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 $\text{BERT}_{\text{LARGE}}$ (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-

†Corresponding authors.
[1]The code and datasets are available at https://github.com/yuanyehome/PALT.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3833–3847
December 7-11, 2022

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 finetuning for knowledge graph (KG) completion. Instead of finetuning 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

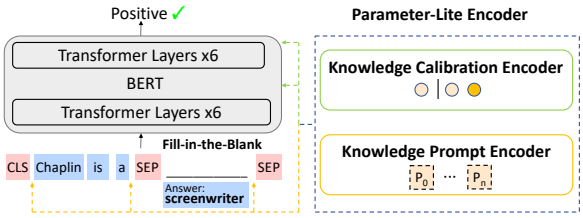


Figure 1: Summary of our approach PALT. Compared to finetuning, PALT is a parameter-lite alternative to transfer the knowledge that pretrained language models know about knowledge graph completion. Our approach first casts knowledge graph completion into a fill-in-the-blank task. This formulation enables pretrained language models to produce general knowledge for knowledge graph completion. By introducing a few trainable parameters via a parameter-lite encoder (in the dashed box), PALT further adapts the general knowledge in language models to different knowledge graph completion tasks without modifying the original language model parameters (in grey).

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.

$$e'_i = \mathbf{W}_p e_i + b_p + e_i \quad (1)$$

where e'_i denotes the virtual token embedding, and e_i denotes the input token embedding. $\mathbf{W}_p$ 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.

$$h'_i = \mathbf{W}_c h_i + b_c + h_i \quad (2)$$

where h'_i is the calibrated hidden state, and h_i is the hidden state of a Transformer layer. $\mathbf{W}_c$ 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_{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.

$$L_h = -\log \Pr(1|h,r,t) - \sum_{i}^{n_{\text{ns}}} \mathbb{E}_{\tilde{h}_i \sim E \setminus \{h\}} \log \Pr(0|\tilde{h}_i,r,t) \quad (3)$$

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.

$$L = \sum_{(h,r,t) \in G} (L_h + L_r + L_t) \quad (4)$$

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.

Dataset	# Entity	# Relation	# Train	# Dev	# Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
UMLS	135	46	5,216	652	661
FB13	75,043	13	316,232	5,908	23,733
WN11	38,696	11	112,581	2,609	10,544

Table 1: Dataset statistics.

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

$\text{PALT}_{\text{BASE}}$ outperforms all task-specific models, and achieves competitive or better performance compared to the finetuning method KG-BERT. $\text{PALT}_{\text{LARGE}}$ gains further improvement. $\text{PALT}_{\text{BASE}}$ 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, $\text{PALT}_{\text{LARGE}}$ 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-

Method	WN11	FB13	Avg
Task-specific models			
NTN (Socher et al., 2013)	86.2	90.0	88.1
TransE (Bordes et al., 2013)	75.9	81.5	78.7
TransH (Wang et al., 2014)	78.8	83.3	81.1
TransR (Lin et al., 2015)	85.9	82.5	84.2
TransD (Ji et al., 2015)	86.4	89.1	87.8
TEKE (Wang and Li, 2016)	86.1	84.2	85.2
TransG (Xiao et al., 2016)	87.4	87.3	87.4
TranSparse-S (Ji et al., 2016)	86.4	88.2	87.3
DistMult (Yang et al., 2015)	87.1	86.2	86.7
DistMult-HRS (Zhang et al., 2020a)	88.9	89.0	89.0
AATE (An et al., 2018)	88.0	87.2	87.6
ConvKB (Nguyen et al., 2018)	87.6	88.8	88.2
DOLORES (Wang et al., 2020b)	87.5	89.3	88.4
General models			
KG-BERT (Yao et al., 2019)	93.5	90.4	91.9
$\text{PALT}_{\text{BASE}}$ (ours)	93.3	91.3	92.3
$\text{PALT}_{\text{LARGE}}$ (ours)	93.8	91.7	92.8

Table 2: Triplet classification accuracy. Task-specific models are designed for knowledge graph completion, while general models are task agnostic.

form the transfer learning performance of finetuning with much fewer parameters.

Link Prediction Link prediction aims to predict a missing entity given relation and the other entity. It is a ranking problem where we are asked to rank all candidate entities and select the top answer to complete the missing part. For each fact *(h, r, t)*, we corrupt it by replacing either its head or tail entity with every other entity to form the candidate set. We follow Bordes et al. (2013) to use a filtered setting, i.e., all facts that appear in either train, dev or test data are removed, and use the remaining facts as the candidate set. Similar to triplet classification, each candidate fact is fed into PALT and the associated score is the output probability of the NSP classifier. We rank all candidates according to these scores. Two standard metrics are used for evaluation: Mean Rank (MR) and Hits@10 (the proportion of the correct entity ranked in the top 10). A lower MR is better while a higher Hits@10 is better.

The evaluation results of link prediction are shown in Table 3. $\text{PALT}_{\text{BASE}}$ achieves competitive or better performance than the finetuning approach. $\text{PALT}_{\text{LARGE}}$ performs better. In particular, $\text{PALT}_{\text{BASE}}$ outperforms KG-BERT by 1.4% in Hits@10 and 4 units in MR on FB15k-237; and 15.5% in Hits@10 and 35 units in MR on WN18RR. $\text{PALT}_{\text{LARGE}}$ outperforms $\text{PALT}_{\text{BASE}}$ by 1% in Hits@10, 5 units in MR on FB15k-237; and 1.4% in Hits@10 and 1 unit in MR on WN18RR. On UMLS, the finetuning model outperforms $\text{PALT}_{\text{BASE}}$ by a small margin. This is because pretrained LMs contain less medical knowledge due to a lack of medical corpus during pretraining. As a result, finetuning has the advantage over our approach on UMLS. The state-of-the-art task-specific model performs better than PALT. This is mainly because they leverage the structure information of KGs while the general models do not.

3.3 Ablation Study

To better understand PALT, we further conduct an ablation study on WN11 to show the effectiveness of different components. Specifically, we evaluate $\text{PALT}_{\text{BASE}}$ 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 $\text{BERT}_{\text{BASE}}$ 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) $\text{PALT}_{\text{BASE}}$ outperforms finetuning 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 our efficient transfer algorithm.

Method	FB15k-237		WN18RR		UMLS	
	Hits@10	MR	Hits@10	MR	Hits@10	MR
Task-specific models						
TransE (Bordes et al., 2013)	0.465	357	0.501	3384	0.989	1.84
DistMult (Yang et al., 2015)	0.419	254	0.49	5110	0.846	5.52
ComplEx (Trouillon et al., 2016)	0.428	339	0.51	5261	0.967	2.59
ConvE (Dettmers et al., 2018)	0.501	244	0.52	4187	0.990	1.51
RotatE (Sun et al., 2019)	0.533	177	0.571	3340	-	-
REFE (Chami et al., 2020)	0.541	-	0.561	-	-	-
HAKE (Zhang et al., 2020b)	0.542	-	0.582	-	-	-
KBGAT (Nathani et al., 2019)	0.626	210	0.581	1940	-	-
GAATs (Wang et al., 2020c)	0.650	187	0.604	1270	-	-
StAR (Wang et al., 2021a)	0.562	117	0.732	46	0.991	1.49
ComplEx-DURA (Zhang et al., 2020a)	0.560	-	0.571	-	-	-
General models						
KG-BERT (Yao et al., 2019)	0.420	153	0.524	97	0.990	1.47
$\text{PALT}_{\text{BASE}}$ (ours)	0.434	149	0.679	62	0.988	1.65
$\text{PALT}_{\text{LARGE}}$(ours)	0.444	144	0.693	61	0.990	1.57

Table 3: Link prediction results. Task-specific models are designed for knowledge graph completion, while general models are task agnostic.

Method	WN11
$\text{PALT}_{\text{BASE}}$	93.3
w/o Prompt	91.7
w/o $\text{Calibration}_{\text{middle}}$	92.2
w/o $\text{Calibration}_{\text{last}}$	93.0
w/o $\text{Calibration}_{\text{both}}$	89.3
w/o Encoder	73.7
Finetuning	93.2

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

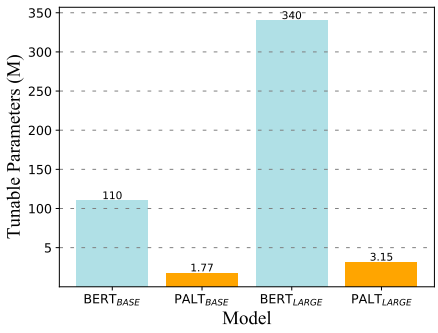


Figure 2: Compare the number of tunable parameters of PALT and BERT.

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. $\text{BERT}_{\text{BASE}}$ has 110M tunable parameters, while $\text{PALT}_{\text{BASE}}$ has 1.77M. $\text{PALT}_{\text{LARGE}}$ has 3.15M tunable parameters, while $\text{BERT}_{\text{LARGE}}$ has 340M. $\text{PALT}_{\text{BASE}}$ only has 1.6% tunable parameters of $\text{BERT}_{\text{BASE}}$, and $\text{PALT}_{\text{LARGE}}$ has 0.9% of $\text{BERT}_{\text{LARGE}}$. 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 = *"evening clothes"*, r = *"type of"* and t = *"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 pink) attends to the head and relation. The third 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.

Head	Relation	Tail	Label
tetsuzan nagata	cause of death	murder	✓
charles eliot	gender	male	✓
bill burrud	institution	harvard university	✓
lothar rendulic	nationality	germany	✓
thomas abbt	profession	philosopher	✓
samuel richardson	profession	priest	✗
nathaniel wallich	ethnicity	white british	✗
fu biao	gender	female	✗
alan turing	institution	harvard law school	✗
alan turing	ethnicity	israelis	✗

Table 5: Samples of PALT's correct predictions on FB13, where the finetuning method (Yao et al., 2019) outputs wrong predictions. Label ✓ means gold positive fact and ✗ indicates gold negative fact.

3.6 Error Analysis

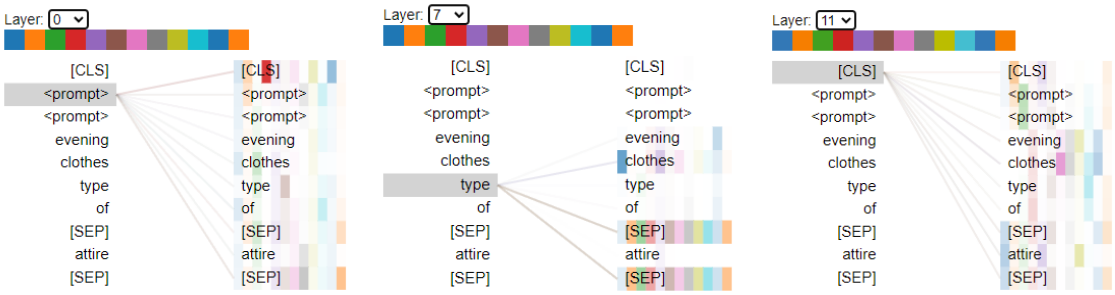
We analyze the errors made by PALT in this section. Here we focus on analyzing relations with the highest and lowest error rates. The detailed error rate statistics are shown in Appendix D. Most of PALT errors are due to *"domain"* relations, with an error rate of 14.7% for the relation *"domain topic"* 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 pretraining. 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.

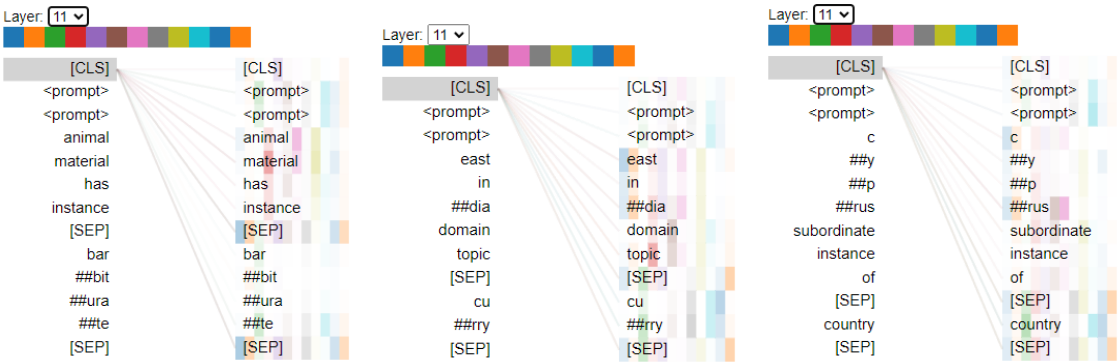
Compared to the existing methods, there are several distinctive features of PALT. First, instead of using the output of a single [MASK] token, we leverage the next sentence prediction, which allows the answers with arbitrary lengths. Second, we automatically acquire the template using the natural language 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.

Traditional KG completion methods mainly rely



(a) The first layer. (b) The middle layer. (c) The last layer.

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.



(a) *Has Instance.* (b) *Domain Topic.* (c) *Subordinate Instance.*

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

on graph structure information. They embed entities and relations into a continuous vector space and learn a score function based on these embeddings for triplets (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Nickel et al., 2011; Balažević et al., 2019; Dettmers et al., 2018; Nguyen et al., 2018; Cai and Wang, 2018). Unlike PALT, these methods treat entities and relations as unique identifiers and ignore their semantic meaning. Another line of research leverages text descriptions of entities and relations for KG completion. For example, KG-BERT (Yao et al., 2019) concatenates the text description of entities and relations into a sequence and feeds it into BERT, and finetunes the task-specific models. LASS (Shen et al., 2022) further uses both text and structure information to solve different KG completion tasks under a unified LM finetuning framework. By contrast, we present an alternative to finetuning for KG completion, and our method unifies different tasks in the same model architecture.

5 Conclusion

We propose PALT, a parameter-lite transfer of pretrained 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 $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. According to the estimation in (Strubell et al., 2019), pretraining a base model costs 1,507 kWh·PUE and emits 1,438 lb CO_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_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

Bo An, Bo Chen, Xianpei Han, and Le Sun. 2018. Accurate text-enhanced knowledge graph representation learning. In *NAACL: HLT*, page 745–755.

Carlos Aspillaga, Marcelo Mendoza, and Alvaro Soto. 2021. Inspecting the concept knowledge graph encoded by modern language models. In *Findings of ACL*, pages 2984–3000.

Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, page 5184–5193.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, page 2787–2795.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*, pages 1877–1901.

Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *NAACL: HLT*, page 1470–1480.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *ACL*, pages 6901–6914.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL-IJCNLP*, pages 3816–3830.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL-IJCNLP*, page 687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI*, page 985–991.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *TNNLS*, pages 494–514.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, pages 423–438.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *TACL*, pages 64–77.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, pages 1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*, pages 7871–7880.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, page 2181–2187.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR*.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Alexa T. McCray. 2003. An upper-level ontology for the biomedical domain. *Comp. Funct. Genomics*, page 80–84.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *ACL*, pages 4710–4723.

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

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. *NAACL: HLT*, page 327–333.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, page 809–816.

Janna Omeliyanenko, Albin Zehe, Lena Hettinger, and Andreas Hotho. 2020. LM4KG: improving common sense knowledge graphs with language models. In *ISWC*, pages 456–473.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *WWW*, page 1419–1428.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *NAACL: HLT*, pages 2523–2544.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR*, pages 1–67.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607.

Jianhao Shen, Chenguang Wang, Linyuan Gong, and Dawn Song. 2022. Joint language semantic and structure embedding for knowledge graph completion. In *COLING*, pages 1965–1978.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*, pages 4222–4235.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, page 926–934.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650.

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

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *CVSC*, page 57–66.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, page 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *ACL*, page 37–42.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP*, pages 353–355.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, and Yi Chang. 2021a. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. *WWW*, pages 1737–1748.

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

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2022. DeepStruct: Pretraining of language models for structure prediction. In *ACL*.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020a. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Haoyu Wang, Vivek Kulkarni, and William Yang Wang. 2020b. Dolores: Deep contextualized knowledge graph embeddings. In *AKBC*.

Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. 2020c. Knowledge Graph Embedding via Graph Attenuated Attention Networks. *IEEE Access*, pages 5212–5224.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021c. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of ACL-IJCNLP*, pages 1405–1418.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, page 1112–1119.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*, page 1293–1299.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Transg: A generative model for knowledge graph embedding. In *ACL*, page 2316–2325.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, page 2659–2665.

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*.

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

Zhanqiu Zhang, Jianyu Cai, and Jie Wang. 2020a. Duality-induced regularizer for tensor factorization based knowledge graph completion. In *NeurIPS*, pages 21604–21615.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI*, pages 3065–3072.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. 2018. Knowledge graph embedding with hierarchical relation structure. In *EMNLP*, page 3198–3207. ACL.

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 [$1.5*10^{-4}$, $1.5*10^{-5}$, $1*10^{-4}$, $1*10^{-5}$] 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 $\text{PALT}_{\text{BASE}}$ and $\text{PALT}_{\text{LARGE}}$. For knowledge prompt encoder, we add it to the input layer. In particular, 10 prompt tokens are added at 3 different positions for $\text{PALT}_{\text{BASE}}$ 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 $\text{PALT}_{\text{LARGE}}$, 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 18 relations and 40,943 entities. 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), TranSparse-S (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).

[2]We refer the readers to (Ji et al., 2022) for a more comprehensive review of the KG completion methods.

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 $n_p * d_e + d_e * d_h$, and that of knowledge calibration encoder is $2 * d_h * d_h$. In total, there are $n_p * d_e + d_e * d_h + 2 * d_h * d_h$ tunable parameters in PALT. For $\text{BERT}_{\text{BASE}}$, $d_e = d_h = 768$, and for $\text{BERT}_{\text{LARGE}}$, $d_e = d_h = 1024$, and we use $n_p = 2$ on WN11. As a result, $\text{PALT}_{\text{BASE}}$ has 1.77M tunable parameters, and $\text{PALT}_{\text{LARGE}}$ 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

Method	Score Function	
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
TransH	$-\left\|\left(\mathbf{h}-\mathbf{w}_r^\top\mathbf{h}\mathbf{w}_r\right)+\mathbf{r}-\left(\mathbf{t}-\mathbf{w}_r^\top\mathbf{t}\mathbf{w}_r\right)\right\|$	$\mathbf{h},\mathbf{t},\mathbf{r},\mathbf{w}_r\in\mathbb{R}^k$
TransR	$-\|\mathbf{M}_r\mathbf{h}+\mathbf{r}-\mathbf{M}_r\mathbf{t}\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k,\mathbf{M}_r\in\mathbb{R}^{k\times d}$
TransD	$-\left\|\left(\mathbf{w}_r\mathbf{w}_h^\top+\mathbf{I}\right)\mathbf{h}+\mathbf{r}-\left(\mathbf{w}_r\mathbf{w}_t^\top+\mathbf{I}\right)\mathbf{t}\right\|$	$\mathbf{h},\mathbf{t},\mathbf{w}_h\mathbf{w}_t\in\mathbb{R}^k,\mathbf{r},\mathbf{w}_r\in\mathbb{R}^d$
TransG	$\sum_i \pi_r^i \exp\left(-\frac{\|\boldsymbol{\mu}_h+\boldsymbol{\mu}_r^i-\boldsymbol{\mu}_t\|}{\sigma_h^2+\sigma_t^2}\right)$	$\mathbf{h}\sim\mathcal{N}\left(\boldsymbol{\mu}_h,\boldsymbol{\sigma}_h^2\mathbf{I}\right),\mathbf{t}\sim\mathcal{N}\left(\boldsymbol{\mu}_t,\Sigma_t\right),\boldsymbol{\mu}_h,\boldsymbol{\mu}_t\in\mathbb{R}^k$
TranSparse-S	$-\left\|\mathbf{M}_r\left(\theta_r\right)\mathbf{h}+\mathbf{r}-\mathbf{M}_r\left(\theta_r\right)\mathbf{t}\right\|_{1/2}^2-\left\|\mathbf{M}_r^1\left(\theta_r^1\right)\mathbf{h}+\mathbf{r}-\mathbf{M}_r^2\left(\theta_r^2\right)\mathbf{t}\right\|_{1/2}^2$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k,\mathbf{r}\in\mathbb{R}^d,\mathbf{M}_r\left(\theta_r\right)\in\mathbb{R}^{k\times d},\mathbf{M}_r^1\left(\theta_r^1\right),\mathbf{M}_r^2\left(\theta_r^2\right)\in\mathbb{R}^{k\times d}$
DistMult	$\langle\mathbf{r},\mathbf{h},\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ConvKB	$\text{concat}(g([\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}]*\omega))\mathbf{w}$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ComplEx	$\Re(\langle\mathbf{r},\mathbf{h},\bar{\mathbf{t}}\rangle)$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k$
ConvE	$\langle\sigma(\text{vec}(\sigma([\bar{\mathbf{r}},\bar{\mathbf{h}}]*\boldsymbol{\Omega}))\mathbf{W}),\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
RotatE	$-\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|^2$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k,\|r_i\|=1$
REFE	$-\text{arctanh}(\|-\langle\mathbf{h},\text{Ref}(\mathbf{r})\rangle\oplus^c\mathbf{t}\|)$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
HAKE	$\text{RotatE}-\|\sin((\mathbf{h}+\mathbf{r}-\mathbf{t})/2)\|_1$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ComplEx-DURA	$\text{ComplEx}-\langle\mathbf{h},\mathbf{r}\rangle^2-\|\mathbf{t}\|^2$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k$

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

(colored 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.

Head	Relation	Tail	Label
center	has instance	olfactory brain	✓
family graminaceae	member meronym	meadow grass	✓
botany	domain region	style	✓
end	has instance	complete	✓
fictionalise	type of	convert	✓
archaeology	domain region	unreactive	✗
cognitive content	has instance	diacritic	✗
anura	member meronym	kuru	✗
atlantic	has part	tocantins	✗
sorb	part of	electric resistance	✗

Table 7: Triplets of WN11 that are correctly predicted by PALT while KG-BERT fails. Label ✓means a positive triplet and ✗means negative.

D Error Analysis

Here we give the error rate of $\text{PALT}_{\text{BASE}}$ on each relation of WN11 in Table 8. "Domain topic" and "domain region" are the two relations with the highest error rates, while "subordinate instance of" has the lowest error rate.

E Prompt Analysis

We evaluate different numbers and positions of prompt tokens on WN11. We use a sequence X_1-X_2-X_3 to denote the numbers of tokens added in

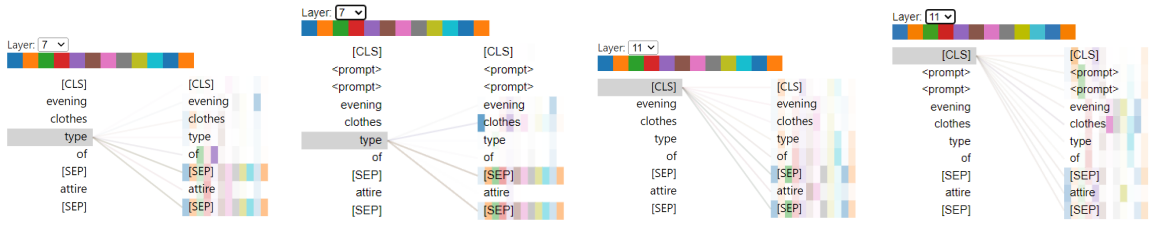
Relation	Error Rate(%)
domain topic	14.7
domain region	10.5
has instance	8.3
member meronym	8.1
synset domain topic	7.4
similar to	7.1
has part	7.0
part of	5.7
type of	5.3
member holonym	3.9
subordinate instance of	2.6

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.

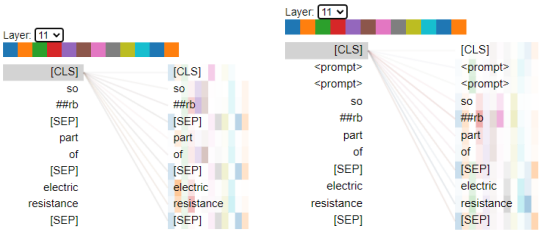
F Effectiveness of Calibration

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



(a) BERT-middle (b) PALT-middle (c) BERT-last (d) PALT-last

Figure 5: Attentions weights of the original BERT and PALT.



(a) KG-BERT (b) 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.

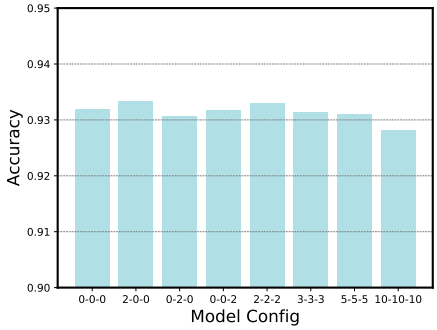


Figure 7: Accuracy for different numbers and positions of prompt tokens on WN11.

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.