Enhancing Contextual Word Representations Using Embedding of Neighboring Entities in Knowledge Graphs

Ryoko Tokuhisa **Keisuke Kawano** **Akihiro Nakamura** **Satoshi Koide**
Toyota Central R&D Labs., Inc.
{tokuhisa, kawano, a-nakamura, koide}@mosk.tytlabs.co.jp

Abstract

Pre-trained language models (PLMs) such as BERT and RoBERTa have dramatically improved the performance of various natural language processing tasks. Although these models are trained on large amounts of raw text, they have no explicit grounding in real-world entities. Knowledge graphs (KGs) are manually annotated with factual knowledge and store the relations between nodes corresponding to entities as labeled edges. This paper proposes a mechanism called *KG-attention*, which integrates the structure of a KG into recent PLM architectures. Unlike the existing PLM+KG integration methods, KG-attention generalizes the embeddings of neighboring entities using the relation embeddings; accordingly, it can handle relations between unconnected entities in the KG. Experimental results demonstrated that our method achieved significant improvements in a relation classification task, an entity typing task, and several language comprehension tasks.

1 Introduction

Pre-trained language models (PLMs) have significantly improved the performance of various natural language processing (NLP) tasks (Devlin et al., 2019; Liu et al., 2019). Although these models are trained on large amounts of raw text, they have no explicit grounding in real-world entities. Figure 1 shows a causal reasoning task on which PLM fails. The performance can be improved by representing factual knowledge as various relations between entities, e.g., "engine is *part of* a car", in the PLM.

Knowledge graphs (KGs) are manually annotated with factual knowledge and store the relations between the nodes corresponding to entities as labeled edges (e.g., *HasA, IsA*, and *PartOf*) (Miller, 1995; Bollacker et al., 2008; Speer et al., 2016). Although the various relations recorded in KGs can potentially improve the performance of PLMs, KGs are *graphs* and structurally unsuitable for direct incorporation into PLMs. Some of the more suitable forms of factual knowledge are pre-trained KG embeddings (Bordes et al., 2013; Sun et al., 2019; Nguyen, 2020; Chao et al., 2021). KG embeddings embed two entities registered in a KG such that the corresponding embedding vectors are at a specific relative position depending on the relation between the entities. The embedding vectors corresponding to the entities are called *KG entity embeddings* and the specific relative positions are called *relation embeddings*. In addition to avoiding the direct use of structural encoding of KGs, KG embeddings can be generalized; that is, entities having a specific relation to a query entity can be estimated even when their relation is not actually recorded in the KGs (Bordes et al., 2013). For example, if there is no *PartOf* connection between `car` and `engine` in the KG, the embedding of `engine` can be estimated from the embedding of `car` and the relation embedding of *PartOf*.

One remaining problem is the incorporation of KG entity embeddings and relation embeddings

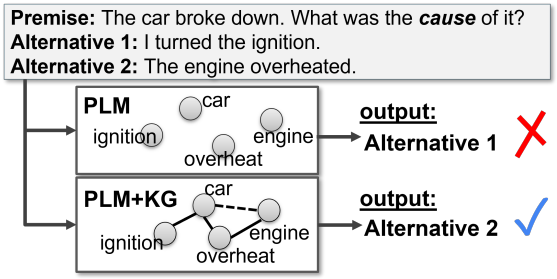

Figure 1: Example of the outputs of Choice of Plausible Alternatives (COPA), a well-known causal reasoning task. "PLM" represents the causal reasoning model using $\text{BERT}_{\text{LARGE}}$ and "PLM+KG" represents the model using $\text{BERT}_{\text{LARGE}}$ and Knowledge Graph (WordNet). Solid and dotted lines indicate the relations between entities connected in KG and entities not connected in KG, respectively. The relation between unconnected entities were estimated from KG embeddings in our proposed method.

Proceedings of the 29th International Conference on Computational Linguistics, pages 3175–3186
October 12–17, 2022.

into PLMs. Recently, several PLM and KG integration methods have been proposed (Zhang et al., 2019; Yang et al., 2019; Lin et al., 2019; Wang et al., 2019; Sun et al., 2020; Wang et al., 2021). For example, Zhang et al. (2019) concatenated embedding vectors obtained from PLM word embeddings and KG entity embeddings for downstream tasks. Wang et al. (2021) jointly trained a masked language model (MLM) and a KG embedding model to align factual knowledge and language representation in the same semantic space. Although KG entity embeddings have been utilized, how the informative relation embeddings should be incorporated into PLMs is not obvious. Actually, the relation embeddings have been ignored, while KG entity embeddings have been well utilized.

Herein, we propose *Integrating PLMs and KGs through Attention Mechanisms (ILKA)*, which handles both the KG entity embeddings and the relation embeddings through our attention mechanism (*KG-attention*). A key feature of KG-attention is that it fully utilizes the generalization ability of pre-trained KG embeddings, meaning that KG-attention can handle relations between entities not connected in the original KG.

Our contributions are summarized below.

- We propose ILKA with KG-attention that incorporates both the KG entity embeddings and relation embeddings with the embedding vectors obtained from PLMs in a consistent manner.
- We experimentally demonstrate that ILKA achieves improvements in a relation classification task, an entity typing task, and several language comprehension tasks.

2 Related Works

Pre-trained language models PLMs such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2020) have achieved great success in many NLP tasks. As a result, pre-training language models and fine-tuning them in downstream tasks has become a new standard in NLP. However, PLMs do not explicitly learn the relations between entities and may not properly identify factual knowledge (Peters et al., 2019; Wang et al., 2021).

Knowledge graph embedding KGs such as WordNet (Miller, 1995) and ConceptNet (Speer et al., 2016) have also become important resources in many NLP tasks. In general, a KG is a collection of relational facts often represented as a triplet $(h, r, t) \in \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ (e.g., (`car`, `IsA`, `vehicle`)), where $\mathcal{V}$ is the vocabulary and $\mathcal{R}$ is the set of relations. h is a *head entity*, r is a *relation*, and t is a *tail entity*.

	Approximate formula
TransE (Bordes et al., 2013)	$\mathbf{v}(t) \approx \mathbf{v}(h) + \mathbf{v}(r)$
TransR (Lin et al., 2015)	$\mathbf{v}_r(t) \approx \mathbf{v}_r(h) + \mathbf{v}(r)$
RotatE (Sun et al., 2019)	$\mathbf{v}(t) \approx \mathbf{v}(h) \circ \mathbf{v}(r)$
PairRE (Chao et al., 2021)	$\mathbf{v}(t) \approx \mathbf{v}(h) \circ \mathbf{v}^H(r) \oslash \mathbf{v}^T(r)$

Table 1: List of approximate formulas of KG embedding models. $\mathbf{v}_r(\cdot)$ is the embedding function specific to the relation r. $\circ$ is the Hadamard product. $\oslash$ denotes the element-wise division. $\mathbf{v}^H(r)$ and $\mathbf{v}^T(r)$ are embeddings of the relation r for head and tail entities, respectively.

Several learning methods have been proposed for embedding KGs into a continuous vector space while preserving the relational structure (Nickel et al., 2011; Bordes et al., 2013; Schlichtkrull et al., 2018; Lin et al., 2015; Sun et al., 2019). In most KG embedding models, the triplet (h, r, t) is embedded to satisfy the following:

$$\mathbf{v}_{\mathrm{KG}}(t) \approx \phi_h(\mathbf{v}_{\mathrm{KG}}(h), \mathbf{v}_{\mathrm{r}}(r)), \qquad (1)$$

where $\mathbf{v}_{\mathrm{KG}} : \mathcal{V} \mapsto \mathbb{R}^{d_{\mathrm{KG}}}$ is the embedding function, and $\mathbf{v}_{\mathrm{r}}(r) \in \mathbb{R}^{d_{\mathrm{KG}}}$ is the embedding vector corresponding to the relation r. ϕ_h denotes the relationship between the embedding vectors; for example, $\phi_h(\mathbf{v}_{\mathrm{KG}}(h), \mathbf{v}_{\mathrm{r}}(r)) = \mathbf{v}_{\mathrm{KG}}(h) + \mathbf{v}_{\mathrm{r}}(r)$ in **TransE** (Bordes et al., 2013). Table 1 gives the approximation formulas of typical KG embedding models.

One benefit of KG embedding is that entities with a specific relation to the query entity can be retrieved even when the relation between the entities is not connected in the KG (Bordes et al., 2013). For example, in Figure 2, entities e connecting the "IsA" relation with `car` can be estimated by extracting the entities satisfying $\mathbf{v}_{\mathrm{KG}}(e) \approx \phi_h(\mathbf{v}_{\mathrm{KG}}(\mathtt{car}), \mathbf{v}_{\mathrm{r}}(\mathtt{IsA}))$. $\mathbf{v}_{\mathrm{KG}}(e)$ encodes not only words having the "IsA" relation with `car` in the KG (i.e., *automobile*, *vehicle*), but also words not having the "IsA" relation with `car` but having similar meanings to *automobile* and *vehicle* (i.e., *motorcar* in Figure 2). In this paper, the vector $\phi_h(\mathbf{v}_{\mathrm{KG}}(w), \mathbf{v}_{\mathrm{r}}(r))$ is called the ***KG-neighbor embeddings*** of entity w and the en-

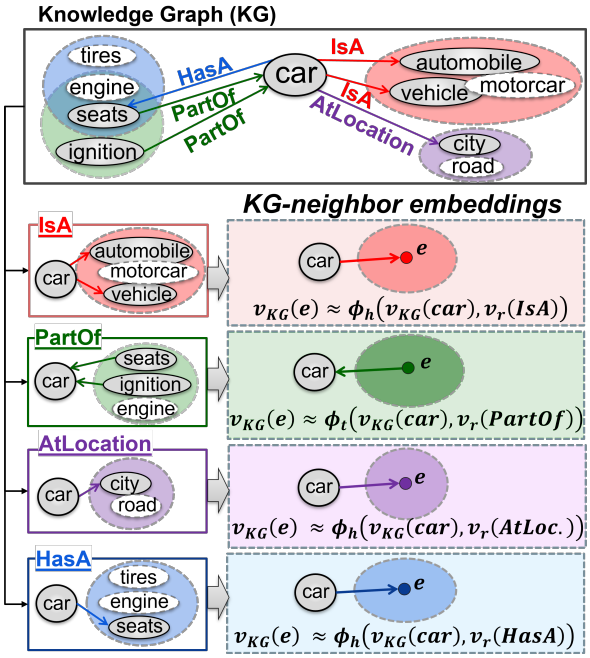

Figure 2: Example of `car` and its neighboring entities in KG. The words in the solid circles (e.g., *automobile*, *city*) have a relation defined in KG whereas the words in the dotted circles (e.g., *motorcar*, *road*) indicate words whose relation is not defined in KG.

tities corresponding to the KG-neighbor embeddings (i.e., e) are called ***KG-neighbors***. Note that KG-neighbors may be unconnected to entity w in the original KG. For some KG embedding methods such as TransE, we can also consider KG-neighbor embeddings as $\phi_t(\mathbf{v}_{\mathrm{KG}}(w), \mathbf{v}_{\mathrm{r}}(r))$, where w corresponds to a tail entity. In Figure 2, the KG-neighbor embeddings in the "PartOf" relation with `car` (i.e., *seats*, *ignition*, *engine*) are represented by $\phi_t(\mathbf{v}_{\mathrm{KG}}(\texttt{car}), \mathbf{v}_{\mathrm{r}}(\texttt{PartOf}))$, such as $\phi_t(\mathbf{v}_{\mathrm{KG}}(\texttt{car}), \mathbf{v}_{\mathrm{r}}(\texttt{PartOf})) = \mathbf{v}_{\mathrm{KG}}(\texttt{car}) - \mathbf{v}_{\mathrm{r}}(\texttt{PartOf})$ in **TransE**. In the following, ϕ refers to either ϕ_h or ϕ_t.

Integrating PLMs and KGs Several recent studies have integrated KGs with PLM. In some studies, the KGs and PLM were combined to perform specific downstream tasks (Yu et al., 2018; Wang and Jiang, 2019; Yasunaga et al., 2021). Yasunaga et al. (2021) proposed an end-to-end question answering (QA) model that leverages PLM and KG by scoring the relevance of KG nodes conditional on a given QA context. These downstream task-focused methods are effective and robust for the given task, but they do not enhance PLM for general purposes.

In another approach, PLM and KG are integrated at the model level (Zhang et al., 2019; Peters et al., 2019; Yang et al., 2019; Lin et al., 2019; Wang et al., 2019; Sun et al., 2020; Wang et al., 2021). Zhang et al. (2019) concatenated PLM word embeddings and KG entity embeddings and Sun et al. (2020) trained an MLM by concatenating the entity sequences obtained from a KG and input word sequences. However, these PLM+KG methods are limited in that they do not effectively employ relation embeddings. A relation embedding encodes the semantic direction of the predefined relations in KGs and plays an important role in representing neighboring entities in KG and generalizing relations between entities not included in the KG. The existing methods either do not consider relation embeddings (Zhang et al., 2019) or cannot handle the relations between entities not connected in the KG (Sun et al., 2020; Wang et al., 2021).

3 Methodology

3.1 Main idea: KG-attention

When KG embedding is integrated into PLMs, the performance can be improved by representing factual knowledge as various relations (e.g., *IsA*, *HasA*, *AtLocation*). Below, we propose ***KG-attention*** that consistently integrates the embedding of entities and relations into an attention mechanism. The benefits of KG-attention are twofold: (1) KG-attention can explicitly assign attention between a query entity and entities in the input sentence that are related through KG embeddings; (2) KG-attention can handle KG-neighbors even when the relation between entities is not connected in the KG.

Figure 3 shows an application example of KG-attention, which explores the attention between an input sentence and KG-neighbors such as car_{IsA}, car_{HasA}, and $car_{\mathrm{AtLocation}}$. For example, the attention between *engine* and car_{HasA} represents the knowledge *"a car has an engine."* When integrating a KG embedding with a PLM, how to incorporate the relation represented by ϕ into a PLM comprising a transformer is not always evident. We solve this difficulty by carefully designing the query, key, and value of the attention. Given an input sequence $\mathbf{w} = [w^{(1)}, \ldots, w^{(L)}]$, where w denotes a subword and L denotes the length of the sequence, KG-attention integrates ϕ into attention mechanism as follows:

$$\mathbf{Q} = \{\mathbf{Q}^{(i)}\}_{i\in\{1,\ldots,L\}}, \tag{2}$$

$$\mathbf{K} = \mathbf{V} = \{\mathbf{K}^{(i,j)}\}_{i\in\{1,\ldots,L\},\, j\in\{0,\ldots,|\mathcal{R}|\}} \tag{3}$$

$$\mathbf{Q}^{(i)} = \mathbf{v}_{\mathrm{KG}}(w^{(i)}) \oplus \mathbf{v}_{\mathrm{PLM}}(w^{(i)}) \tag{4}$$

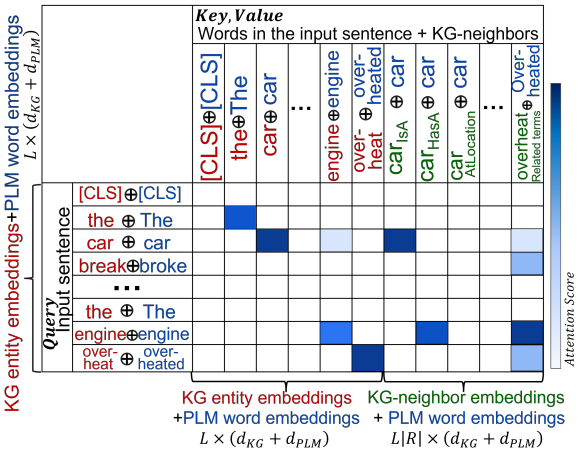

Figure 3: Example of KG-attention applied to sentence classification. The input sentence is "Premise + Alternative 2" in Figure 1.

$$\mathbf{K}^{(i,j)} = \begin{cases} \mathbf{v}_{\mathrm{KG}}(w^{(i)}) \oplus \mathbf{v}_{\mathrm{PLM}}(w^{(i)}) & \text{if } j = 0 \\ \phi(\mathbf{v}_{\mathrm{KG}}(w^{(i)}), \mathbf{v}_{\mathrm{r}}(r^{(j)})) \oplus \mathbf{v}_{\mathrm{PLM}}(w^{(i)}) & \\ \qquad \text{if } j \in \{1, \ldots, |\mathcal{R}|\} \end{cases} \tag{5}$$

where $\mathbf{Q}^{(i)}$ and $\mathbf{K}^{(i,j)}$ are $d_{\mathrm{KG}} + d_{\mathrm{PLM}}$ dimensional vectors, $\mathbf{Q}$ is a set of L vectors, and $\mathbf{K}(= \mathbf{V})$ is a set of $L \times (|\mathcal{R}| + 1)$ vectors. $\mathbf{v}_{\mathrm{KG}}(\cdot) \in \mathbb{R}^{d_{\mathrm{KG}}}$ is the KG entity embedding and $\mathbf{v}_{\mathrm{PLM}}(\cdot) \in \mathbb{R}^{d_{\mathrm{PLM}}}$ is the PLM embeddings. d_{KG} and d_{PLM} are the dimensions of the KG and PLM embeddings, respectively. $\oplus$ is the concatenation operation, and $r^{(j)}$ represents the j-th relation.

Why and how does KG-attention treat semantically related tokens defined in KGs? In general, the attention mechanism seeks key tokens, which are similar to query tokens. Accordingly, the KG-attention aims to seek for key tokens that are located at the KG-neighbors of the query token. For example, $\mathbf{K}^{(i,j)}$ (for $w^{(i)} = car$) in Figure 3 is constructed from KG-neighbor embeddings such as car_{IsA}, car_{HasA}, and $car_{\mathrm{AtLocation}}$. These KG embeddings are concatenated with the PLM embedding of car and then used as the keys and values of the attention. To obtain $\mathbf{Q}^{(i)}$ (for $w^{(i)} = car$), we can simply compute and concatenate the KG and PLM embeddings of car. As KG-neighbor embeddings are obtained from KG entity embeddings of w and $r^{(j)}$ by the approximation function ϕ, they can be regarded as the embeddings of a **representative entity** having relation r with an entity w. If *engine* and the representative entity of car_{HasA} are similar, KG-attention can obtain the attention between them even if *engine* and *car* are unconnected in the original KG. Thus, KG-attention achieves flexible attention by exploiting generalized KG-neighbor embeddings.

Model	PLM +KG	Relation between entities connected	Relation between entities not connected
ERNIE	✓		
KnowBERT	✓	✓	
CoLAKE	✓	✓	
KEPLER	✓	✓	
ILKA (ours)	✓	✓	✓

Table 2: Comparison of ILKA with existing methods. "Relation between entities connected" indicates that the model only handle the relation between entities connected in the KG (e.g., *automobile* in Figure 2); "Relation between entities not connected" indicates that the model handle relations between entities not connected in KG (e.g., *motorcar* in Figure 2).

3.2 ILKA

This section describes our proposed model **ILKA** incorporating KG-attention. As shown in Figure 4, ILKA consists of the following three modules.

(1) PLM and KG Embeddings: The input sentences are fed to the PLM and pre-trained KG embedding model to obtain the PLM word embeddings and KG entity embeddings. The KG embedding model additionally obtains the KG relation embeddings of all relations defined in the KG.

(2) KG-attention: The KG-attention mechanism integrates the PLM word embeddings, KG entity embeddings, and KG relation embeddings.

(3) Classifier: Classification is performed by a transformer encoder-based classifier with self-attention. KG-attention, which consistently incorporates the relational information of the KG and PLM embeddings, can be regarded as the first layer of the transformer encoder-based classification model.

Table 2 compares the features of our method with those of four existing PLM+KG methods. ILKA approximates KG-neighbor embeddings based on KG relation embeddings and can consider the attention between the input sentence and KG-neighbors even if the relation between the entities is not connected in the KG. Among the compared methods, only ILKA achieves this functionality.

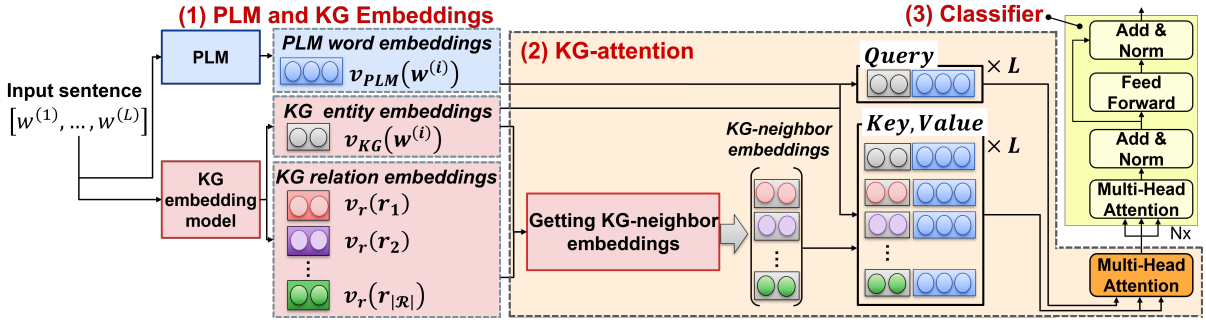

Figure 4: Overview of ILKA, which comprises three steps: (1) Obtain the PLM and KG embeddings; (2) Apply KG-attention to integrate the PLM word embeddings, KG entity embeddings, and KG relation embeddings; (3) Perform classification using a transformer encoder-based model.

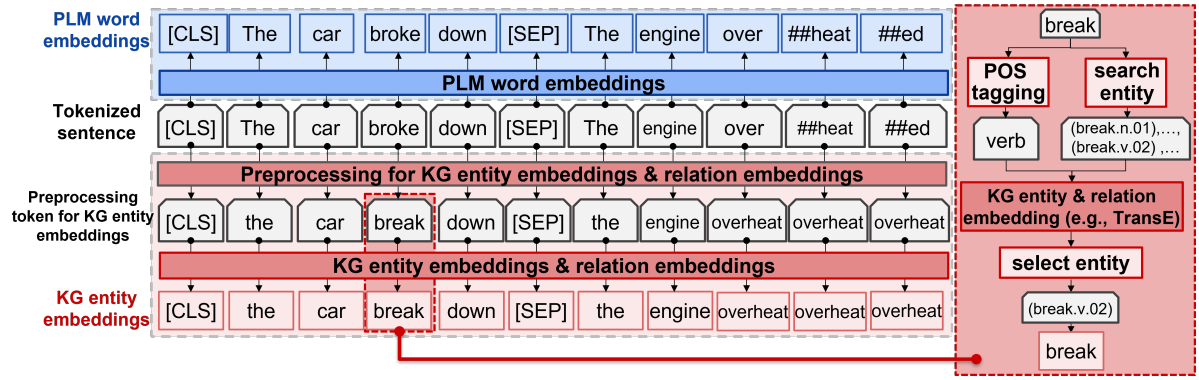

Figure 5: Procedure for obtaining the word embeddings shown in Figure 4 (1). The blue and red shading highlight the PLM word embeddings and KG entity embeddings, respectively.

3.3 Implementation

PLM word embeddings (shaded blue in Figure 5): Transformer-based PLMs often require a tokenization step to solve the out-of-vocabulary problem. For example, the sentence "The engine **overheated**" can be divided into the subword sequence "The engine **over ##heat ##ed**." PLMs give the word embeddings of each subword unit. We use existing PLMs for PLM word embeddings.

KG entity embeddings (shaded red in Figure 5): Unlike PLMs, KG entities are registered as word or phrase units. To obtain KG entity embeddings, the tokenized sentences are preprocessed as follows:

(1) After reverting the subwords to the original words, the original word is duplicated n times, where n is the number of subwords (e.g., [over, ##heat, ##ed] → [overheated, overheated, overheated]).

(2) The conjugated words are returned to their original form (e.g., broke → break, overheated → overheat). Uppercase letters are made lowercase (e.g., The → the).

In typical KGs, words with multiple meanings are registered as different entities. ILKA morphologically analyzes the input sentence to find the part-of-speech (POS), and then selects an entity with the same POS among the registered entities[1]. Among various registered entities, this process allows the selection of words with similar usage to those in the input sentences. If no embedding vector is obtained from the KG embedding model, then a zero vector is used. A zero vector is also allocated to special tokens such as [CLS], [SEP], and [PAD].

KG-attention Here we describe three implementations of KG-attention on TransE as an example.

1) ILKA-head: TransE embeds the head and tail entities as $\mathbf{v}_{\mathrm{KG}}(t) \approx \mathbf{v}_{\mathrm{KG}}(h) + \mathbf{v}_{\mathrm{r}}(r)$. Our KG-attention is designed to integrate the *head* → *tail* relation with the PLM embeddings as follows:

$$\mathbf{K}_{\mathrm{head}}^{(i,j)} = (\mathbf{v}_{\mathrm{KG}}(w^{(i)}) + \mathbf{v}_{\mathrm{r}}(r^{(j)})) \oplus \mathbf{v}_{\mathrm{PLM}}(w^{(i)}). \quad (6)$$

[1]If multiple entities have the same POS as the input word, the first entity is used. In contrast, if no entity has the same POS as the input word, an entity with a different POS is used.

2) ILKA-tail: A similar integration of the *tail* $\rightarrow$ *head* relation with the PLM embeddings is given as

$$\mathbf{K}_{\text{tail}}^{(i,j)} = (\mathbf{v}_{\text{KG}}(w^{(i)}) - \mathbf{v}_{\text{r}}(r^{(j)})) \oplus \mathbf{v}_{\text{PLM}}(w^{(i)}). \quad (7)$$

3) ILKA-both: The two relations above can be combined by concatenating the head and tail vectors from $\mathbf{v}_{\text{KG}}(w^{(i)})$:

$$\mathbf{K}_{\text{both}}^{(i,j)} = \begin{cases} \mathbf{K}_{\text{head}}^{(i,j)} \text{ if } j \in \{1, \ldots, |\mathcal{R}|\} \\ \mathbf{K}_{\text{tail}}^{(i,j-|\mathcal{R}|)} \text{ if } j \in \{|\mathcal{R}|+1, \ldots, 2|\mathcal{R}|\}. \end{cases} \quad (8)$$

Note that $\mathbf{K}^{(i,0)} = \mathbf{v}_{\text{KG}}(w^{(i)}) \oplus \mathbf{v}_{\text{PLM}}(w^{(i)})$ for all implementations and $\mathbf{K}_{\text{head}}, \mathbf{K}_{\text{tail}} \in \mathbb{R}^{L(|R|+1)\times(d_{\text{KG}}+d_{\text{PLM}})}$, while $\mathbf{K}_{\text{both}} \in \mathbb{R}^{L(2|R|+1)\times(d_{\text{KG}}+d_{\text{PLM}})}$.

Transformer encoder-based classifier For the classifier, we employed the transformer encoder model proposed by Vaswani et al. (2017), which comprises a stack of $N = 6$ identical layers. Each layer has two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

4 Experiments

4.1 Benchmark Methods

The ILKA model was competed against two PLM-only methods: **BERT** (Devlin et al., 2019), **RoBERTa** (Liu et al., 2019)), and four PLM+KG methods: **ERNIE** (Zhang et al., 2019), **KnowBERT** (Peters et al., 2019), **CoLAKE** (Sun et al., 2020) and **KEPLER** (Wang et al., 2021).

4.2 Experimental Settings

The KG-attention was implemented in PyTorch (Paszke et al., 2019). As the KG embedding model, we employed the TransE model trained on WordNet and Wikipedia. The KG was based on WordNet (WN18 (Bordes et al., 2014) containing 18 relations) or Wikipedia (DBpedia50k (Shi and Weninger, 2018) containing 351 relations). Details can be found in Appendix A. We implemented TransE using PyKEEN (Ali et al., 2021) and trained it for 1,000 epochs. Following Bordes et al. (2013), the dimension of the KG embeddings was set to $d_{\text{KG}} = 50$ (see Appendix B for details). The PLM was implemented in two settings: $\text{BERT}_{\text{BASE}}$ for comparison with the BERT-based models ERNIE and KnowBERT and $\text{RoBERTa}_{\text{BASE}}$ for comparison with the RoBERTa-based models CoLAKE and KEPLER. The PLM implementations were obtained from HuggingFace Transformers (Wolf et al., 2020). For the classifier, we employed a transformer encoder with six self-attention layers. The batch sizes were set to 32 and 4 for the models with WN18 and DBPedia50k, respectively, and the maximum sequence length of the token was 128. All experiments were conducted on a 24GB NVIDIA® TITAN RTX™ GPU.

4.3 Tasks

ILKA was evaluated on three tasks: relation classification, entity typing, and language comprehension.

Relation Classification Relation classification determines the type of relation between two entities in a text. We employed TACRED as the relation classification task[2]. TACRED contains more than 106k sentences with typed subject and object spans and relation labels across 41 relations along with a *no-relation* label. The hyperparameters and other experimental settings were set following KEPLER.

Entity Typing Entity typing is the task of classifying a given entity into a pre-defined type. This task was evaluated using OpenEntity (Choi et al., 2018)[3]. Comparisons with ERNIE, KnowBERT, CoLAKE, and KEPLER were performed on nine general entity types under the same experimental settings.

Language Comprehension We adopted the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) and Choice of Plausible Alternatives (COPA) (Roemmele et al., 2011) as language comprehension tasks. GLUE is a collection of diverse natural language comprehension tasks used in various PLM papers (Devlin et al., 2019; Liu et al., 2019). Existing PLM+KG methods have evaluated the GLUE task to determine whether their methods degenerate the performance on common NLP tasks (Zhang et al., 2019; Wang et al., 2021) (see Appendix C for details). We used the implementation and evaluation script implemented by the HuggingFace Transformers library

[2]TACRED is available from LDC. https://catalog.ldc.upenn.edu/LDC2018T24

[3]OpenEntity data can be obtained from ERNIE's github page. https://github.com/thunlp/ERNIE

Base-model	Model	P	R	F1
BERT	BERT	67.2	64.8	66.0
	ERNIE	70.0	66.1	68.0
	KnowBERT	73.5	64.1	68.5
	ILKA-head	72.6	68.9	70.7
RoBERTa	RoBERTa	70.8	69.6	70.2
	KEPLER	71.5	**72.5**	72.0
	ILKA-head	**73.9**	71.9	**72.9**

Table 3: Micro precision, recall and F1 scores on TACRED (%). The KnowBERT results shown in Table 3 were reevaluated in Wang et al. (2021).

Base-model	Model	P	R	F1
BERT	BERT	76.4	71.0	73.6
	ERNIE	78.4	72.9	75.6
	KnowBERT	77.9	71.2	74.4
	ILKA-head	75.5	74.4	75.0
RoBERTa	RoBERTa	75.1	73.4	74.3
	CoLAKE	77.0	75.7	76.4
	KEPLER	**77.8**	74.6	76.2
	ILKA-head	75.3	**78.1**	**76.7**

Table 4: Micro precision, recall and F1 scores on OpenEntity (%). The results of BERT and RoBERTa were reported by Wang et al. (2021). Other benchmark results were taken from the respective papers.

for GLUE [4]. To investigate whether our method is effective for complex language comprehension tasks using COPA, a causal inference task that requires world knowledge. In COPA, the models were given a premise and two alternatives and were required to choose the alternative with a higher causal relation to the premise. Figure 1 is an example of COPA.

In the GLUE and COPA tasks, we selected the best fine-tuning learning rate (among 5e-5, 4e-5, 3e-5, and 2e-5) on the Dev set for the BERT setting according to the BERT experiments and (among 3e-5, 2e-5, and 1e-5) for the RoBERTa setting according to the RoBERTa experiments. Our classification was essentially that of BERT's framework of "[CLS] sentenceA [SEP] sentenceB [SEP]," where [CLS] is a special token for classification purposes.

4.4 Results

Relation Classification Table 3 shows the result of TACRED. As in Wang et al. (2021), we show the KnowBERT result without entity type inputs for a fair evaluation. ILKA-head represents our model with WN18. ILKA-head achieved higher F1 scores than the existing studies using PLM and KG (BERT-based and RoBERTa-based models). The results suggest that the relation classification was accurately solved by both the KG embedding of the input sentence and the generalized embedding of neighboring entities.

Entity Typing Table 4 shows the experimental results of OpenEntity. ILKA-head based on RoBERTa achieved higher F1 scores than the existing models using PLM and KG. In the BERT-base model, ERNIE and ILKA-head obtained the highest F1 and Recall scores, respectively. In the RoBERTa-base model, ILKA-head had the highest F1 score. As our method obtained the highest Recall in both BERT-base and RoBERTa-base, we inferred that using the KG-neighbors provides a consistently high coverage of various relations.

Language Comprehension The experimental results of GLUE are shown in Table 5. ILKA-head$_{\text{BERT}}$ outperformed BERT-original and ERNIE in the following tasks: MNLI-mm, QQP, CoLA, STS-B, MRPC, and RTE. Moreover, ILKA-head$_{\text{RoBERTa}}$ outperformed RoBERTa-original, CoLAKE, and KEPLER in the following tasks: MNLI-m/mm, CoLA, and MRPC. On average, our model with a RoBERTa performed comparably to the RoBERTa-original model. The result suggests that integrating PLM and KG exerts no significant negative impact.

Table 6 presents the COPA results. ILKA outperformed the original BERT and RoBERTa in all settings. Causal reasoning is one of the tasks requiring world knowledge. ILKA worked effectively even for the large-scale PLM, suggesting that KG-attention provides an inference ability that cannot be matched by word co-occurrence alone.

5 Analysis

5.1 Ablation Study

Model Variants Table 7 presents the results of an ablation study for ILKA$_{\text{RoBERTa}}$ on the TACRED and OpenEntity tasks. In the "ILKA$_{\text{w/o-KGneighbors}}$" model, only the KG entity embeddings are provided in the input sentence (the KG-neighbor embeddings are omitted). This model is similar to ERNIE but has a slightly different network structure. "ILKA-head", "ILKA-tail", and "ILKA-both" represent different implementations of ILKA as described in subsection 3.3.

ILKA-head and ILKA-tail obtained higher F1 scores "ILKA$_{\text{w/o-KGneighbors}}$", suggesting that when

[4] https://github.com/huggingface/transformers/blob/master/examples/pytorch/text-classification/run_glue.py

Base-model	Model	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k
BERT	BERT (Devlin et al., 2019)	**84.6**/83.6	71.2	90.5	**93.5**	52.1	85.8	88.9	66.4
	ERNIE (Zhang et al., 2019)	84.0/83.2	71.2	**91.3**	**93.5**	52.3	83.2	88.2	68.8
	ILKA-head$_{\text{BERT}}$ (OURS)	83.8/**84.4**	**87.7**	90.7	92.2	**62.6**	**89.3**	**91.1**	**69.0**
RoBERTa	RoBERTa (Liu et al., 2019)	87.5/87.3	91.9	**92.8**	**94.8**	63.6	**91.2**	90.2	78.7
	CoLAKE (Sun et al., 2020)	87.4/87.2	**92.0**	92.4	94.6	63.4	90.8	90.9	77.9
	KEPLER (Wang et al., 2021)	87.2/86.5	91.7	92.4	94.5	63.6	**91.2**	89.3	**85.2**
	ILKA-head$_{\text{RoBERTa}}$ (OURS)	**87.7/87.6**	88.1	92.6	94.5	**64.4**	91.1	**92.0**	75.8

Table 5: Test results of GLUE. The number below each task denotes the number of training examples. The results of the benchmark methods were taken from the respective references. KEPLER represents the result of KEPLER-wiki, the most accurate model for GLUE in (Wang et al., 2021). The F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and the accuracy scores are reported for the other tasks.

	Model	Accuracy
BERT$_{\text{BASE}}$	BERT$_{\text{BASE}}$	68.8 ± 1.3
	ILKA-head$_{\text{BERTBASE}}$	**69.9 ± 1.1**
BERT$_{\text{LARGE}}$	BERT$_{\text{LARGE}}$	76.5 ± 2.7
	ILKA-head$_{\text{BERTLARGE}}$	**77.7 ± 1.2**
RoBERTa$_{\text{BASE}}$	RoBERTa$_{\text{BASE}}$	73.7 ± 1.3
	ILKA-head$_{\text{RoBERTaBASE}}$	**74.9 ± 2.1**
RoBERTa$_{\text{LARGE}}$	RoBERTa$_{\text{LARGE}}$	87.7 ± 0.9
	ILKA-head$_{\text{RoBERTaLARGE}}$	**88.6 ± 1.6**

Table 6: Test results on COPA. The results of BERT$_{\text{LARGE}}$ and RoBERTa$_{\text{LARGE}}$ were reported by Kavumba et al. (2019).

	TACRED			OpenEntity		
Model	P	R	F1	P	R	F1
ILKA$_{\text{w/o-KGneighbors}}$	73.7	70.8	72.2	77.8	71.0	74.3
ILKA-head	73.9	71.9	72.9	75.3	78.1	76.7
ILKA-tail	75.4	72.9	74.1	76.4	79.9	78.1
ILKA-both	76.4	70.7	73.5	74.0	73.2	73.6

Table 7: Results of the ablation study on TACRED and OpenEntity using the proposed ILKA$_{\text{RoBERTa}}$.

both the KG-neighbor and KG entity embeddings are employed, the model becomes more effective. ILKA-both, which simply concatenates the KG-neighbor embeddings of the head and tail entities, was less accurate than ILKA-head and ILKA-tail.

Knowledge Graph Variants Table 8 shows the experimental results for different KGs. ILKA-head$_{\text{WordNet}}$ outperformed ILKA-head$_{\text{Wiki}}$, possibly because the large number of relations in ILKA-head$_{\text{Wiki}}$ leads to a sparser embedding space. Nevertheless, ILKA-head$_{\text{Wiki}}$ is still comparable to benchmark methods shown in Table 3 and Table 4.

5.2 Training Runtime Comparison

Most of the existing PLM+KG methods jointly learn contextualized representations of both language and KG with the MLM objective. In general, learning a model from scratch by this method is excessively time-consuming. For example, CoLAKE training on 8 32G NVIDIA V100 GPUs required 38 hours (Sun et al., 2020). Our method reduced the training time because it requires only fine-tuning. Specifically, the training runtimes of ILKA$_{\text{RoBERTa}}$ with fine-tuning in three epochs on CoLA, STS-B, MRPC, and RTE were approximately 10 mins (see Appendix D for details). Despite its short learning time, our method was more accurate than the existing PLM+KG methods.

	TACRED			OpenEntity		
Model	P	R	F1	P	R	F1
ILKA-head$_{\text{WordNet}}$	73.9	71.9	72.9	75.3	78.1	76.7
ILKA-head$_{\text{Wiki}}$	74.5	69.5	71.9	75.1	77.8	76.4

Table 8: Results of the Knowledge Graph Variants on TACRED and OpenEntity using the proposed ILKA$_{\text{RoBERTa}}$.

Our method needs more training time than pure PLMs because it requires POS tagging first and then entity lookup. In our experimental environment (24GB NVIDIA® TITAN RTX™ GPU), RoBERTa's fine-tuning runtime is 3.9 and 2.8 min for CoLA and STS-B, respectively. In contrast, ILKA-head takes about 11.5 and 7.8 min for CoLA and STS-B, respectively (details in Appendix D), so it takes about three times longer for fine-tuning. For inference, we observed the same tendency. We believe that the increase in learning time does not pose a significant practical problem.

5.3 Discussion

Our proposed method improves the F1 scores for the relation classification task (TACRED) and the entity typing task (OpenEntity), as shown in Tables 3 and 4. However, it is also true that the improvement is insignificant. An analysis of the number of KG entities in a sentence shows that only a few

entities appear in an input sentence. Specifically, the average number of entities in a sentence is 6.5 subwords (5% for a maximum token length of 128), and the rest are empty. This could have hampered learning efficiency.

6 Conclusion

We proposed the KG-attention mechanism and developed the ILKA model to integrate KGs into PLMs in a consistent manner. Unlike the existing PLM+KG integration methods, KG-attention generalizes the embeddings of neighboring entities using the relation embeddings, and selects a proportion of these embeddings through the attention mechanism. Accordingly, it can handle relations between unconnected entities in the KG. In the relation classification and entity typing experiments, ILKA yielded higher Recall and F1 scores than conventional PLM+KG methods. In the GLUE task, factual knowledge produced no negative effect on language comprehension by ILKA. In addition, the higher scores for COPA than for the PLM alone suggest that our method adequately processes complex language comprehension tasks.

Our proposed method has one limitation: when the number of relations is large, the dimension of the attention enlarges accordingly. To avoid this size explosion, we could employ the weighted sums of relation embeddings as keys and queries instead of arranging all relation embeddings in the direction of attention.

Ethical Concerns

This study used the existing datasets WN18 (Bordes et al., 2013), DBpedia50k (Shi and Weninger, 2018), TACRED (Zhang et al., 2017), OpenEntity (Choi et al., 2018), GLUE (Wang et al., 2018), and COPA (Roemmele et al., 2011), which are not expected to present any ethical concerns.

References

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2021. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, page 1247–1250. Association for Computing Machinery.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. 2021. PairRE: Knowledge graph embeddings via paired relation vectors. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4360–4369, Online. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

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*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pride Kavumba, Naoya Inoue, Benjamin Heinzerling, Keshav Singh, Paul Reisert, and Kentaro Inui. 2019. When choosing plausible alternatives, clever hans can be clever. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 33–42, Hong Kong, China. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. 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*, AAAI'15, page 2181–2187. AAAI Press.

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*, abs/1907.11692.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Dat Quoc Nguyen. 2020. A survey of embedding models of entities and relationships for knowledge graph completion. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 1–14, Barcelona, Spain (Online). Association for Computational Linguistics.

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 International Conference on Machine Learning*, ICML'11, page 809–816.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford University.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *AAAI*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. CoLAKE: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3660–3670, Barcelona, Spain (Online). International Committee on 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 *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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 *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2263–2272, Florence, Italy. Association for Computational Linguistics.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19, page 950–958, New York, NY, USA. Association for Computing Machinery.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357, Florence, Italy. Association for Computational Linguistics.

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

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Details of WN18 and DBpedia50k

Table 9 present the scale of WN18 and DBPedia50k and Table 10 presents the types of relations in WN18.

	WN18	DBPedia50k
#(ENTITIES)	40,943	49,900
#(RELATIONS)	18	351
#(Triples in Train)	141,442	32,388
#(Triples in Valid)	5,000	399
#(Triples in Test)	5,000	10,969

Table 9: Scales of WN18 and DBPedia50k.

_hyponym,_hypernym,_part_of,
_derivationally_related_form,
_has_part,_similar_to,_also_see
_member_meronym,_member_holonym,
_member_of_domain_topic,
_synset_domain_topic_of,
_instance_hyponym,_instance_hypernym,
_member_of_domain_region,_verb_group,
_synset_domain_region_of,
_member_of_domain_usage,
_synset_domain_usage_of

Table 10: Types of relations in WN18.

B Training TransE using PyKEEN

PyKEEN is a Python package designed for training and evaluating KG embedding models. It simply describes the training of the knowledge embedding model in the form of a pipeline. Figure 6 shows the code for training the knowledge embedding model. A KG embedding model can be trained by specifying the training conditions in PyKEEN (e.g., the model, dataset name, and number of dimensions).

C GLUE task

The GLUE is the common language comprehension task in NLP. It consists of nine tasks: two single-sentence tasks (SST-2, CoLA), three sentence similarity tasks (MRPC, STS-B, QQP), and four natural language inference (NLI) tasks (MNLI, QNLI, RTE, WNLI). Following Zhang et al. (2019), we evaluated ILKA for eight tasks excluding WNLI. WNLI was excluded from the evaluation data because none of the benchmark methods were evaluated against it. The reasons are described in (Devlin et al., 2019).

Code 1 Training TransE in PyKEEN

```
modeldir = "dirname"
model = pipeline(
        model='TranE',
        dataset='wn18',
        model_kwargs=dict(
                embedding_dim=50,
        ),
        training_kwargs=dict(
                num_epochs=1000,
                batch_size=128,
                checkpoint_name='checkpoint.pt',
                checkpoint_frequency=50,
                checkpoint_directory=dirname,
        ),
)
```

Figure 6: Code for training the knowledge embedding model in PyKEEN. This model uses WN18 with TransE.

D Training Runtime

Figure 7 shows the training runtimes of $\text{ILKA}_{\text{RoBERTa}}$ over three epochs of fine tuning in the CoLA, STS-B, MRPS, and RTE of the GLUE task. Plotted are the averages of five training runs under each condition. The fine-tuning time of $\text{ILKA}_{\text{RoBERTa}}$ on these tasks was approximately 10 minutes.

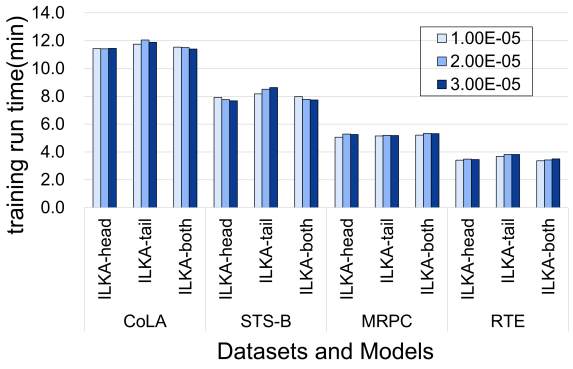

Figure 7: Training runtime for fine-tuning (epochs = 3).