Dipping PLMs Sauce: Bridging Structure and Text for Effective Knowledge Graph Completion via Conditional Soft Prompting

Chen Chen[1], **Yufei Wang**[2], **Aixin Sun**[1], **Bing Li**[3,4] and **Kwok-Yan Lam**[1*]
Nanyang Technological University, Singapore[1]
Macquarie University, Sydney, Australia[2]
IHPC[3] and CFAR[4], Agency for Science, Technology and Research (A*STAR), Singapore
`{S190009,axsun,kwokyan.lam}@ntu.edu.sg,`
`yufei.wang@students.mq.edu.au`
`li_bing@cfar.a-star.edu.sg`

Abstract

Knowledge Graph Completion (KGC) often requires both KG structural and textual information to be effective. Pre-trained Language Models (PLMs) have been used to learn the textual information, usually under the fine-tune paradigm for the KGC task. However, the fine-tuned PLMs often overwhelmingly focus on the textual information and overlook structural knowledge. To tackle this issue, this paper proposes CSProm-KG (**C**onditional **S**oft **Prom**pts for **KG**C) which maintains a balance between structural information and textual knowledge. CSProm-KG only tunes the parameters of *Conditional Soft Prompts* that are generated by the entities and relations representations. We verify the effectiveness of CSProm-KG on three popular static KGC benchmarks WN18RR, FB15K-237 and Wikidata5M, and two temporal KGC benchmarks ICEWS14 and ICEWS05-15. CSProm-KG outperforms competitive baseline models and sets new state-of-the-art on these benchmarks. We conduct further analysis to show (i) the effectiveness of our proposed components, (ii) the efficiency of CSProm-KG, and (iii) the flexibility of CSProm-KG [1].

1 Introduction

Knowledge Graphs (KGs) have both complicated graph structures and rich textual information over the facts. Despite being large, many facts are still missing. Knowledge Graph Completion (KGC) is a fundamental task to infer the missing facts from the existing KG information.

Graph-based KGC models (Bordes et al., 2013; Yang et al., 2015; Dettmers et al., 2018) represent entities and relations using trainable embeddings. These models are trained to keep the connections between entities and relations over structural paths,

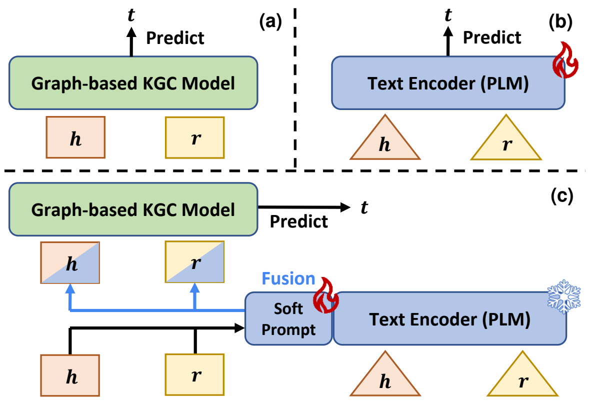

Figure 1: Given head entity h and relation r, KGC is to find out the true tail entity t. Graph-based KGC models represent h and r as embeddings (rectangular boxes) to learn the KG structure information (Figure.a). PLM-based KGC models only feed the textual knowledge (triangle boxes) of h and r into the Pre-trained Language Model (PLM) to predict the missing entity (Figure.b). CSProm-KG fuses both types of information via the *Soft Prompt* and uses a graph-based KGC model to make the final prediction (Figure.c).

and tail entities are inferred via various transitional relations. Despite being effective in modelling KG structural information, these methods are unable to incorporate linguistic context. Recently, pre-trained language models (PLMs) are applied to fill up this gap (Yao et al., 2019; Wang et al., 2021a; Xie et al., 2022). The proposed solutions often directly fine-tune the PLMs to choose the correct entities either relying on pure textual context or using structural add-ons as a complementary (Wang et al., 2021a). However, PLMs are normally equipped with large-scale parameters and linguistic inherence obtained from their pre-training stage. As a result, these PLM-based models remain overwhelmingly focusing on the textual information in KGs and tend to overlook the graph structure. For example, given an incompleted fact *(Mona Lisa, painted by, ?)*, the PLM-based models may con-

[*] Corresponding author
[1] Our source code is available at `https://github.com/chenchens190009/CSProm-KG`

fuse between *Leonardo DiCaprio* and *Leonardo da Vinci* simply because they are textually similar. Thus, in this paper, we focus on the research question: *Can we effectively fuse the KG structural information into the PLM-based KGC models?*

To this end, we propose a novel CSProm-KG model (**C**onditional **S**oft **Prom**pts for **KG**C) which is a structure-aware frozen PLMs that could effectively complete the KGC task. The core of CSProm-KG is *Conditional Soft Prompt* that is an structure-aware version of *Soft Prompt* (Li and Liang, 2021; Lester et al., 2021). Previously, *Soft Prompt* is a sequence of *unconditional* trainable vectors that are prepended to the inputs of frozen PLMs. Such design could effectively avoid the issue of over-fitting towards textual information caused by fine-tuning and allow the frozen PLMs to learn the downstream tasks (Wang et al., 2022). However, such naive *Soft Prompts* cannot represent any structural information in KG. To remedy this, as shown in Figure 1 (c), we propose the prompt vectors *conditioned* on the KG entities and relations embeddings. Specifically, we use the entity and relation embeddings to generate *Conditional Soft Prompts* which are then fed into the frozen PLMs to fuse the textual and structural knowledge together. The fused *Conditional Soft Prompts* are used as inputs to the graph-based KGC model that produces the final entity ranking results. We further propose *Local Adversarial Regularization* to improve CSProm-KG to distinguish textually similar entities in KG.

We evaluate CSProm-KG on various KGC tasks and conduct experiments on WN18RR, FB15K-237 and Wikidata5M for Static KGC (SKGC), and on ICEWS14 and ICEWS05-15 for Temporal KGC (TKGC). CSProm-KG outperforms a number of competitive baseline models, including both graph-based and PLM-based models. We conduct ablation studies to show the strength of prompt-based methods against the fine-tuning counterparts and the effectiveness of each proposed components. We also demonstrate the flexibility of CSProm-KG with different graph-based models, and the training and inference efficiency of CSProm-KG.

2 Related Work

Graph-based methods Graph-based methods represent each entity and relation with a continuous vector by learning the KG spatial structures. They use these embeddings to calculate the distance between the entities and KG query to determine the correct entities. The training objective is to assign higher scores to true facts than invalid ones. In static KGC (SKGC) task, there are two types of methods: 1) Translational distance methods measure the plausibility of a fact as the distance between the two entities, (Bordes et al., 2013; Lin et al., 2015; Wang et al., 2014); 2) Semantic matching methods calculate the latent semantics of entities and relations (Nickel et al., 2011; Yang et al., 2015; Dettmers et al., 2018). In temporal KGC (TKGC) task, the systems are usually based on SKGC methods, with additional module to handle KG factual tuples timestamps (Dasgupta et al., 2018; Goel et al., 2020; Han et al., 2021).

PLM-based methods PLM-based methods represent entities and relations using their corresponding text. These methods introduce PLM to encode the text and use the PLM output to evaluate the plausibility of the given fact. On SKGC, Yao et al. (2019) encode the combined texts of a fact, then a binary classifier is employed to determine the plausibility. To reduce the inference cost in Yao et al. (2019), Wang et al. (2021a) exploit Siamese network to encode (h, r) and t separately. Unlike previous encode-only model, Xie et al. (2022); Saxena et al. (2022) explore the *Seq2Seq* PLM models to directly generate target entity text on KGC task.

Prompt tuning Brown et al. (2020) first find the usefulness of prompts, which are manually designed textual templates, in the GPT3 model. Wallace et al. (2019); Shin et al. (2020) extend this paradigm and propose hard prompt methods to automatically search for optimal task-specific templates. However, the selection of discrete prompts involves human efforts and difficult to be optimized together with the downstream tasks in an end-to-end manner. (Li and Liang, 2021; Lester et al., 2021) relax the constraint of the discrete template with trainable continuous vectors (soft prompt) in the frozen PLM. As shown in Li and Liang (2021); Lester et al. (2021); Liu et al. (2021), frozen PLM with *Soft Prompt* could achieve comparative performance on various NLP tasks, despite having much less parameters than fully trainable PLM models. To the best of our knowledge, we are the first to apply *Soft Prompt* to PLM-based KGC model.

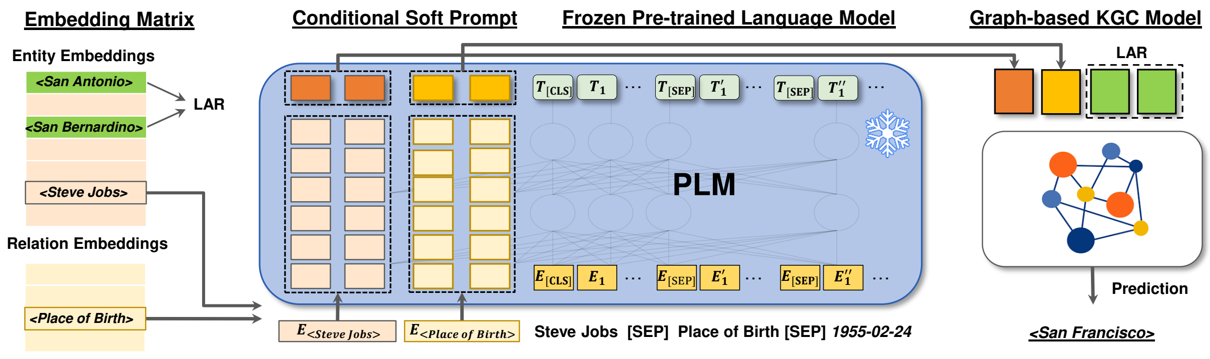
Embedding Matrix
Entity Embeddings
<San Antonio>
<San Bernardino>
LAR
<Steve Jobs>
Relation Embeddings
<Place of Birth>
Conditional Soft Prompt
Frozen Pre-trained Language Model
PLM
$T_{[CLS]}$ T_1 $T_{[SEP]}$ T'_1 $T_{[SEP]}$ T''_1
$E_{[CLS]}$ E_1 $E_{[SEP]}$ E'_1 $E_{[SEP]}$ E''_1
$E_{<Steve\ Jobs>}$ $E_{<Place\ of\ Birth>}$
Steve Jobs [SEP] Place of Birth [SEP] 1955-02-24
Graph-based KGC Model
LAR
Prediction
<San Francisco>

Figure 2: An example of CSProm-KG for the KG query (*Steve Jobs*, *Place of Birth*, ?, 1955-02-24). CSProm-KG uses the embeddings of entities and relations (randomly initialized before training) to generate *Conditional Soft Prompt*. In the frozen PLMs, *Conditional Soft Prompt* fully interacts with the textual information of the KG queries. The outputs are fed into graph-based KGC model to make the final prediction. To improve CSProm-KG's ability in distinguishing textually similar entities, we further add *LAR* examples that are similar to the tail entities during training. CSProm-KG effectively learns both structural and textual knowledge in KG.

3 Method

In this section, we first formulate *Knowledge Graph Completion* in Sec. 3.1. We then introduce CSProm-KG in Sec. 3.2 to Sec. 3.7.

3.1 Knowledge Graph Completion

Knowledge graph (KG) is a directed graph with a collection of fact tuples. Let $T = \{V, R, L, M\}$ be a KG instance, where V, R, L and M denote the entity, relation, edge (fact) and meta information set respectively. Each edge $e \in L$ is $(h, r, t, m) \in V \times R \times V \times M$ which connects head entity h and target entity t with relation type r, and is associated with meta information m. In Static KGs (SKG), no meta information is involved (i.e. $M = \emptyset$). In Temporal KGs (TKG), each fact has a corresponding timestamp and M includes all fact timestamps. *Knowledge Graph Completion* (KGC) is to predict the target entity for KG queries $(h, r, ?, m)$. The queries $(?, r, t, m)$ are converted into $(t, r^{-1}, ?, m)$, where r^{-1} is the inverse of r. In this paper, CSProm-KG learns a score function $f(h, r, t, m) : V \times R \times V \times M \rightarrow V$ that assigns a higher score for valid facts than the invalid ones.

3.2 CSProm-KG Overview

Motivated by the observation that *Soft Prompts* in a frozen PLM is effective in solving the over-fitting issue (Wang et al., 2022), we apply *Soft Prompts* in CSProm-KG to avoid the KGC models overly focusing on the textual information. Although several research initiatives have explored the utilization of both structural and textual information for NLP tasks (Li et al., 2022; Xiao et al., 2021), none of them is capable of solving the over-fitting issue over textual information in the context of KGC. Figure 2 shows the architecture of CSProm-KG which includes three important components: a fully trainable *Graph-based KGC model* G, a frozen Pre-trained language model (PLM) P, and a trainable *Conditional Soft Prompt* S. Firstly, the embeddings in G, which are *explicitly* trained to predict entities using structural knowledge, are used to generate the parameters of S. In this way, S is equipped with KG structural knowledge. We then feed the generated S, as well as the corresponding text of entities and relations, into P. Finally, the PLM outputs of S are extracted as the final inputs to G which produces final results for the KGC tasks. This allows the structural knowledge from G and the textual knowledge from P to be equally fused via S. To further improve the robustness of CSProm-KG, we propose *Local Adversarial Regularization*, which selects textually similar entities for training to be detailed shortly.

3.3 Graph-based KGC Model G

In CSProm-KG, the graph-based KGC models G represents KG entities and relations as continuous embeddings. Given a KG query $(h, r, ?, m)$, we represent h and r as embeddings E_e and $E_r \in \mathbb{R}^d$ where d is the embedding size. E_e and E_r are used at both *inputs* and *outputs*. At *inputs*, we use these embeddings to generate *Conditional Soft Prompt* which further interacts with the textual inputs of the frozen PLM P. At *outputs*, we use these embeddings to calculate $f(h, r, t, m)$ which produces the entity ranking for KG queries. For example, when using ConvE as G, the corresponding $f(h, r, t, m)$ is the dot-product between the representation of (h, r) and the tail entity embeddings. Note that, CSProm-KG is flexible enough to work well with any existing graph-based KGC models. We will show this flexibility in Sec. 4.4.

3.4 Pre-trained Language Model P

Let's assume that the pre-trained language model P has l transformer layers with hidden size H. To represent a KG query $(h, r, ?, m)$, we jointly represent h, r and m by extracting and concatenating their corresponding raw tokens, including their names and their corresponding descriptions if available. We connect the texts of h and r with a special token [SEP], and feed the joint text into the frozen PLM P. For TKGC tasks, we simply add the event timestamp after the joint text of h and r. We show the effectiveness of this design choice in Sec. 4.2.

3.5 Conditional Soft Prompt S

Soft Prompt prepends a sequence of trainable embeddings at the inputs to a frozen Pre-trained Language model. Li and Liang (2021) propose *Layer-wise Soft Prompt* which inserts relatively short prompt sequences (e.g., 5 - 10 vectors) at each layer and allows frequent interaction with the entities' and relations' textual information in PLMs. Inspired by this, we propose a novel *Conditional Soft Prompt* which has k trainable vectors on each layer. Specifically, the i^{th} input for the j^{th} layer $\boldsymbol{h}_i^j \in \mathbb{R}^H$ is defined as:

$$\boldsymbol{h}_i^j = \begin{cases} \boldsymbol{s}_i^j & i \leq k \\ \boldsymbol{w}_i & (i > k) \wedge (j = 0) \\ Trans(\boldsymbol{h}_:^{j-1})_i & \text{Otherwise} \end{cases} \tag{1}$$

where $Trans(\cdot)$ is the forward function of Transformer layer in P, w_i is the fixed input word embedding vector and $\boldsymbol{s}_i^j$ is the i^{th} prompt vector at j^{th} layer. The $Trans(\cdot)$ works on the entire sequence (prompt + text). *Conditional Soft Prompt* is designed to connect with embeddings in G, we use the embeddings of entities and relations E_e and E_r to generate *Conditional Soft Prompt* S. Formally,

$$S = [F(E_e); F(E_r)] \tag{2}$$

$$F(x) = W_{out} \cdot (\text{ReLU}(W_{in} \cdot x)) \tag{3}$$

where $W_{in} \in \mathbb{R}^{d_h \times d}$ and $W_{out} \in \mathbb{R}^{(l*H*k/2)\times d_h}$ are trainable weight matrices and d_h is the middle hidden size for the mapping layers. We then re-organize $F(E_e)$ and $F(E_r)$ into a sequence of input embeddings and evenly distribute them into each PLM layer. In this process, the input tokens for P and *Conditional Soft Prompt* S are fully interacted with each other, allowing the structural knowledge in G (linearly mapped to S) and textual knowledge in P to be fully fused together.

3.6 Local Adversarial Regularization

As PLMs are frozen, the model may lose part of flexibility in distinguishing textually similar entities via tuning of the Transformer layers. To enhance CSProm-KG's ability to distinguish textually similar entities, inspired by (Goodfellow et al., 2015), we introduce an Adversarial Regularization term. Different from conventional adversarial regularization which generates virtual examples that do not exist, our adversarial examples are picked from the local entity set V that are of concrete meanings. Specifically, given a KG query $(h, r, ?, m)$ and ground-truth entity t, CSProm-KG treats entities that are textually similar to t as adversarial examples. We refer these samples as *Local Adversarial Regularization* (LAR) entities. To allow efficient training, we define LAR samples as the ones sharing the common tokens in entity names and descriptions with t, enabling us to pre-compute these LAR samples before training. This is different from previous works (Miyato et al., 2017; Madry et al., 2018; Goodfellow et al., 2015) that generate virtual adversarial examples using training perturbation with large computational costs. Specifically, the LAR training objective is:

$$\begin{aligned}\mathcal{L}_l(h, r, t, m) = \max(&f(h, r, t, m) \\ &-\frac{1}{n}\sum_{i=0}^{n} f(h, r, t_i^{\Delta}, m) + \gamma, 0)\end{aligned} \tag{4}$$

where t_i^{Δ} is an sampled LAR entity of t, γ is the margin hyperparameter, n is the number of sampled LAR entities.

3.7 Training and Inference

For training, we leverage the standard cross entropy loss with label smoothing and LAR:

$$\begin{aligned}\mathcal{L}_c(h,r,t,m) = &-(1-\epsilon)\cdot \log p(t|h,r,m) \\ &-\frac{\epsilon}{|V|}\sum_{t' \in V/t} \cdot \log p(t'|h,r,m)\end{aligned} \tag{5}$$

$$\mathcal{L} = \sum_{(h,r,t,m)\in T} \mathcal{L}_c(h,r,t,m) + \alpha \cdot \mathcal{L}_l(h,r,t,m) \tag{6}$$

	WN18RR				FB15K-237				Wikidata5M			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Graph-Based Methods												
TransE (Bordes et al., 2013)	.243	.043	.441	.532	.279	.198	.376	.441	.253	.170	.311	.392
DistMult (Yang et al., 2015)	.444	.412	.470	.504	.281	.199	.301	.446	.253	.209	.278	.334
ComplEx (Trouillon et al., 2016)	.449	.409	.469	.530	.278	.194	.297	.450	.308	.255	-	.398
ConvE (Dettmers et al., 2018)	.456	.419	.470	.531	.312	.225	.341	.497	-	-	-	-
RotatE (Sun et al., 2019)	.476	.428	.492	.571	.338	.241	.375	.533	.290	.234	.322	.390
CompGCN (Vashishth et al., 2020)	.479	.443	.494	.546	.355	.264	.390	.535	-	-	-	-
PLM-Based Methods												
KG-BERT (Yao et al., 2019)	.216	.041	.302	.524	-	-	-	.420	-	-	-	-
MTL-KGC (Kim et al., 2020)	.331	.203	.383	.597	.267	.172	.298	.458	-	-	-	-
StAR (Wang et al., 2021a)	.401	.243	.491	**.709**	.296	.205	.322	.482	-	-	-	-
MLMLM (Clouâtre et al., 2021)	.502	.439	.542	.611	-	-	-	-	.223	.201	.232	.264
KEPLER (Wang et al., 2021b)	-	-	-	-	-	-	-	-	.210	.173	.224	.277
GenKGC (Xie et al., 2022)	-	.287	.403	.535	-	.192	.355	.439	-	-	-	-
KGT5 (Saxena et al., 2022)	.508	.487	-	.544	.276	.210	-	.414	.300	.267	.318	.365
KG-S2S (Chen et al., 2022)	.574	**.531**	.595	.661	<u>.336</u>	<u>.257</u>	<u>.373</u>	.498	-	-	-	-
CSProm-KG	**.575**	<u>.522</u>	**.596**	<u>.678</u>	**.358**	**.269**	**.393**	**.538**	**.380**	**.343**	**.399**	**.446**

Table 1: Experimental results of different baseline methods on the SKGC datasets. WN18RR and FB15K-237 results are taken from Wang et al. (2021a). Wikidata5M results are taken from Saxena et al. (2022). The best PLM-based method results are in bold and the second best results are underlined.

where $p(t|h,r,m) = \frac{\exp f(h,r,t,m)}{\sum_{t' \in V} \exp f(h,r,t',m)}$, ϵ is the label smoothing value and α is the LAR term weight. For inference, CSProm-KG first computes the representations for KG query $(h, r, ?, m)$, then uses the entity embeddings in G to compute the entity ranking. While other PLM-Based KGC models such as StAR (Wang et al., 2021a) requires $|V|$ PLM forward pass computation for entity embeddings. Thus, CSProm-KG is more computationally efficient than these baselines (See Sec. 4.3).

4 Experiments

In this section, we first compare CSProm-KG with other competitive baselines in the SKGC and TKGC benchmarks in Sec. 4.1. We then conduct ablation studies to verify the effectiveness of our propose components in CSProm-KG in Sec. 4.2. We further show the efficiency and flexibility of CSProm-KG in Sec. 4.3 and 4.4, respectively.

Dataset WN18RR (Dettmers et al., 2018) and FB15K-237 (Toutanova and Chen, 2015) are the most popular SKGC benchmarks where all inverse relations are removed to avoid data leakage. Wikidata5M (Wang et al., 2021b) is a recently proposed large-scale SKGC benchmark. For TKGC, we use ICEWS14 (García-Durán et al., 2018) and ICEWS05-15 (García-Durán et al., 2018) which include political facts from the Integrated Crisis Early Warning System (Boschee et al., 2015). More dataset details are shown in Table 8.

Implementation Details All the experiments are conducted on a single GPU (RTX A6000). We tune the learning rate $\eta \in \{1e{-}3, 5e{-}4, 1e{-}4\}$, batch size $\mathcal{B} \in \{128, 256, 384, 450\}$, prompt length $\mathcal{P}_l \in \{2, 5, 10\}$ and LAR term weight $\alpha \in \{0.0, 0.1, 0.2\}$. While $\alpha > 0$, we employ 8 LAR samples for each training instance and gradually increase the LAR term weight from 0 to α using a step size of $\alpha_{step} = 1e{-}5$. CSProm-KG uses the BERT-Large (Devlin et al., 2019) and ConvE (Dettmers et al., 2018) model. We set the label smoothing to 0.1 and optimize CSProm-KG with AdamW (Loshchilov and Hutter, 2019). We choose the checkpoints based on the validation mean reciprocal rank (MRR). We follow the *filtered setting* in Bordes et al. (2013) to evaluate our model. Detailed model hyperparameters for each dataset are shown in Appendix B.

4.1 Main result

Table 1 and Table 2 present the main SKGC and TKGC results, respectively, which demonstrate statistical significance (t-student test, $p < 0.05$).

Results on SKGC As for the popular medium-sized KGC benchmarks, CSProm-KG achieves state-of-the-art or competitive performance compared with PLM-based KGC models. In particular, on FB15K-237, CSProm-KG consistently outperforms all PLM-based KGC models and achieves 6.5% (from 0.336 to 0.358) relative MRR im-

	ICEWS14				ICEWS05-15			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Graph-Based Methods								
TTransE (Leblay and Chekol, 2018)	.255	.074	-	.601	.271	.084	-	.616
HyTE (Dasgupta et al., 2018)	.297	.108	.416	.655	.316	.116	.445	.681
ATiSE (Xu et al., 2019)	.550	.436	.629	.750	.519	.378	.606	.794
DE-SimplE (Goel et al., 2020)	.526	.418	.592	.725	.513	.392	.578	.748
Tero (Xu et al., 2020)	.562	.468	.621	.732	.586	.469	.668	**.795**
TComplEx (Lacroix et al., 2020)	.560	.470	.610	.730	.580	.490	.640	.760
TNTComplEx (Lacroix et al., 2020)	.560	.460	.610	.740	.600	.500	.650	.780
T+TransE (Han et al., 2021)	.553	.437	.627	.765	-	-	-	-
T+SimplE (Han et al., 2021)	.539	.439	.594	.730	-	-	-	-
PLM-Based Methods								
KG-S2S (Chen et al., 2022)	.595	.516	.642	.737	-	-	-	-
CSProm-KG	**.628**	**.548**	**.677**	**.773**	**.628**	**.545**	**.678**	.783

Table 2: Experimental results of different baseline methods on the TKGC datasets. The results of baseline are obtained from original papers.

provement. These PLM-based baselines are all fully fine-tuned, indicating the importance of using parameter-effective prompts in the KGC task. Compared with graph-based methods, CSProm-KG outperforms baseline methods by a large margin on WN18RR (i.e. 0.575 *v.s.* 0.479 on MRR) and on FB15K-237 (i.e. 0.358 *v.s.* 0.355 on MRR). Noted that the improvement on FB15K-237 is barely comparable to that on WN18RR, and this discrepancy can be explained by the existence of Cartesian Product Relations (CPRs) in FB15K-237, which are noisy and semantically meaningless relations (Chen et al., 2022; Lv et al., 2022; Akrami et al., 2020). On the Wikidata5M benchmark, CSProm-KG significantly outperforms previous methods, showing the advantages of CSProm-KG on the large-scale KGs. These results verify that with frozen PLM and accordingly much less trainable parameters, CSProm-KG can achieve remarkable performance on various KGs with different scales.

Results of TKGC Table 2 reports the experiment results on the ICEWS14 and ICEWS05-15 benchmarks. On ICEWS14, CSProm-KG substantially outperforms existing TKGC methods (e.g., at least 0.03 MRR higher than previous works). On ICEWS05-15, CSProm-KG is 0.028 and 0.045 higher than the best TKGC methods in terms of MRR and H@1, though being slightly worse on H@10 than Tero and ATiSE. On both benchmarks, CSProm-KG sets new state-of-the-art performance. Note that the TKGC baseline models are often specifically designed and optimized for the TKGC task, while the only modification to CSProm-KG is to add timestamp into its input. This further shows that our proposed CSProm-KG method is a generally strong solution for various of KGC tasks.

4.2 Ablation Studies

We conduct ablation study to show the effectiveness of our proposed components on WN18RR. Table 3 and Figure 5 summarize the ablation study results.

No.	Model	MRR	H@1	H@10
1	CSProm-KG	.575	.522	.678
2	CSProm-KG w/ *Separated Strategy*	.520	.470	.622
3	CSProm-KG w/o Graph KGC model	.545	.495	.645
4	CSProm-KG w/ non-LW Soft Prompt	.522	.473	.612
5	CSProm-KG w/o LAR	.534	.489	.624
6	CSProm-KG w/ LAR from Name	.557	.513	.643
7	CSProm-KG w/ LAR from Description	.551	.501	.647
8	CSProm-KG w/ Random LAR	.545	.500	.630
9	CSProm-KG w/ the last layer tunable	.537	.494	.621
10	CSProm-KG w/ the last 4 layers tunable	.437	.410	.488
11	CSProm-KG w/ the last 6 layers tunable	.441	.415	.493
12	CSProm-KG w/ fully finetune	.436	.409	.484
13	Ensemble model	.481	.549	.630

Table 3: Ablation Study regarding important components in CSProm-KG on the benchmark of WN18RR.

KG Query Structure As we discussed in Sec. 3, for each KG Query $(h, r, ?, m)$, we *jointly* concatenate their textual information and feed them into the frozen PLM (as shown in Figure 3). To demonstrate the effectiveness of this design choice, we replace it with a *Separated Strategy* that is similar to the Siamese network used in Wang et al. (2021a). That is, as shown in Figure 4, we separately encode the textual information of h and r using PLMs. Table 3 Line 2 shows the performance of this *Sep-*

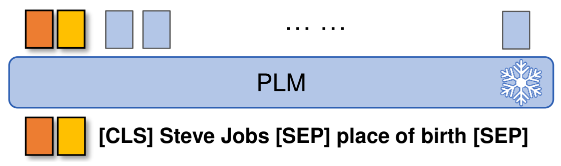

Figure 3: *Joint Strategy* used in CSProm-KG.

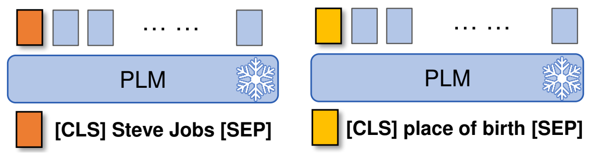

Figure 4: *Separated Strategy* used in the ablation study.

arated Strategy. Compared to CSProm-KG, the performance drops by 0.055 on MRR and 0.056 on H@10. The mixture of soft prompts and text representation concatenation increase the interaction between entity and relations, allowing better representation of KG Query.

Role of Graph-based KGC Models Table 3 Line 3 shows the performance of CSProm-KG without any graph-based KGC models. For this ablation, we directly use the outputs of PLM to predict the target entity. We observe that removing this graph-based KGC model leads to a performance drop (i.e., by 0.030 MRR and 0.033 H@10). This shows that even after the complex interaction in the PLMs, an appropriate graph-based KGC model could still provide additional useful structural knowledge. This experiment verifies the necessity of combining PLM-based and graph-based KGC models together.

Soft Prompt Design Lester et al. (2021) recently propose another *Soft Prompt* variant which puts longer trainable vectors at the bottom input layer. We refer it as *non-layer-wise Soft Prompt*. Table 3 Line 4 shows the performance using this variant on WN18RR. CSProm-KG with *layer-wise soft prompt* model outperforms the non-layer-wise counterpart by a large margin (i.e., 0.053 MRR and 0.066 H@10), which suggests that the layer-wised *Soft Prompt* is more effective on KGC tasks. This could be explained by the fact that, to maintain similar trainable parameters, non-layer-wised *Soft Prompt* requires much longer prompt vector sequences at the input, while self-attention modules are often ineffective when handling long sequences (Zaheer et al., 2020).

Local Adversarial Regularization Table 3 Lines 5 to 8 show the ablation for adversarial regularization. Line 5 shows CSProm-KG without LAR falls behind the full CSProm-KG model by 0.041 MRR, indicating the important of LAR. From Lines 6, 7, 8, we investigate the importance of LAR entity source. We observe that CSProm-KG with LAR entities that share common keywords (in name or description) outperforms the one with random LAR entities, indicating the importance of selecting appropriate adversarial examples.

PLM Training Strategy We empirically verify the effect of freezing PLM in CSProm-KG. Table 3 Lines 9 - 12 show the performance of CSProm-KG with different level of parameter frozen. In general, the more trainable parameters in CSProm-KG, the poorer CSProm-KG performs. CSProm-KG w/ fully fine-tuned drops significantly, by 0.138 MRR (Line 12). We further show the changes of performance as we increase the number of trainable parameters of the PLMs in Figure 5. We freeze the PLM parameters starting from bottom layers (orange) and starting from top layers (blue). Both experiments suggest that the performance of CSProm-KG remains nearly unchanged until the freezing operations are applied to the last few layers. As most of the layers frozen, the performance of CSProm-KG grows dramatically. Interestingly, we find freezing parameters from bottom layers performs slightly better than from top layers. This could be because lower layers in BERT could capture low-level semantics (e.g., phrase features) and this information is more beneficial to the KGC task. In summary, the frozen PLM prevents CSProm-KG from over-fitting the KG textual information, and therefore allows CSProm-KG to achieve substantial improvements in KGC tasks.

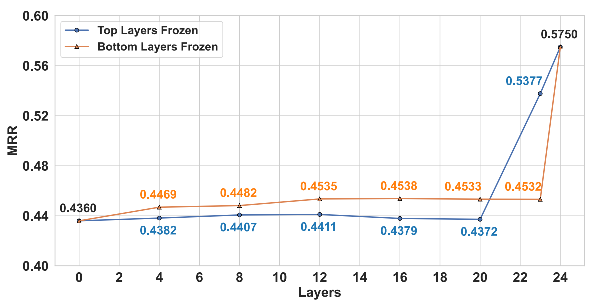

Figure 5: The effect of parameter frozen on WN18RR. Orange and Blue lines indicate the performance when freezing parameters from bottom and top layers in PLM. The X-axis shows the number of frozen layers and the Y-axis shows the corresponding performance MRR.

Ensemble Model CSProm-KG has successfully combined both textual and structure knowledge for KGC using *Conditional Soft Prompt*. To show the effectiveness of this design choice, we adopt a straightforward full-sized bagging strategy to combine the prediction from a graph-based KGC model and a PLM-based KGC model. We separately run the ConvE model and BERT model used in CSProm-KG (i.e., same configuration for fair comparsion) and use the averaged results from both models. Table 3 Line 13 shows that this ensemble model is far less effective than CSProm-KG. We believe this is because the ensemble model cannot deeply fuse structural and textual information like our proposed conditional soft-prompt.

Prompt Length As shown in Table 4, we conduct extensive studies to examine the impact of prompt length for CSProm-KG. We observe that as the prompt length increases, there is a proportional rise in both memory and computational requirements. However, the corresponding improvement in performance is marginal. Moreover, a further increase in prompt length presents considerable challenges in training the prompt model, leading to a decline in performance.

length	MRR	H@1	H@3	H@10	T/EP	#Trainable
10	.575	.522	.596	.678	12min	28M
50	.577	.523	.601	.680	23min	104M
100	.434	.419	.450	.483	41min	200M

Table 4: Prompt length study of CSProm-KG on WN18RR

Furthermore, we conduct an investigation involving the utilization of a fully fine-tuned BERT to represent the input head entity and relation, without using prompt learning or a graph-based models. However, we find instability during the training process of this model, and consequently, the resulting model achieve very low performance compared to the results reported above.

4.3 Model Efficiency

Table 5 shows the model efficiency for CSProm-KG and other PLM-based KGC methods on a single RTXA6000 GPU. CSProm-KG requires much less training and evaluation time. Compared with KG-BERT (Yao et al., 2019) and StAR (Wang et al., 2021a), CSProm-KG is 10x faster in training and 100x faster in evaluation. This is because both

Method	PLM	#Total	#Trainable	T/Ep	Inf
KG-BERT	RoBERTa base	125M	125M	79m	954m
	RoBERTa large	355M	355M	142m	2928m
StAR	RoBERTa base	125M	125M	42m	27m
	RoBERTa large	355M	355M	103m	34m
GenKGC	BART base	140M	140M	5m	88m
	BART large	400M	400M	11m	104m
KG-S2S	T5 base	222M	222M	10m	81m
	T5 large	737M	737M	27m	115m
CSProm-KG	BERT base	126M	17M	4m	0.1m
	BERT large	363M	28M	12m	0.2m

Table 5: Comparisons of model efficiency for CSProm-KG and other PLM-based methods on WN18RR with FP32 precision. #Total and #Trainable denotes the total and trainable parameters, respectively. T/Ep and Inf denotes the training time per epoch and inference time.

KG-BERT and StAR require the PLM outputs to represent all KG entities, which introduces significant computational cost. In contrast, CSProm-KG only applies BERT to represent the input queries and directly uses entity embedding matrix to compute entity ranking. We also compare CSProm-KG with GenKGC (Xie et al., 2022) and KG-S2S (Chen et al., 2022), recently proposed PLM-based Sequence-to-Sequence KGC models. They directly generate the correct entity names and does not require to use the outputs of PLMs to represent large-scale KG entities. However, it has to maintain a huge search space for the entity names during inference and becomes much slower than CSProm-KG (e.g., 0.2m *vs.* 104m and 115m). In summary, CSProm-KG maintains higher-level efficiency (as well as performance) compared to other PLM-based KGC methods with similar model size.

4.4 Flexibility to Graph-based KGC models

As we discussed in Sec. 3.3, CSProm-KG is able to incorporate other graph-based KGC methods. To verify the flexibility of CSProm-KG, we replace the ConvE with another two popular graph-based KGC methods: TransE and DistMult. As shown in Table 6, CSProm-KG can always improve the KGC task performance after integrating with TransE, DistMult and ConvE. This indicates that CSProm-KG successfully incorporate the text information into these graph-based KGC models. In particular, CSProm-KG with TransE achieves a 2x improvement on MRR (from .243 to .499) and 10x improvement on H@1 (from .043 to .462). In short, CSProm-KG is capable of fusing its textual knowledge with the structural knowledge provided by various of graph-based KGC models.

Methods	MRR	H@1	H@3	H@10
TransE	.243	.043	.441	.532
+ CSProm-KG	.499$_{\uparrow.256}$	.462$_{\uparrow.419}$	.515$_{\uparrow.074}$	.569$_{\uparrow.037}$
DistMult	.444	.412	.470	.504
+ CSProm-KG	.543$_{\uparrow.099}$	.494$_{\uparrow.082}$	.562$_{\uparrow.092}$	.639$_{\uparrow.135}$
ConvE	.456	.419	.470	.531
+ CSProm-KG	.575$_{\uparrow.119}$	.522$_{\uparrow.103}$	.596$_{\uparrow.126}$	.678$_{\uparrow.147}$

Table 6: WN18RR results of CSProm-KG with different graph-based methods.

4.5 Case Study

In this section, we showcase how *Conditional Soft Prompt* could prevent CSProm-KG from over-fitting to textual information. Table 7 lists the top two entities ranked by CSProm-KG and CSProm-KG w/o *Conditional Soft Prompt* (i.e., CSProm-KG w/ FT in Table 3). In the first case, CSProm-KG produces two different occupations that are relevant to the *whaler* in the KG Query, whilst CSProm-KG w/o *Conditional Soft Prompt* ranks two sea animal names as the outputs. This could be caused by the surface keywords *seaman* and *ship* in the KG Query. In the second case, the expected entity should be an award for the band *Queen*. CSProm-KG successful pick up the correct answer from many award entities using the existing KG structures, while CSProm-KG w/o *Conditional Soft Prompt* confuses in those candidates which are textually similar and unable to rank the ground-truth entity into top-2. In summary, CSProm-KG maintains a balance between textual and structural knowledge, while CSProm-KG w/o *Conditional Soft Prompt* often focuses too much on the textual information in the KG Query.

KG Query: whaler [a seaman who works on a ship that hunts whales] | hypernym
CSProm-KG:
*A*1*: tar [a man who serves as a sailor]
A2: crewman [a member of a flight crew]
CSProm-KG w/o *Conditional Soft Prompt*:
A1: pelagic bird [bird of the open seas]
A2: mackerel [any of various fishes of the family scombridae]
KG Query: Queen [queen are a british rock band formed in london in 1970 ...] | award
CSProm-KG:
*A*1*: Grammy Award for Best Pop Performance by Group with Vocal [...]
*A*2: MTV Video Music Award for Best Visual Effects [*the following is* ...]
CSProm-KG w/o *Conditional Soft Prompt*:
*A*1: Grammy Award for Best Music Film [*the grammy award for best* ...]
*A*2: Razzie Award for Worst Original Song [*the razzie award for worst*...]

Table 7: Case study of CSProm-KG. Texts in brackets are entity descriptions. $*$ denotes ground-truth entity.

5 Conclusion and Future Work

In this paper, we propose CSProm-KG, a PLM-based KGC model that effectively fuses the KG structural knowledge and avoids over-fitting towards textual information. The key innovation of CSProm-KG is the *Conditional Soft Prompt* that connects between a graph-based KGC models and a frozen PLM avoiding the textual over-fitting issue. We conduct experiments on five popular KGC benchmarks in SKGC and TKGC settings and the results show that CSProm-KG outperforms several strong graph-based and PLM-based KGC models. We also show the efficiency and flexibility of CSProm-KG. For future work, we plan to adapt our method to other relevant knowledge-intensive downstream tasks, such as fact checking and open-ended question answering.

6 Limitations

CSProm-KG successfully integrates both graph-based and textual representations in the KGC task, achieving substantial performance and efficiency improvement. However, similar to other PLM-based methods, this comes at the cost of increased computational resources (v.s. graph-based KGC models). In addition, we find that CSProm-KG may occasionally collapse on small KGC benchmarks (e.g. WN18RR) under specific random seeds. This is probably due to the nature of *Soft Prompts*, which involve much smaller number of trainable parameters, compared to fine-tuned models. However, we never see similar phenomena when training CSProm-KG in the large KGC benchmarks (e.g., Wikidata5M). We plan to solve these issues for CSProm-KG as future work.

Acknowledgement

We thank the anonymous reviewers for their insightful suggestions to improve this paper. This research / project is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative and A*STAR SERC Central Research Fund (UIBR). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1995–2010. ACM.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data. *Harvard Dataverse*.

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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. 2022. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 4005–4017. International Committee on Computational Linguistics.

Louis Clouâtre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. 2021. MLMLM: link prediction with mean likelihood masked language model. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4321–4331. Association for Computational Linguistics.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2001–2011. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4816–4821. Association for Computational Linguistics.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3988–3995. AAAI Press.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhen Han, Gengyuan Zhang, Yunpu Ma, and Volker Tresp. 2021. Time-dependent entity embedding is not all you need: A re-evaluation of temporal knowledge graph completion models under a unified framework. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8104–8118. Association for Computational Linguistics.

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING*

2020, Barcelona, Spain (Online), December 8-13, 2020, pages 1737–1743. International Committee on Computational Linguistics.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1771–1776. ACM.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Jia Li, Yuyuan Zhao, Zhi Jin, Ge Li, Tao Shen, Zhengwei Tao, and Chongyang Tao. 2022. Sk2: Integrating implicit sentiment knowledge and explicit syntax knowledge for aspect-based sentiment analysis. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1114–1123, New York, NY, USA. Association for Computing Machinery.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. *CoRR*, abs/2203.10321.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *8th International Conference on Learning Representations,* *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2153–2162. Association for Computational Linguistics.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021a. Structure-augmented text representation learning for efficient knowledge graph completion. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1737–1748. ACM / IW3C2.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022. PromDA: Prompt-based data augmentation for low-resource NLU tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4255, Dublin, Ireland. Association for Computational Linguistics.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1112–1119. AAAI Press.

Zeguan Xiao, Jiarun Wu, Qingliang Chen, and Congjian Deng. 2021. BERT4GCN: Using BERT intermediate layers to augment GCN for aspect-based sentiment classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9193–9200, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. 2022. From discrimination to generation: Knowledge graph completion with generative transformer. *CoRR*, abs/2202.02113.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. 2019. Temporal knowledge graph embedding model based on additive time series decomposition. *CoRR*, abs/1911.07893.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020. Tero: A time-aware knowledge graph embedding via temporal rotation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1583–1593. International Committee on Computational Linguistics.

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 *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

A Dataset

We use SKGC datasets released from (Yao et al., 2019) and TKGC datasets from (García-Durán et al., 2018). We follow the original split in our experiments. Table 8 shows the statistics of the datasets. All of these datasets are open-source English-written sources without any offensive content. They are introduced only for research use.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	|Train|	|Valid|	|Test|
SKGC					
WN18RR	40,943	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,466
Wikidata5M	4,594,485	822	20,614,279	5,163	5,133
TKGC					
ICEWS14	6,869	230	72,826	8,941	8,963
ICEWS05-15	68,544	358	189,635	1,004	2,158

Table 8: Statistics of the Datasets.

B Hyperparameters

Hyperparameters are selected with grid search on the validation set. The optimal hyperparameters are presented in Table 9

Dataset	η	$\mathcal{B}$	$\mathcal{P}_l$	α
WN18RR	5e-4	128	10	0.1
FB15K-237	5e-4	128	10	0.1
Wikidata5M	1e-4	450	5	0.0
ICEWS14	5e-4	384	5	0.1
ICEWS05-15	5e-4	384	5	0.0

Table 9: Optimal hyperparameters.

C Baseline Methods

CSProm-KG is compared against a variety of state-of-the-art baseline methods on SKGC and TKGC tasks. For SKGC, we include popular graph-based methods, i.e. TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), RotatE (Sun et al., 2019) and CompGCN (Vashishth et al., 2020). We also compare CSProm-KG against several competitive PLM-based methods, i.e. KG-BERT (Yao et al., 2019), MTL-KGC (Kim et al., 2020), StAR (Wang et al., 2021a), MLMLM (Clouâtre et al., 2021), KEPLER (Wang et al., 2021b), GenKGC (Xie et al., 2022), KGT5 (Saxena et al., 2022) and KG-S2S (Chen et al., 2022). For TKGC, we compare CSProm-KG with graph-based TKGC baselines, including: TTransE (Leblay and Chekol, 2018), HyTE (Dasgupta et al., 2018), ATiSE (Xu et al., 2019), DE-SimplE (Goel et al., 2020), Tero (Xu et al., 2020), TComplEx (Lacroix et al., 2020), TNTComplEx (Lacroix et al., 2020), T+TransE (Han et al., 2021), T+SimplE (Han et al., 2021). PLM-based baselines for TKGC includes KG-S2S (Chen et al., 2022)

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 6

☒ A2. Did you discuss any potential risks of your work?
The potential risk of this line of work has already been discussed in previous research and our base methods (Bert).

☑ A3. Do the abstract and introduction summarize the paper's main claims?
both of abstract and introduction (section 1) do

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

section 4

☑ B1. Did you cite the creators of artifacts you used?
data citation: 4.1 Dataset, model citation: 4.1 Implementation Details

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
data: Appendix A, model: 4.1 Implementation Details.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Appendix A

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix A

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4.3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4.1 Implementation details and Appendix B

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4.2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.1, Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.