12.33	10.21	12.78	16.20
SGBC$_{\text{GCN+ConvTransE}}$	49.12	37.71	56.67	71.29	10.25	8.72	10.54	13.26
SGBC$_{\text{SIM+GCN+ConvTransE}}$	29.80	21.25	33.04	47.50	13.12	10.70	13.74	17.68
SGBC$_{\text{GCN+BERT+ConvTransE}}$	50.38	38.79	56.46	72.96	10.8	9.04	11.21	14.10
SGBC$_{\text{SIM+GCN+BERT+ConvTransE}}$	51.11	39.42	59.58	73.59	10.33	8.41	10.79	13.86
InductivE	56.92	45.54	63.38	78.63	13.19	10.26	13.61	18.83
CNPC-S	54.52	45.33	61.46	75.92	13.14	10.11	13.75	18.80
CNPC-I	59.00	**48.29**	65.04	79.13	14.38	10.53	15.22	21.79
Bi-CoRPe	57.24	46.75	65.66	77.67	15.36	**13.53**	15.83	18.68
ATAP	**59.13**	47.54	**66.83**	**80.48**	**15.53**	12.76	**16.23**	**22.67**

Table 1: Performance comparison on Concept-100k and ATOMIC datasets, with the highest scores in bold. The results of all other baselines are from (Pan et al., 2024).

However, the introduction of the PLM's knowledge brings additional noise, which may interfere with the model's ability to rank accurately predicted entities very high when performing link prediction tasks, resulting in ATAP's performance on Hits@1 being lower than the baseline.

In addition, to test the performance of the ATAP model in the Large Language Models (LLMs), we replace the Pre-Trained Language Models (PLMs) in ATAP with the LLMs (LLAMA3.1-8B and Qwen2-7B) as the base for the experiment. Detailed results and analyses of the experiments are provided in Appendix B.

4.3 Ablation Study

To better understand ATAP, we conduct an ablation study on CN-100K and ATOMIC to show the effectiveness of different components in Table 2. Specifically, we evaluate ATAP without CKG Triple Classification (i.e., we will generate a common prompt for all triples, denoted as "w/o classification"), without fine-tuning the PLM (denoted as "w/o fune-tuning"). We also replace our automatic continuous template with manually constructed template in Table 16 in the Appendix (denoted as "w/o continuous template").

Ablation study for CKG Triple Classification. CKG triple classification is a key pre-processing step for subsequently obtaining the optimal prompt for each relation. To study the impact of CKG Triple Classification on ATAP, we remove this module. We directly input CKG triples into the model to obtain a **common prompt for all triples**. Experimental results show that removing this module has a cliff-like effect on the performance of ATAP. MRR dropped by **33.9**% On CN-100K and **8.2**% on ATOMIC. The common prompt is to aggregate all triple knowledge into one prompt, which can solve a wider range of CKGC tasks. However, for a CKGC task $\mathcal{T} = (h, r, ?)$, it will be affected by other relation triple knowledge besides r.

Ablation study for fine-tuning. In ATAP, we fine-tune the pre-trained language model using CN-100K and AOTMIC to enable it to have the ability of CKG completion. In order to study the impact of fine-tuning on ATAP, we use the original ability of the PLM to solve the CKGC completion task. Hits@10 decreased by **28.52**% on CN-100K and **7.44**% on ATOMIC. This result shows that it is difficult to achieve the best effect of the PLM without fine-tuning.

Ablation study for automatic continuous prompt and manual prompt. To study the impact of continuous template on ATAP's performance, we manually constructed a template for each relation in CN-100K and ATOMIC. Experimental results show that MRR decreased significantly overall (CN-100K decreased by **14.06**%, ATOMIC de-

Model	CN-100K				ATOMIC			
	MRR	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
ATAP	**59.13**	**47.54**	**66.83**	**80.48**	**15.53**	**12.76**	**16.23**	**22.67**
$\text{ATAP}_{\text{w/o classification}}$	25.23	16.08	29.58	42.25	7.33	4.36	7.34	11.32
$\text{ATAP}_{\text{w/o fune-tuning}}$	38.26	30.02	41.03	51.96	10.33	7.26	11.36	15.23
$\text{ATAP}_{\text{w/o continuous template}}$	45.07	32.69	53.88	66.72	11.53	8.39	11.73	15.79

Table 2: Ablation results of ATAP.

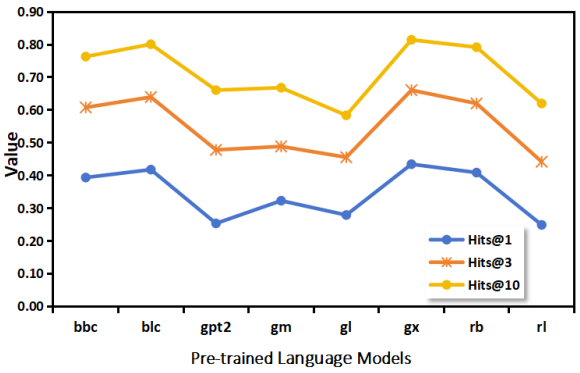

Figure 3: Performance of our ATAP under different PLMs on CN100k, where [bert-base-cased(bbc), bert-large-cased(blc)] are Bert series; [gpt2, gpt2-medium(gm), gpt2-large(gl), gpt2-xl(gl)] are GPT2 series; and [robert-base(rb), robert-large(rl)] are RoBerta series.

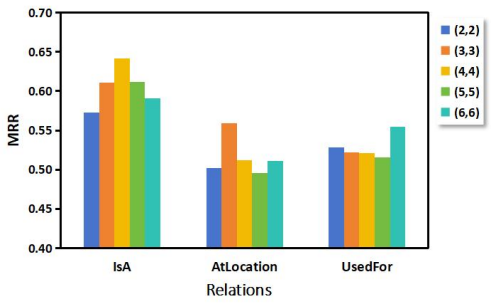

Figure 4: MRR of different k on three relations (IsA, Atlocaton, UsedFor).

creased by **4**%), and Hits@n $(n = 1, 3, 10)$ also decreased to varying degrees. Compared with manual template, automatic continuous template can modify template in time during training, thereby obtaining the optimal template to solve CKGC.

4.4 Further Study

Next, we analyze five questions: the choice of ATAP's base for PLMs, the detailed selection of models in Prompt Encoder, the optimal value of k for each relation, the generalization of ATAP and the impact of number of relations on performance.

Q1: How to select a suitable PLM as the base for `ATAP`*?* On the CN-100K dataset, we test Hits@n $(n = 1, 3, 10)$ of ATAP on 8 PLMs (the parameter of each PLM is shown in Table 15 in the Appendix). The experimental results are shown in Figure 3, the training parameters of each series of PLMs increase from left to right. In theory, their performance should also increase. The BERT and GPT2 series basically conforms to this rule, while the RoBerta series shows a decreasing trend. Considering that the previous CKGC models uses the BERT model. In order to align with it, ATAP also uses bert-large-cased as the base.

Q2: How to select the optimal neural network model as the base for `Prompt Encoder`*?* The quality of prompt encoding is very important for the PLMs to complete CKGC. Four models were used in Prompt Encoder for testing, and the results are shown in Table 4. We believe that the pseudo-marks generated by prompt encoding are interdependent. Each pseudo-mark should be affected by the contextual pseudo-marks. The bidirectional model can better extract the the contextual pseudo-marks information of each pseudo-mark. In theory, the bidirectional model is better than the unidirectional model in CKGC.

In Table 4, this conclusion is verified. Bidirectional RNN is better than unidirectional RNN in various evaluation metrics. The same is true for bidirectional LSTM and unidirectional LSTM. Both RNN and LSTM can capture contextual information in sequence data. However, RNN is more effective when processing short sequences, and prompts are represented by shorter sequences. Therefore, for ATAP, bidirectional RNN is more suitable for CKG completion tasks than bidirectional LSTM. And it is 3.34% higher on Hits@3.

Models	FB15k-237				WN18RR			
	MRR	$Hits@1$	$Hits@3$	$Hits@10$	MRR	$Hits@1$	$Hits@3$	$Hits@10$
RESCAL	0.356	0.266	0.390	0.535	0.467	0.439	0.478	0.516
TransE	0.279	0.198	0.376	0.441	0.243	0.043	0.441	0.532
DisMult	0.241	0.155	0.263	0.419	0.430	0.390	0.440	0.490
ComplEx	0.247	0.158	0.275	0.428	0.440	0.410	0.460	0.510
RotatE	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571
TuckER	0.358	0.266	0.394	0.544	0.470	0.443	0.482	0.526
HAKE	0.346	0.250	0.381	0.542	0.497	0.452	0.516	0.582
CompGCN	0.355	0.264	0.390	0.535	0.479	0.443	0.494	0.546
HittER	0.344	0.246	0.380	0.535	0.496	0.449	0.514	0.586
Pretrain-KGE	0.332	-	-	0.529	0.235	-	-	0.557
KG-BERT	-	-	-	0.420	0.216	0.041	0.302	0.524
StAR	0.263	0.171	0.287	0.452	0.364	0.222	0.436	0.647
MEM-KGC(w/o EP)	0.339	0.249	0.372	0.522	0.533	0.473	0.570	0.636
MEM-KGC(w EP)	0.346	0.253	0.381	0.531	0.557	0.475	0.604	0.704
KICGPT	0.412	**0.327**	0.448	0.554	0.549	0.474	0.585	0.641
KICGPT$_{tsa}$	0.410	0.321	0.430	0.581	0.564	**0.478**	0.612	0.677
ATAP	**0.424**	0.316	**0.458**	**0.607**	**0.571**	0.465	**0.629**	**0.719**

Table 3: Performance comparison on traditional KG completion datasets FB15k-237 and WN18RR, with the highest scores in bold. The results of all other traditional KGC models are from (Wei et al., 2023).

Model	CN-100K			
	MRR	Hits@1	Hits@3	Hits@10
blc+unidLSTM	52.82	38.53	61.88	79.23
blc+bidLSTM	53.87	40.87	61.55	79.57
blc+unidRNN	53.47	40.37	61.80	78.32
blc+bidRNN	**55.35**	**41.62**	**64.89**	**79.90**

Table 4: The impact of changing the Prompt Encoder model on blc. [bidLSTM, unidLSTM, bidRNN, unidRNN] represent [Bidirectional LSTM, unidirectional LSTM, bidirectional RNN, unidirectional RNN].

k	CN-100K	ATOMIC
(2,2)	Causes, Desires, ReceivesAction, DefinedAs	xWant,oWant
(3,3)	AtLocation, HasSubevent, HasPrerequisite, NotCapableOf, NotIsA, HasFirstSubevent, CreateBy	xNeed
(4,4)	IsA, PartOf	xEffect,oEffect
(5,5)	CapableOf, HasProperty, CausesDesire, MotivateByGoal, NotHasProperty	xIntent, xReact, oReact
(6,6)	UsedFor, HasA, MadeOf	xAttr

Table 5: The optimal prompt size for each relation in CN-100K and ATOMIC. $(2, 2)$ means that k is equal to 4 and the template is transformed into the following form: $\{v_1, v_2, e(Head), v_3, v_4, e([MASK])\}$.

Q3: How to determine the k value in the pseudo-mark v_i $(1 \leq i \leq k)$ for each relation prompt? To analyze the impact of k on model performance, we select different $k \in \{4, 6, 8, 10, 12\}$ values and test MMR on each relation, and finally determine the optimal k value for each relation as shown in Table 5. In order to show the impact of different k on the MRR of each relation, we select three relations on CN-100K for display, and the results are shown in Figure 4. It can be seen that the best k for [IsA, Atlocation, UsedFor] is [8, 6, 12].

Moreover, our experiments reveal that the best performance is to *evenly* insert the pseudo-marks $\{v_1, ..., v_k\}$ into the template as shown in Appendix C. Therefore, the best template form for [IsA, Atlocation, UsedFor] is [(4,4), (3,3), (6,6)].

Q4: How does ATAP perform on other datasets that are not specifically presented to evaluate the CKGC task? We select traditional KG completion datasets (FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018)) for experiment. The results are shown in Table 3. Considering that some models in Table 3 are not proposed for CKGC and have not been evaluated on the CKGC task, we do not include them in the previous comparative experiments (i.e., Table 1). Detailed introduction of the datasets and the experimental settings for Q4 can be found in the Appendix E. It can be seen that from Table 3, ATAP achieves the best performance on most metrics. On FB15k-237, MRR is improved by 1.2% and Hits@10 is improved by 2.6%. On WN18RR, Hits@10 is improved by 1.5%. This once gain demonstrates that our ATAP framework based on automatic continuous prompts has good generalization ability.

Q5: How does the performance change with

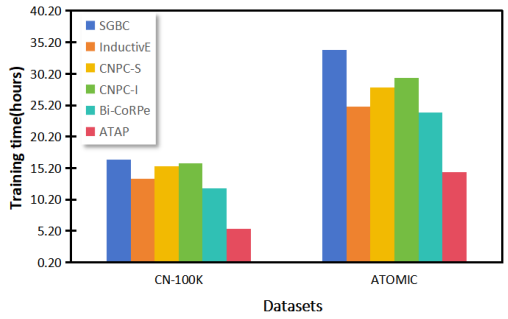

Figure 5: Comparison of training time on CN-100K and ATOMIC datasets.

varying numbers of relations? In ATOMIC, the number of relations is relatively small, whereas ConceptNet contains a larger number of relations. It is important to investigate whether the number of relations affects the performance of ATAP. To demonstrate whether the performance varies with the number of relations, we conduct two sets of experiments as shown in Appendix F.

4.5 Efficiency Analysis

To study the efficiency of ATAP, we compare with existing models in several aspects. First, ATAP takes less training time on CKGC datasets as shown in Figure 5, similar to the traditional KGC dataset FB15k-237 (Figure 8, Appendix D). Second, Figures 9 and 10 in Appendix D illustrate that ATAP converges faster than other baselines on CN-100K and FB15K-237 (even if the number of relations in FB15K-237 is large, which has 237 relations).

5 Conclusion

In this study, we propose a new framework ATAP to solve the CKGC task using automatically generated continuous prompts combined with PLMs. In ATAP, we propose a method to automatically generate continuous prompt templates to replace traditional manual and discrete templates. Moreover, we propose a new prompt template training strategy specifically designed for relations in CKGC, wihch can guide the model to generate an optimal prompt template corresponding to each relation in the CKGC task. Extensive experiments on CKGC datasets show that ATAP achieves state-of-the-art performance overall and reduces the cost of manual labor. Experiments on traditional datasets also verify the robustness of our model. In our near future work, we intend to continue investigating the performance of LLMs on CSKG tasks.

Limitations

We currently evaluate our method on CKGC datasets without considering scenarios such as temporal knowledge graph completion (Garcia-Duran et al., 2018) and few-shot knowledge graph completion (Xiong et al., 2018). In future research, we plan to study the effectiveness of our method in other scenarios.

Acknowledgments. The authors sincerely thank the anonymous reviewers for their valuable comments and suggestions, which improved the paper. The work is supported by the National Natural Science Foundation of China (62276057), and Sponsored by CAAI-MindSpore Open Fund, developed on OpenI Community.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. Hitter: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Bonggeun Choi, Daesik Jang, and Youngjoong Ko. 2021. Mem-kgc: Masked entity model for knowledge graph completion with pre-trained language model. *IEEE Access*, 9:132025–132032.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

Stephen Grossberg. 2013. Recurrent neural networks. *Scholarpedia*, 8(2):1888.

Pengcheng Jiang, Shivam Agarwal, Bowen Jin, Xuan Wang, Jimeng Sun, and Jiawei Han. 2023. Text augmented open knowledge graph completion via pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11161–11180.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Shuqi Lu, Zhicheng Dou, Chenyan Xiong, Xiaojie Wang, and Ji-Rong Wen. 2020. Knowledge enhanced personalized search. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval*, pages 709–718.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pretrained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2925–2933.

Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 3104482–3104584.

Yudai Pan, Jun Liu, Tianzhe Zhao, Lingling Zhang, and Qianying Wang. 2024. Context-aware commonsense knowledge graph reasoning with path-guided explanations. *IEEE Transactions on Knowledge and Data Engineering*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3027–3035.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3060–3067.

Xiangqing Shen, Siwei Wu, and Rui Xia. 2023. Dense-atomic: Towards densely-connected atomic with high knowledge coverage and massive multi-hop paths. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. *The People's Web Meets NLP: Collaboratively Constructed Language Resources*, pages 161–176.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Shane Storks, Qiaozi Gao, and Joyce Y Chai. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*, pages 1–60.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *ICLR*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Bin Wang, Guangtao Wang, Jing Huang, Jiaxuan You, Jure Leskovec, and C-C Jay Kuo. 2021a. Inductive learning on commonsense knowledge graph completion. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021b. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238.

Xintao Wang, Qianyu He, Jiaqing Liang, and Yanghua Xiao. 2022. Language models as knowledge embeddings. *arXiv preprint arXiv:2206.12617*.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. Kicgpt: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683.

Siwei Wu, Xiangqing Shen, and Rui Xia. 2023. Commonsense knowledge graph completion via contrastive pretraining and node clustering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13977–13989.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362.

Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020a. Empower entity set expansion via language model probing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8151–8160.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3065–3072.

Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. 2020c. Pretrain-kge: learning knowledge representation from pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 259–266.

A Detailed prompt training process

Algorithm 1: Training process of ATAP.

Input: CKG training triple set $\mathcal{T}_{\text{train}}$, verification triple set $\mathcal{T}_{\text{valid}}$, and test triple set $\mathcal{T}_{\text{test}}$.

Output: Specific prompt r_prompt for each relation r.

```
Classify T_train, T_valid, and T_test according to the relation r, assuming that the total number of relations is N;
for relation_index ← 1 to N do
    Initialize model parameters;
    for epoch ← 1 to max_epoch_num do
        for step ← 1 to max_step_num do
            b ← sample the training batch;
            for (h, r, [MASK]) in b do
                Splice input template Eq. (6);
                Calculate cross entropy loss Eq. (9);
            Obtain the overall loss and optimize models;
        Validate ATAP using the validation set of r;
        if early_stop > k then break;
    Test ATAP using the test set of r;
    Obtain the optimal prompt r_prompt of r;
```

B Performance of Large Language Models in ATAP

The experimental results are shown in Table 6. it can be seen that LLAMA3.1-8B and Qwen2-7B achieve the best results in some indicators. However, considering that fine-tuning LLMs requires a lot of computing resources and time, and most mainstream CSKG completion baseline methods (such as InductivE (Wang et al., 2021a) and SGBC (Malaviya et al., 2020)) are based on bert-large-cased with smaller parameters. Therefore, in our current work, for fair comparison, we also choose bert-large-cased as the base.

C Experiments of how to insert the pseudo-marks $\{v_1, ..., v_k\}$ into template

In the pseudo-labels $\{v_1, ..., v_k\}$ of length k, we need to split them into two parts to generate the input template $\{v_1, ..., v_i, e(Head), v_{i+1}, ..., v_k, e([MASK])\}$. In order to study the impact of different templates of k on MRR, we test each relation on CN-100K. For the convenience of presentation, we show the experimental results of IsA and Atlocation, as shown in Figures 6 and 7. It can be seen that the trend of MRR is to increase first and then decrease, and the value is the highest at the bisection point of k. Therefore, the performance of inserting the pseudo-marks into the template in *equal* parts is the best.

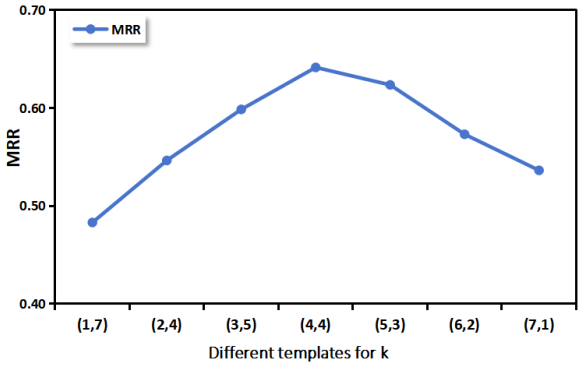

Figure 6: The relation is IsA. The impact of different templates on MRR when $k = 8$.

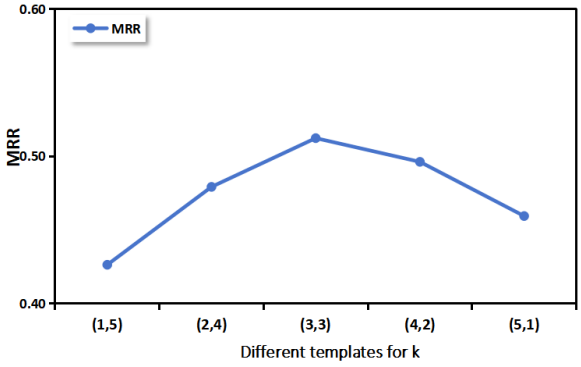

Figure 7: The relation is Atlocation. The impact of different templates on MRR when $k = 6$.

D Supplementary Experiments of Efficiency Analysis

To further verify the efficiency of our model, in addition to experiments on the commonsense KGC

Models	CN-100K				ATOMIC			
	MRR	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
ATAP$_{\text{bert-large-cased}}$	59.13	**47.54**	66.83	80.48	**15.53**	**12.76**	16.23	22.67
ATAP$_{\text{llama3.1-8b}}$	**59.45**	47.23	**67.43**	80.79	15.21	12.35	16.57	22.98
ATAP$_{\text{Qwen2-7b}}$	59.37	47.06	65.95	**81.28**	14.87	11.76	**17.31**	**23.12**

Table 6: Performance comparison on the Concept-100k and ATOMIC datasets using different language models, with the highest scores in bold.

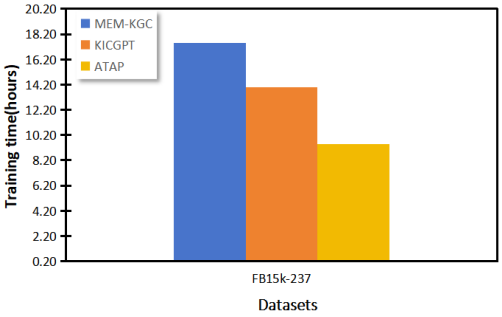

Figure 8: Comparison of training time on the traditional KGC dataset FB15k-237.

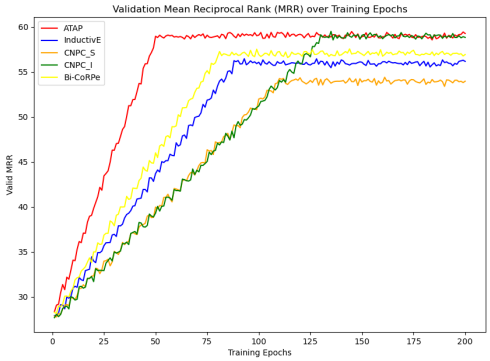

Figure 9: Comparison of convergence speed on the commonsense KGC dataset CN-100K.

datasets (as described in Section 4.5), we compare the training time of the model ATAP on the traditional KGC dataset FB15K-237. The experiments are conducted on a workstation with 4 GeForce RTX 3090 GPUs. As shown in Figure 8, the training time of ATAP is shorter than other baselines. Furthermore, to verify the convergence speed of our model, we also conduct experiments on the commonsense and traditional datasets CN-100K and FB15K-237. The experimental results in Figures 9 and 10 show that even when the number of relations is large (FB15K-237 has 237 relations), ATAP still converges the fastest.

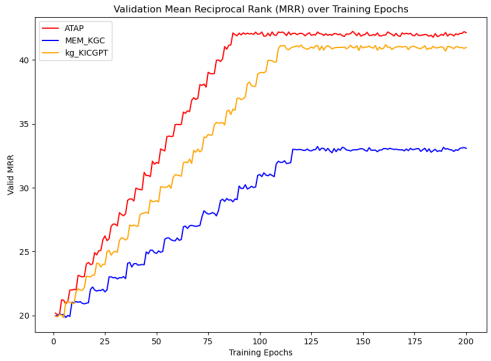

Figure 10: Comparison of convergence speed on the traditional KGC dataset FB15k-237.

E Details of ATAP on Traditional KGC experiments

As we verified in Section 4.4, ATAP alsoyifan performs better on other datasets that are not specifically used to evaluate CKGC tasks. Below we give some specific experimental details, including datasets, baseline models, and experimental settings.

Datasets. We use the following datasets for evaluation: (*i*) WN18RR is a link prediction dataset which is a subset of WordNet (Dettmers et al., 2018); (*ii*) FB15k-237 contains triples of knowledge base relations and textual mentions of Freebase entity pairs (Toutanova and Chen, 2015). Statistics of these datasets are provided in Table 7.

Baselines. We compare ATAP with the state-of-the-art traditional KGC baselines, including:

- **Triple-based methods:** RESCAL (Nickel et al., 2011), TransE (Bordes et al., 2013), DisMult (Schlichtkrull et al., 2018), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2018), TuckER (Kolda and Bader, 2009), HAKE (Zhang et al., 2020b), CompGCN (Vashishth et al., 2019), HittER (Chen et al., 2021).

Dataset	#Entities	#Relations	#Train	#Valid	#Test
FB15k-237	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134

Table 7: Statistics of FB15k-237 and WN18RR

Relation	k
_member_of_domain_usage	(6,6)
_has_part	(2,2)
_also_see	(2,2)
_hypernym	(3,3)
_synset_domain_topic_of	(5,5)
_derivationally_related_form	(6,6)
_similar_to	(2,2)
_instance_hypernym	(4,4)
_verb_group	(3,3)
_member_meronym	(4,4)
_member_of_domain_region	(5,5)

Table 8: The optimal prompt size for each relation in WN18RR.

- **Text-based methods:** Pretrain-KGE (Zhang et al., 2020c), KG-BERT (Yao et al., 2019), StAR (Wang et al., 2021b), MEM-KGC (Choi et al., 2021).

Implementation Details. We choose bert-large-cased as the base for ATAP. Prompt Encoder uses a bidirectional RNN. The choice of k value will be described in detail in Table 8 and Table 13. The rest of the parameters are the same as in Section 4.1.

F Impact of number of relations on performance

To demonstrate whether the performance varies with the number of relations, we conduct two sets of experiments.

F.1 Different numbers of relations in the same dataset

We select ATOMIC (with 9 relations) and CN-100K (with 34 relations) in CKGC, as well as FB15k-237 (with 237 relations) in traditional KGC for experiments. ATOMIC is divided into three sub-datasets ATOMIC_Rel_3, ATOMIC_Rel_6, and ATOMIC_Rel_9 according to different numbers of relations, where ATOMIC_Rel_3 means that only 3 relations are included in the training set and test set, and the relation categories in the training and test sets are the same. Similarly, CN-100K is divided into CN_Rel_5, CN_Rel_10, ..., and CN_Rel_34. FB15k-237 is divided into FB_Rel_30, ..., and FB_Rel_237.

Dataset	ATOMIC			
	MRR	Hits@1	Hits@3	Hits@10
ATOMIC_Rel_3	15.16	12.33	15.98	22.45
ATOMIC_Rel_6	15.78	12.93	16.45	23.13
ATOMIC_Rel_9	15.53	12.76	16.23	22.67

Table 9: Performance for different numbers of relations in ATOMIC.

Dataset	CN-100K			
	MRR	Hits@1	Hits@3	Hits@10
CN_Rel_5	57.34	46.73	64.89	79.61
CN_Rel_10	58.66	46.89	65.77	80.06
CN_Rel_15	59.47	47.67	66.46	80.47
CN_Rel_20	58.83	47.43	66.72	80.81
CN_Rel_25	59.31	47.61	66.70	80.21
CN_Rel_30	59.22	47.33	67.36	80.57
CN_Rel_34	59.13	47.54	66.83	80.48

Table 10: Performance for different numbers of relations in CN-100K.

Dataset	FB15k-237			
	MRR	Hits@1	Hits@3	Hits@10
FB_Rel_30	40.36	30.46	44.78	59.86
FB_Rel_60	41.52	30.98	45.24	60.08
FB_Rel_90	42.37	31.59	45.75	60.67
FB_Rel_120	42.67	31.84	45.93	60.81
FB_Rel_150	42.32	31.69	45.70	60.36
FB_Rel_180	42.29	31.67	45.66	60.64
FB_Rel_210	42.39	31.57	45.94	60.76
FB_Rel_237	42.43	31.64	45.83	60.72

Table 11: Performance for different numbers of relations in FB15k-237.

The experimental results are shown in Table 9 Table 10 and Table 11. As can be seen from these tables, ATAP's evaluation indicators first increase with the increase in the number of relations in the same dataset, and then fluctuate within a small range.

F.2 Different datasets with different numbers of relations

In addition to the CKGC datasets ATOMIC (9 relations) and CN-100K (34 relations), we also select FB15k-237 (237 relations) and WN18RR (11 relations) which are commonly used to evaluate traditional knowledge graph completion. The number of relations in FB15k-237 is 26 times that of ATOMIC.

Datasets	Performance			
	MRR	Hits@1	Hits@3	Hits@10
ATOMIC	15.53	12.76	16.23	22.67
WN18RR	57.13	46.49	62.87	71.86
CN-100K	59.13	47.54	66.83	80.48
FB15k-237	42.43	31.64	45.83	60.72

Table 12: Performance for different datasets with different numbers of relations.

The experimental results are shown in Table 12. due to different datasets, we do not get a consistent trend between the number of relations and performance, which may be affected by the number of relation annotations in each dataset and the characteristics of each dataset.

Relation	k
/soccer/football_team/current_roster./soccer/football_roster_position/position	(4,4)
/music/artist/origin	(2,2)
/ice_hockey/hockey_team/current_roster./sports/sports_team_roster/position	(5,5)
/food/food/nutrients./food/nutrition_fact/nutrient	(2,2)
/film/actor/film./film/performance/film	(3,3)
/award/award_nominee/award_nominations./award/award_nomination/nominated_for	(3,3)
/government/political_party/politicians_in_this_party./government/political_party_tenure/politician	(3,3)
/base/schemastaging/person_extra/net_worth./measurement_unit/dated_money_value/currency	(2,2)
/people/deceased_person/place_of_death	(3,3)
/people/person/profession	(4,4)
/location/administrative_division/first_level_division_of	(2,2)
/sports/sports_team/roster./sports/sports_team_roster/player	(3,3)
/base/schemastaging/person_extra/net_worth./measurement_unit/dated_money_value/currency	(2,2)
/location/county/county_seat	(5,5)
/location/location/contains	(3,3)
/tv/tv_program/program_creator	(3,3)
/music/performance_role/regular_performances./music/group_membership/group	(2,2)
/education/educational_institution/languages_spoken	(4,4)
/business/business_operation/industry	(3,3)
/biology/organism_classification/higher_classification	(5,5)

Table 13: The best optimal sizes for the top 20 relations in FB15k-237.

Dataset	#Entities	#Relations	#Train	#Valid	#Test
CN-100K	78,334	34	100,000	1,200	1,200
ATOMIC	304,388	9	610,536	87,700	87,701

Table 14: Statistics of the CKGC datasets CN-100K and ATOMIC.

Model	bert-base-cased	bert-large-cased	gpt2	gpt2-medium	gpt2-large	gpt2-xl	roberta-base	roberta-large
Vocabulary length	28996	30522	50257	34560	50257	50257	50265	50265
Training parameters	110M	340M	117M	345M	774M	1.5B	125M	355M

Table 15: Pre-trained language models vocabulary length and training parameter information.

Datasets	Prompt Template
CN-100K	IsA: [head] is a [tail]. AtLocation: [head] is located at [tail]. UsedFor: [head] is used for [tail]. CapableOf: [head] is capable of [tail]. HasProperty: [head] has the property of [tail]. HasSubevent: [head] has the subevent [tail]. HasPrerequisite: [head] has the prerequisite [tail]. Causes: [head] causes [tail]. HasA: [head] has a [tail]. PartOf: [head] is part of [tail]. MadeOf: [head] is made of [tail]. ReceivesAction: [head] receives the action [tail]. NotCapableOf: [head] is not capable of [tail]. CausesDesire: [head] causes the desire for [tail]. Desires: [head] desires [tail]. MotivatedByGoal: [head] is motivated by the goal [tail]. NotIsA: [head] is not a [tail]. HasFirstSubevent: [head] has the first subevent [tail]. NotHasProperty: [head] does not have the property of [tail]. CreatedBy: [head] is created by [tail]. DefinedAs: [head] is defined as [tail].
ATOMIC	xWant: [head] wants [tail]. oWant: Others want [tail] because of [head]. xNeed: [head] needs [tail]. xEffect: [head] causes [tail] as an effect. oEffect: [head] causes others to [tail]. xIntent: [head] does [tail] with the intention of [tail]. xReact: [head] reacts by [tail]. oReact: Others react to [head] by [tail]. xAttr: [head] is characterized as [tail].

Table 16: Manually construct input templates for each relation for CN-100K and ATOMIC.