PALT: Parameter-Lite Transfer of Language Models for Knowledge Graph Completion

Jianhao Shen\(^1\), Chenguang Wang\(^2\), Ye Yuan\(^1\), Jiawei Han\(^3\)
Heng Ji\(^3\), Koushik Sen\(^4\), Ming Zhang\(^1\), Dawn Song\(^4\)
\(^1\)Peking University, \(^2\)Washington University in St. Louis, \(^3\)University of Illinois at Urbana-Champaign, \(^4\)UC Berkeley
{jhshen,yuanye_pku,mzhang_cs}@pku.edu.cn, chenguangwang@wustl.edu, {hanj,hengji}@illinois.edu, {ksen,dawnsong}@berkeley.edu

Abstract

This paper presents a parameter-lite transfer learning approach of pretrained language models (LM) for knowledge graph (KG) completion. Instead of finetuning, which modifies all LM parameters, we only tune a few new parameters while keeping the original LM parameters fixed. We establish this via reformulating KG completion as a “fill-in-the-blank” task, and introducing a parameter-lite encoder on top of the original LMs. We show that, by tuning far fewer parameters than finetuning, LMs transfer non-trivially to most tasks and reach competitiveness with prior state-of-the-art approaches. For instance, we outperform the fully finetuning approaches on a KG completion benchmark by tuning only 1% of the parameters.\(^1\)

1 Introduction

Pretrained language models (LM) such as BERT and GPT-3 have enabled downstream transfer (Devlin et al., 2019; Brown et al., 2020). Recent studies (Petroni et al., 2019; Jiang et al., 2020; He et al., 2021) show that the implicit knowledge learned during pretraining is the key to success. Among different transfer learning techniques (Shin et al., 2020; Liu et al., 2021a,b; Houlsby et al., 2019; Devlin et al., 2019), finetuning is the de facto paradigm to adapt the knowledge to downstream NLP tasks. Knowledge graph (KG) completion is a typical knowledge-intensive application. For example, given a fact (Chaplin, profession, ___) missing an entity, it aims to predict the correct entity “screenwriter”. This task provides a natural testbed to evaluate the knowledge transfer ability of different transfer learning approaches.

Finetuning (Yao et al., 2019; Shen et al., 2022) has been recently adopted to advance the KG completion performance. However, it presents two fundamental limitations. First, finetuning is computationally inefficient, requiring updating all parameters of the pretrained LMs. This ends up with an entirely new model for each KG completion task. For example, storing a full copy of pretrained BERT\(_{\text{LARGE}}\) (340M parameters) for each task is non-trivial, not to mention the billion parameter LMs. Second, the finetuning approaches often rely on task-specific architectures for various KG completion tasks. For instance, KG-BERT (Yao et al., 2019) designs different model architectures to adapt a pretrained BERT to different tasks. This restricts its usability in more downstream tasks.

In this work, we enable parameter-lite transfer of the pretrained LMs to knowledge-intensive tasks, with a focus on KG completion. As an alternative to finetuning, our method, namely PALT, tunes no existing LM parameters. We establish this by casting the KG completion into a “fill-in-the-blank” task. This formulation enables eliciting general knowledge about KG completion from pretrained LMs. By introducing a parameter-lite encoder consisting of a few trainable parameters, we efficiently adapt the general model knowledge to downstream tasks. The parameters of the original LM network remain fixed during the adaptation process for different KG completion tasks. In contrast to finetuning which modifies all LM parameters, PALT is lightweight. Instead of designing task-specific model architectures, PALT stays with the same model architecture for all KG completion tasks that we evaluate.

The contributions are as follows:

- We propose parameter-lite transfer learning for pretrained LMs to adapt their knowledge to KG completion. The reach of the results is vital for broad knowledge-intensive NLP applications.
- We reformulate KG completion as a “fill-in-
the-blank” task. This new formulation helps trigger pretrained LMs to produce general knowledge about the downstream tasks. The new formulation implies that the KG completion can serve as a valuable knowledge benchmark for pretrained LMs, in addition to benchmarks such as LAMA (Petroni et al., 2019) and KILT (Petroni et al., 2021).

- We introduce a parameter-lite encoder to specify general model knowledge to different KG completion tasks. This encoder contains a few parameters for providing additional context and calibrating biased knowledge according to the task. The module is applicable to other deep LMs.

- We obtain state-of-the-art or competitive performance on five KG completion datasets spanning two tasks: link prediction and triplet classification. We achieve this via learning only 1% of the parameters compared to the fully finetuning approaches. In addition, compared to task-specific KG completion models, PALT reaches competitiveness with a unified architecture for all tasks.

2 PALT

We propose parameter-lite transfer learning, called PALT, as an alternative to fine-tuning for knowledge graph (KG) completion. Instead of fine-tuning which modifies all the language model (LM) parameters and stores a new copy for each task, this method is lightweight for KG completion, which keeps original LM parameters frozen, but only tunes a small number of newly added parameters. The intuition is that LMs have stored factual knowledge during the pretraining, and we need to properly elicit the relevant knowledge for downstream tasks without much modification to the original LMs. To do so, PALT first casts KG completion into a “fill-in-the-blank” task (Sec. 2.1), and then introduces a parameter-lite encoder consisting of a few trainable parameters, while parameters of the original network remain fixed (Sec. 2.2). The overall architecture of PALT is shown in Figure 1.

2.1 Knowledge Graph Completion as Fill-in-the-Blank

We reformulate KG completion as a fill-in-the-blank task. The basic idea of this task formulation is that pretrained LMs are able to answer questions formatted in cloze-style statements, and having a proper context helps to trigger LMs to produce general knowledge for the task of interest. For example, the KG completion task aims to predict the missing entity in a fact (Chaplin, profession, __), which is closely related to a cloze statement. We therefore frame the KG completion as “fill-in-the-blank” cloze statements. In this case, “Chaplin is a” provides the proper context for LMs to elicit the correct answer “screenwriter” that is generally relevant to the task.

In more detail, a fact is in the form of (head, relation, tail) or in short (h, r, t). The LM needs to predict a missing entity. A typical KG completion task provides a partial fact (h, r, __) and a set of candidate answers for the missing entity. To perform this task, at test time, we convert (h, r, t) into a cloze statement, where t indicates an answer candidate for filling the blank. For example, given a partial fact (Chaplin, profession, __), an LM needs to fill in the blank of the cloze statement “Charlie is a __” by providing it as the model input. In our case, a candidate answer (Chaplin, profession, screenwriter) is given (e.g., “screenwriter” is one of the candidates), the corresponding cloze statement will turn into “[CLS] Chaplin is a [SEP] screenwriter [SEP]” (Figure 1). We use this statement as an input to a pretrained LM. [CLS] and [SEP] are special tokens of the pretrained LMs, e.g., BERT. “Chaplin” is the head entity name or description. “is a” is relation name or description. “screenwriter” is the candidate tail entity name or
description. Sec 3.1 includes resources for obtaining the entity or relation descriptions.

2.2 Parameter-Lite Encoder

While the new formulation helps pretrained LMs to provide general knowledge about the tasks, downstream tasks often rely on task-specific or domain-specific knowledge. To adapt the general knowledge in pretrained LMs to various KG completion tasks, we introduce a parameter-lite encoder including two groups of parameters: (i) a prompt encoder serving as the additional task-specific context in the cloze statement, and (ii) contextual calibration encoders aiming to mitigate model’s bias towards general answers. The encoder is added on top of the original LM network whose parameters remain frozen during tuning.

Knowledge Prompt Encoder  Beyond general context from the task formulation, we believe that task-specific context helps better recall the knowledge of interest in pretrained LMs. For example, if we want the LM to produce the correct answer “screenwriter” for “Charlie is a __”, a task-specific prefix such as “profession” in the context will help. The LM will then assign a higher probability to “screenwriter” as the correct answer. In other words, we want to find a task-specific context that better steers the LM to produce task-specific predictions. Intuitively, the task-specific tokens influence the encoding of the context, thus impacting the answer predictions. However, it is non-trivial to find such task-specific tokens. For example, manually writing these tokens is not only time consuming, but also unclear whether it is optimal for our task. Therefore, we design a learnable and continuous prompt encoder.

Specifically, we use “virtual” prompt tokens as continuous word embeddings. As shown in Figure 1, we append these prompt tokens to different positions in the context. The embeddings of prompt tokens are randomly initialized and are updated during training. To allow more flexibility in context learning, we add a linear layer with a skip connection on top of the embedding layer to project the original token embeddings to another subspace. This projection enables learning a more tailored task-specific context that better aligns with LM’s knowledge. The knowledge prompt encoder is defined in Eq. 1.

\[ e'_i = W_p e_i + b_p + e_i \]  (1)

where \( e'_i \) denotes the virtual token embedding, and \( e_i \) denotes the input token embedding. \( W_p \) and \( b_p \) are the tunable weight and bias of the prompt encoder. The knowledge prompt encoder provides task-specific context for KG completion as it is tuned on task-specific training data.

Knowledge Calibration Encoder  Another main pitfall of pretrained LMs is that they tend to be biased towards common answers in their pretraining corpus. For example, the model prefers “United States” over “Georgia” for the birth place of a person, which is suboptimal for KG completion. We actually view this as a shift between the pretraining distribution and the distribution of downstream tasks.

We counteract such biases by calibrating the output distribution of pretrained LMs. Concretely, we introduce task-specific calibration parameters between Transformer layers of LMs (Figure 1) to gradually align the pretraining distribution with the downstream distribution. We choose a linear encoder with a skip connection to capture the distribution shifts, as shown in Eq. 2.

\[ h'_i = W_c h_i + b_c + h_i \]  (2)

where \( h'_i \) is the calibrated hidden state, and \( h_i \) is the hidden state of a Transformer layer. \( W_c \) and \( b_c \) are the tunable weight and bias of the knowledge calibration encoder.

Training and Inference  We keep all LM parameters fixed and only tune the parameters in the parameter-lite encoder. After formatting the KG completion tasks following our formulation, a candidate fact is in the standard sentence pair format of BERT. For example, the candidate (Chaplin, profession, screenwriter) is formulated as “[CLS] Chaplin is a [SEP] screenwriter [SEP]”. “Chaplin is a” is the first sentence as the cloze-style question, while the second sentence is “screenwriter” implying an answer candidate. LM then decides whether the second sentence is a correct answer to the question or not. This naturally aligns with the next sentence prediction (NSP) task of BERT, which outputs a positive label if the answer is correct; otherwise negative. Therefore, we directly utilize the next sentence prediction to perform KG completion thanks to our formulation.

The training objective is to decide whether the second sentence is the correct next sentence to the first sentence. The small number of tunable param-
eters are then updated with respect to the objective. To optimize those parameters, we need both positive and negative examples. We use negative sampling (Mikolov et al., 2013) for efficiency consideration. To be more specific, for a positive fact \((h, r, t)\), we first corrupt its head entity with \(n_{\text{ns}}\) random sampled entities to form negative facts, e.g., \((\tilde{h}_i, r, t)\). If a sampled fact is in the KG, it should be considered positive so we will re-sample it. The loss function for the head entity is defined in Eq. 3.

\[
L_h = -\log \Pr(1|h, r, t) - \sum_{i} \mathbb{E}_{\tilde{h}_i \sim E \setminus \{h\}} \log \Pr(0|\tilde{h}_i, r, t) \tag{3}
\]

where \(\Pr(\cdot|h, r, t)\) is the output probability of the BERT NSP classifier.

For each fact, the losses for its relation \(L_r\) and tail entity \(L_t\) are similarly defined. There are \(3 * n_{\text{ns}}\) negative facts in total for each fact. Similar to the negative facts for its head entity (e.g., \((\tilde{h}_i, r, t)\)), we have the negative facts for its relation (e.g., \((h, \tilde{r}_i, t)\)), and its tail entity (e.g., \((h, r, \tilde{t}_i)\)) respectively. The joint loss function is the sum of the three components as defined in Eq. 4.

\[
L = \sum_{(h, r, t) \in G} (L_h + L_r + L_t) \tag{4}
\]

where \(G\) is a collection of all KG facts.

3 Experiments

In this section, we evaluate the parameter-lite transfer ability of PALT on two KG completion tasks: triplet classification and link prediction. The details of the experimental setup, datasets, and comparison methods are described in Appendix A.

3.1 Experimental Setup

Datasets We conduct the experiments on five datasets: WN11 (Socher et al., 2013) and FB13 (Socher et al., 2013) for triplet classification, and FB15k-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018) and UMLS (Dettmers et al., 2018) for link prediction. A detailed description for these datasets is in Appendix A.3. Table 1 summarizes the statistics of the datasets.

Comparison Methods We compare PALT with the following KG completion models. (i) task-specific models (designed for KG completion):

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Entity</th>
<th># Relation</th>
<th># Train</th>
<th># Dev</th>
<th># Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15k-237</td>
<td>14,541</td>
<td>237</td>
<td>272,115</td>
<td>17,535</td>
<td>20,466</td>
</tr>
<tr>
<td>WN18RR</td>
<td>40,933</td>
<td>11</td>
<td>86,835</td>
<td>3,034</td>
<td>3,134</td>
</tr>
<tr>
<td>UMLS</td>
<td>1,135</td>
<td>46</td>
<td>5,216</td>
<td>552</td>
<td>661</td>
</tr>
<tr>
<td>FB13</td>
<td>75,043</td>
<td>13</td>
<td>116,232</td>
<td>5,908</td>
<td>23,731</td>
</tr>
<tr>
<td>WN11</td>
<td>58,096</td>
<td>11</td>
<td>112,381</td>
<td>2,609</td>
<td>10,544</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics.

TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015), TransD (Ji et al., 2015), DistMult (Yang et al., 2015), TransG (Xiao et al., 2016), TranSparse (Ji et al., 2016), ComplEx (Trouillon et al., 2016), R-GCN (Schlichtkrull et al., 2018), ConvE (Dettmers et al., 2018), ConvKB (Nguyen et al., 2018), DistMult-HRS (Zhang et al., 2018), RotatE (Sun et al., 2019), REFE (Chami et al., 2020), HAKE (Zhang et al., 2020b), and ComplEx-DURA (Zhang et al., 2020a), NTN (Socher et al., 2013), DOLORES (Wang et al., 2020b), KBGAT (Nathani et al., 2019), and GAATs (Wang et al., 2020c), TEKE (Wang and Li, 2016), stAR (Wang et al., 2021a); and (ii) a general model KG-BERT (Yao et al., 2019). It utilizes finetuning to transfer LMs for KG completion, and is task agnostic.

3.2 Main Results

Triplet Classification Triplet classification is a binary classification task to predict whether a given fact \((h, r, t)\) is correct or not. For each fact, we prepare the input following Sec. 2.1 (Figure 1) and feed the input into the model. The prediction score is the output probability of the NSP classifier. If the score is above a threshold, the fact is predicted as positive, otherwise negative. We tune the threshold on dev sets and report the accuracy on test data. The results are summarized in Table 2.

PALT\textit{BASE} outperforms all task-specific models, and achieves competitive or better performance compared to the finetuning method KG-BERT. PALT\textit{LARGE} gains further improvement. PALT\textit{BASE} outperforms the best task-specific model by 4.4% on WN11 and 1.3% on FB13. It outperforms the finetuning method by 0.4% on average (with a 0.9% improvement on FB13). It is slightly worse than the finetuning model on WN11. Compared to finetuning, PALT\textit{LARGE} gains 0.3% and 1.3% improvements on WN11 and FB13 respectively. These results suggest that PALT is able to transfer knowledge in pretrained LMs for KG completion tasks. Importantly, it is able to outper-
Table 2: Triplet classification accuracy. Task-specific models are designed for knowledge graph completion, while general models are task agnostic.

<table>
<thead>
<tr>
<th>Method</th>
<th>WN11</th>
<th>FB13</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task-specific models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NTN (Socher et al., 2013)</td>
<td>86.2</td>
<td>90.0</td>
<td>88.1</td>
</tr>
<tr>
<td>TransE (Bordes et al., 2013)</td>
<td>75.9</td>
<td>81.5</td>
<td>78.7</td>
</tr>
<tr>
<td>TransH (Wang et al., 2014)</td>
<td>78.8</td>
<td>83.3</td>
<td>81.1</td>
</tr>
<tr>
<td>TransR (Lin et al., 2015)</td>
<td>85.9</td>
<td>82.5</td>
<td>84.2</td>
</tr>
<tr>
<td>TransD (Ji et al., 2015)</td>
<td>86.4</td>
<td>89.1</td>
<td>87.8</td>
</tr>
<tr>
<td>TEKE (Wang and Li, 2016)</td>
<td>86.1</td>
<td>84.2</td>
<td>85.2</td>
</tr>
<tr>
<td>TransG (Xiao et al., 2016)</td>
<td>87.4</td>
<td>87.3</td>
<td>87.4</td>
</tr>
<tr>
<td>TransSparse-S (Ji et al., 2016)</td>
<td>86.4</td>
<td>88.2</td>
<td>87.3</td>
</tr>
<tr>
<td>DistMult (Yang et al., 2015)</td>
<td>87.1</td>
<td>86.2</td>
<td>86.7</td>
</tr>
<tr>
<td>DistMult-HRS (Zhang et al., 2020a)</td>
<td>88.9</td>
<td>89.0</td>
<td>89.0</td>
</tr>
<tr>
<td>AATE (An et al., 2018)</td>
<td>88.0</td>
<td>87.2</td>
<td>87.6</td>
</tr>
<tr>
<td>ConvKB (Nguyen et al., 2018)</td>
<td>87.6</td>
<td>88.8</td>
<td>88.2</td>
</tr>
<tr>
<td>DOLORES (Wang et al., 2020b)</td>
<td>87.5</td>
<td>89.3</td>
<td>88.4</td>
</tr>
<tr>
<td><strong>General models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KG-BERT (Yao et al., 2019)</td>
<td>93.5</td>
<td>90.4</td>
<td>91.9</td>
</tr>
<tr>
<td>PALT_BASE (ours)</td>
<td>95.5</td>
<td>91.5</td>
<td>93.5</td>
</tr>
<tr>
<td>PALT_LARGE (ours)</td>
<td>93.8</td>
<td>91.7</td>
<td>92.8</td>
</tr>
</tbody>
</table>

To better understand PALT, we further conduct an ablation study on WN11 to show the effectiveness of different components. Specifically, we evaluate PALT_BASE without knowledge prompt encoder (denoted as “w/o Prompt”) or knowledge calibration encoder (denoted as “w/o Calibration”). We also remove the entire parameter-lite encoder (denoted as “w/o Encoder”). Note this will make PALT a zero-shot model since there are no tunable parameters. For comparison, we also test BERT_BASE under the finetuning setting, where we do not add any new parameters and directly finetune BERT for triplet classification with our formulation. The results are shown in Table 4.

We have the following observations: (i) all components have a positive effect on the final performance. The knowledge prompt encoder brings the most improvement which is 1.6%. The knowledge calibration encoder at the middle layer brings a 1.1% improvement, and that at the last layer brings a 0.3% improvement. The results indicate that it is more important to recall and prepare the knowledge in earlier layers for the task of interest. (ii) Removing both knowledge calibration encoders results in the worst accuracy. The knowledge calibration encoders are important for knowledge transfer. (iii) PALT_BASE outperforms all parameters, which suggests that PALT is an effective way to adapt pretrained LMs for KG completion since it requires far less computation and storage. (iv) Furthermore, we can see that without the entire parameter-lite encoder, our model still achieves promising results. On WN11, it achieves 73.7% accuracy, which is approximately 1.5 times the accuracy of random guesses (50%). This shows the effectiveness of our task formulation. Formulating KG completion as a “fill-in-the-blank” task triggers the knowledge that an LM learned during pretraining. This enables an efficient transfer algorithm.
<table>
<thead>
<tr>
<th>Method</th>
<th>WN11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PALT BASE</td>
<td>93.3</td>
</tr>
<tr>
<td>w/o Prompt</td>
<td>91.7</td>
</tr>
<tr>
<td>w/o Calibration_{middle}</td>
<td>92.2</td>
</tr>
<tr>
<td>w/o Calibration_{last}</td>
<td>93.0</td>
</tr>
<tr>
<td>w/o Calibration_{both}</td>
<td>89.3</td>
</tr>
<tr>
<td>w/o Encoder</td>
<td>73.7</td>
</tr>
<tr>
<td>Finetuning</td>
<td>93.2</td>
</tr>
</tbody>
</table>

Table 4: Ablation study on WN11. We remove knowledge prompt encoder, or knowledge calibration encoder, or the entire parameter-lite encoder.

3.4 Parameter Efficiency Analysis

The advantage of PALT is that only a small amount of newly added parameters are tuned while all LM parameters are fixed. This brings two benefits: space-efficient model storage and efficient computation. Here we compare the numbers of tunable parameters of PALT and BERT, and the detailed calculation of PALT is presented in Appendix B. BERT BASE has 110M tunable parameters, while PALT BASE has 1.77M. PALT LARGE has 3.15M tunable parameters, while BERT LARGE has 340M. PALT BASE only has 1.6% tunable parameters of BERT BASE, and PALT LARGE has 0.9% of BERT LARGE. We show the numbers of tunable parameters of different models in Figure 2.

3.5 Case Study

In this section, we perform a case study analysis to illustrate why PALT performs well. We use BertViz (Vig, 2019) to visualize the attention weights of PALT. We take an example of a positive fact \((h, r, t)\): \(h = \text{“evening clothes”}, r = \text{“type of”} \) and \(t = \text{“attire”}\) and show the attention weights of the first, the middle and the last attention layers in Figure 3. In the first layer, the prompt token attends to all tokens, indicating that it helps to recall general knowledge. In the middle layer, the attention weights concentrate on the most relevant parts of the tokens. Specifically, the attention weight between “type” and “clothes” is large. The “type” token also pays attention to [SEP] token. It is mainly because the [SEP] token marks the boundary of two sentences and the pretrained LM uses it as an aggregation representation of each sentence.
In the last layer, different heads of [CLS] focus on different parts of the text. For example, the first head (in blue) attends to the tail entity. The seventh head (in green) attends to prompt tokens. This shows that [CLS] gathers task-specific knowledge for the NSP classifier.

In Table 5, we give some examples of FB13 that are improved by PALT compared to the finetuning approach (i.e., facts that are correctly predicted by PALT while KG-BERT fails). We further show the comparison between attention weights of PALT and the finetuning approach in Appendix C.

3.6 Error Analysis

We analyze the errors made by PALT in this section. Here we focus on analyzing relations with the highest and lowest error rates. The detailed error rate statistics are shown in Appendix D. Most of PALT errors are due to “domain” relations, with an error rate of 14.7% for the relation “domain topic” and 10.5% for “domain region”. The reason is that we find the relations of the “domain” are not well defined, and the boundary between relations can be unclear. For the relation “subordinate instance of”, PALT performs the best with an error rate of 2.6%, since it is more related to semantic information. We further analyze the attention weights of some error cases in Figure 4. For the first case, the [CLS] token attends mainly to the head and relation tokens but little to the tail entity. This is because “barbiturate” is a rare entity and the LM does not capture much knowledge for it during pre-training. PALT fails on the second case mainly because “domain topic” covers a wide range of concepts. This results in a uniform distribution of attention so it is difficult to make a correct prediction. For the last case, [CLS] attends to both the head and tail entities but little to the relations. This leads to the error. We notice that most entities are segmented into sub-words based on BERT’s tokenizer. This may result in a poor understanding of entities. We believe other pretraining paradigms like span-masking (Joshi et al., 2020) will help and leave it as future work.

4 Related Work

Pretrained LMs (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Wang et al., 2022) have achieved state-of-the-art results in many NLP tasks (Wang et al., 2018, 2019). Some work also uses pretrained LMs for knowledge-driven tasks (Yao et al., 2019; Omeliyanenko et al., 2020; Aspillaga et al., 2021) by finetuning them on downstream tasks, which has been the de facto method to achieve superior results. However, finetuning modifies all parameters, which requires a large amount of computation and storage resources. Recently, prompt-tuning (Shin et al., 2020; Liu et al., 2021b), adaptors (Houlsby et al., 2019; Wang et al., 2021c; Newman et al., 2022), and factual probing (He et al., 2021; Petroni et al., 2019; Wang et al., 2020a, 2021b) are developed to transfer pretrained LMs to downstream tasks and show improvements on many NLP tasks. Gao et al. (2021) focus on prompt based finetuning, while Jiang et al. (2020) follow standard prompt formulation for a single token.

Compared to the existing methods, there are several distinctive features of PALT. First, instead of using the output of a single [MASK] token, we leverage the next sentence prediction, which allows the answers with arbitrary lengths. Second, we automatically acquire the template using the natural language descriptions of the relations available in the downstream KGs. Besides, we also use the corresponding entity descriptions in the cloze statement, providing richer context. Third, our method differs from prompt-tuning, which only inserts virtual tokens into the input. Fourth, in contrast to the empirical calibration procedures that are highly customized for each task, our method automatically learns a few calibration parameters for each task. Overall, compared to previous methods, our approach is lightweight. The parameter-lite encoder is unique and particularly useful for more NLP tasks.
Figure 3: Visualization of attention weights of different Transformer layers of PALT. The 0th layer is the first attention layer. The 7th layer is the attention layer after our middle calibration encoder. The 11th layer is the last attention layer. Different color represents different attention heads. The darker the color is, the larger the attention score.

Figure 4: The attention weights of the last layer of PALT on three error cases involving different relations.

5 Conclusion

We propose PALT, a parameter-lite transfer of pre-trained language models (LM) for knowledge graph completion. To efficiently elicit general knowledge of LMs learned about the task during pretraining, we reformulate KG completion as a “fill-in-the-blank” task. We then develop a parameter-lite encoder including two groups of parameters. First, it contains a knowledge prompt encoder consisting of learnable continuous prompt tokens to better recall task-specific knowledge from pretrained LMs. Second, it calibrates pretrained LMs representations and outputs for KG completion via two knowledge calibration encoders. As a result, our method achieves competitive or even better results than finetuning with far fewer tunable parameters. Both the task formulation and parameter-lite encoder can be inspiring for a wide range of knowledge-intensive tasks and deep LMs. We hope this research can foster future research along the parameter-lite knowledge transfer direction in NLP.
6 Limitations

As for the limitations of our method, the input is constructed based on the natural language descriptions of the entities and relations, and such descriptions may need additional efforts to obtain in different application scenarios. Although our method achieves competitive results in the medical domain (UMLS), the main finding of our study is that our method is more capable of transferring general knowledge in LMs to KG completion tasks. We welcome more studies on strengthening its performance in specific domains, e.g., using domain-specific LMs for a particular domain (e.g., BioBERT (Lee et al., 2020) for the medical domain). Finally, our method shares some common limitations with most deep learning approaches. For example, the decisions are not easy to interpret, and the predictions can retain the biases of the training data.

7 Ethical Considerations

We hereby acknowledge that all of the co-authors of this work are aware of the provided ACM Code of Ethics and honor the code of conduct. The followings give the aspects of both our ethical considerations and our potential impacts to the community. This work uses pretrained LMs for KG completion. We develop an encoder especially the knowledge calibration encoder to mitigate the potential knowledge biases in pretrained LMs. The risks and potential misuse of pretrained LMs are discussed in (Brown et al., 2020). There are potential undesirable biases in the datasets, such as unfaithful descriptions from Wikipedia. We do not anticipate the production of harmful outputs after using our model, especially towards vulnerable populations.

8 Environmental Considerations

We build PALT based on pretrained BERT\textsubscript{BASE} and BERT\textsubscript{LARGE}. According to the estimation in (Strubell et al., 2019), pretraining a base model costs 1,507 kWh·PUE and emits 1,438 lb CO\textsubscript{2}, while pretraining a large model requires 4 times the resources of a base model. Our methods only tune 1% parameters with fewer than 1% gradient-steps of the number of steps of pretraining. Therefore, our energy cost and CO\textsubscript{2} emissions are relatively small.

Acknowledgements

We would like to thank the anonymous reviewers for their suggestions and comments. This paper is partially supported by National Key Research and Development Program of China with Grant No. 2018AAA0101902 and the National Natural Science Foundation of China (NSFC Grant Numbers 62106008 and 62276002). This material is in part based upon work supported by Berkeley DeepDrive and Berkeley Artificial Intelligence Research. The research was also supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS-17-04532.

References


Tianxing He, Kyunghyun Cho, and James Glass. 2021. An empirical study on few-shot knowledge probing for pretrained language models. CoRR.


Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhihlin Yang, and Jie Tang. 2021b. GPT Understands, Too. CoRR.


Benjamin Newman, Prafulla Kumar Choube, and Nazneen Rajani. 2022. P-adapters: Robustly extracting factual information from language models with diverse prompts. In ICLR.


Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by rotational relation in complex space. In ICLR.


Chenguang Wang, Xiao Liu, Zai Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2021b. Zero-shot information extraction as a unified text-to-triple translation. In EMNLP.


Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In ICLR.

Li Yang, Chengsheng Mao, and Yuan Luo. 2019. Knowledge graph completion. CoRR.
A Experimental Setup Details

We describe additional details of our experimental setup including implementation, datasets and comparison methods in this section.

A.1 Implementation Details

We implement our algorithm using the Hugging Face Transformers package. We optimize PALT with AdamW (Loshchilov and Hutter, 2019). The hyper-parameters are set as follows. We use 8 GPUs and set the batch size to 32 per GPU, and set the learning rate to \([1.5 \times 10^{-4}, 1.5 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-5}]\) for WN11, FB13, FB15k-237, WN18RR, respectively. We set the warm-up ratio to 0.1 and set weight decay as 0.01. The number of training epochs is 10 for link prediction and 40 for triplet classification. For link prediction, we sample 5 negative samples for the head entity, relation and tail entity, resulting in 15 negative triplets in total for each sample. And for triplet classification, we only sample one negative sample for each entity. Note that the negative samples here are used for training (Eq. 4), which are different from the candidate sets for link prediction during evaluation. We adopt grid search to tune the hyper-parameters on the dev set. For learning rates, we search from 1e-5 to 5e-4 with an interval of 5e-6. For the number of negative examples, we test values in \{1, 5, 10\}. For the remaining hyper-parameters, we generally follow BERT’s setup.

For model inputs, we use synset definitions as entity descriptions for WN18RR, and descriptions produced by Xie et al. (2016) for FB15k-237. For FB13, we use entity descriptions in Wikipedia. We use entity names for WN11 and UMLS. For all datasets, we use relation names as relation descriptions.

For PALT architecture, we insert two knowledge calibration encoders to the middle layer and last layer of BERT. This applies to both PALT_{BASE} and PALT_{LARGE}. For knowledge prompt encoder, we add it to the input layer. In particular, 10 prompt tokens are added at 3 different positions for PALT_{BASE} on all datasets except for WN11. 2 prompt tokens are added at the beginning of WN11. This is because the entity description of WN11 is short. For PALT_{LARGE}, we add 2 prompt tokens at the beginning.

A.2 Datasets

We introduce the link prediction and triplet classification datasets below.

A.2.1 Link Prediction

- **FB15k-237.** Freebase is a large collaborative KG consisting of data composed mainly by its community members. It is an online collection of structured data harvested from many sources, including individual and user-submitted wiki contributions (Pellissier Tanon et al., 2016). FB15k is a selected subset of Freebase that consists of 14,951 entities and 1,345 relationships (Bordes et al., 2013). FB15K-237 is a variant of FB15K where inverse relations and redundant relations are removed, resulting in 237 relations (Toutanova et al., 2015).

- **WN18RR.** WordNet is a lexical database of semantic relations between words in English. WN18 (Bordes et al., 2013) is a subset of WordNet which consists of 14,951 entities and 1,345 relationships (Bordes et al., 2013). WN18RR is created to ensure that the evaluation dataset does not have inverse relations to prevent test leakage (Dettmers et al., 2018).

- **UMLS.** UMLS semantic network (McCray, 2003) is an upper-level ontology of the Unified Medical Language System. The semantic network, through its 135 semantic types, provides a consistent categorization of all concepts represented in the UMLS. The 46 links between the semantic types provide the structure for the network and represent important relationships in the biomedical domain.

A.2.2 Triplet Classification

- **WN11 and FB13** are subsets of WordNet and Freebase respectively for triplet classification, where the Socher et al. (2013) randomly switch entities from correct testing triplets resulting in a total of doubling the number of test triplets.
with an equal number of positive and negative examples.

A.3 Comparison Methods

We compare PALT to three types of KG completion methods: shallow structure embedding, deep structure embedding, and language semantic embedding.\(^2\)

A.3.1 Shallow Structure Embedding

TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015), TransD (Ji et al., 2015), TransG (Xiao et al., 2016), TransSparseS (Ji et al., 2016), DistMult (Yang et al., 2015), ConvKB (Nguyen et al., 2018), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), RotatE (Sun et al., 2019), REFE (Chami et al., 2020), HAKE (Zhang et al., 2020b), and ComplEx-DURA (Zhang et al., 2020a) are methods based only on the structure of the KG. DistMult-HRS (Zhang et al., 2018) is an extension of DistMult which is combined with a three-layer hierarchical relation structure (HRS) loss. Each of these methods proposes a scoring function regarding a knowledge fact, without using the natural language descriptions or names of entities or relations. The scoring functions are shown in Table 6.

A.3.2 Deep Structure Embedding

- **NTN** (Neural Tensor Network) (Socher et al., 2013) models entities across multiple dimensions by a bilinear tensor neural layer.
- **DOLORES** (Wang et al., 2020b) is based on bi-directional LSTMs and learns deep representations of entities and relations from constructed entity-relation chains.
- **KBGAT** proposes an attention-based feature embedding that captures both entity and relation features in any given entity’s neighborhood, and additionally encapsulates relation clusters and multi-hop relations (Nathani et al., 2019).
- **GAATs** integrates an attenuated attention mechanism in a graph neural network to assign different weights in different relation paths and acquire the information from the neighborhoods (Wang et al., 2020c).

A.3.3 Language Semantic Embedding

- **TEKE** (Wang and Li, 2016) takes advantage of the context information in a text corpus. The textual context information is incorporated to expand the semantic structure of the KG and each relation is enabled to own different representations for different head and tail entities.
- **AATE** (An et al., 2018) is a text-enhanced KG representation learning method, which can represent a relation/entity with different representations in different facts by exploiting additional textual information.
- **KG-BERT** (Yao et al., 2019) considers facts in KG as textual sequences, where each textual sequence is a concatenation of text descriptions of the head entity, the relation, and the tail entity. Then KG-BERT treats the KG completion task as a text binary classification task, and then solves it by fine-tuning a pre-trained BERT.
- **StAR** (Wang et al., 2021a) partitions each fact into two asymmetric parts as in translation-based graph embedding approach, and encodes both parts into contextualized representations by a Siamese-style textual encoder (BERT or RoBERTa) (Wang et al., 2021a).

B Number of Tunable Parameters

Here we give a detailed calculation of the number of tunable parameters of PALT. We denote \(d_e\) as the dimension of token embeddings and \(d_h\) as the hidden size in the pretrained LM, and \(n_p\) as the number of prompts added in PALT. The number of tunable parameters for the prompt embeddings and linear mapping weights is \(n_p \times d_e + d_e \times d_h\), and that of knowledge calibration encoder is \(2 \times d_h \times d_h\). In total, there are \(n_p \times d_e + d_e \times d_h + 2 \times d_h \times d_h\) tunable parameters in PALT. For BERT\(_{BASE}\), \(d_e = d_h = 768\), and for BERT\(_{LARGE}\), \(d_e = d_h = 1024\), and we use \(n_p = 2\) on WN11. As a result, PALT\(_{BASE}\) has 1.77M tunable parameters, and PALT\(_{LARGE}\) has 3.15M.

C Case Study

To illustrate the difference between PALT and KG-BERT, we show the attention weights of [CLS] in the last layer in Figure 6. We can see that for KG-BERT, most attention heads attend to the whole sequence, while for PALT each head attends to a specific part of the sequence. For example, the third

\(^2\)We refer the readers to (Ji et al., 2022) for a more comprehensive review of the KG completion methods.
Method | Score Function | $h, r, t \in \mathbb{R}^k$
---|---|---
TransE | $-\|h + r - t\|$ | $h, r, t \in \mathbb{R}^k$
TransH | $\|M_h h + r - M_t t\|$ | $h, t \in \mathbb{R}^k, M_h, M_t \in \mathbb{R}^{k \times d}$
TransR | $\|\sum_{i} \exp \left(-\frac{|\langle h, w_i \rangle + r - \langle t, w_i \rangle |}{\sigma^2}\right)\|$ | $h \sim \mathcal{N}(\mu_h, \sigma^2_h), t \sim \mathcal{N}(\mu_t, \sigma^2_t), r \sim \mathcal{N}(\mu_r, \sigma^2_r), \mu_h, \mu_t, \mu_r \in \mathbb{R}^k$
TransD | $\|M_r h + r - M_t t\|$ | $h, t, w, r, w_r \in \mathbb{R}^k$
TransG | $\|h + r - t\|$ | $h, r, t \in \mathbb{R}^k$

Table 6: The score functions $f_{\theta}(h, t)$ of shallow structure embedding models for KG embedding, where $(\langle \cdot \rangle)$ denotes the generalized dot product, $\circ$ denotes the Hadamard product, $\sigma$ denotes activation function and $*$ denotes 2D convolution. $\odot$ denotes conjugate for complex vectors, and 2D reshaping for real vectors in the ConvE model. $Ref(\theta)$ denotes the reflection matrix induced by rotation parameters $\theta$. $\odot^c$ is Möbius addition that provides an analogue to Euclidean addition for hyperbolic space.

(coloed green) head has large weights on prompts and the tail entity, and the fourth (colored red) head pays attention to the head entity and relation. This demonstrates that PALT recalls and calibrates related knowledge in a more disentangled way than KG-BERT, and as a result, it succeeds to predict this triplet as negative.

In Table 7 we demonstrate some triplets of WN11 that are correctly predicted by PALT while KG-BERT fails.

| Tails | Label | Errors Rate(%)
|---|---|---
| domain topic | 14.7 |
| domain region | 10.5 |
| has instance | 8.3 |
| member meronym | 8.1 |
| synset domain topic | 7.4 |
| similar to | 7.1 |
| has part | 7.0 |
| part of | 5.7 |
| type of | 5.3 |
| member holonym | 3.9 |
| subordinate instance of | 2.6 |

Table 8: The error rates of triplet classification on different relations.

different positions in order. For example, “2-0-0” means we add 2 prompt tokens before the head entity and no prompt tokens after the relation and after the tail entity. The results are shown in Figure 7. We observe that “2-0-0” performs better than “0-0-0”, and the difference between token numbers and positions is marginal, meaning that what matters is whether to add prompt tokens or not, and numbers and positions are not very important.

F Effectiveness of Calibration

In this section, we show the effectiveness of knowledge calibration encoder. We show the two layers of attention weights of the original BERT and our calibrated PALT in Figure 5. The left two are attention weights in the middle layer and the right two are in the last layer. For the original BERT, the
Figure 5: Attention weights of the original BERT and PALT.

Figure 6: Comparison between the attention weights of PALT and KG-BERT. In this example, PALT correctly predicts it as negative and KG-BERT fails.

Figure 7: Accuracy for different numbers and positions of prompt tokens on WN11.

The attention weight in the middle layer between “type” and “clothes” is small, but it is larger for the PALT. And in the last layer, the attention weights of the original BERT between “[CLS]” and “clothes” and “type” are smaller than those of PALT. These indicate that our knowledge calibration encoder helps to calibrate pretrained LMs for KG completion.