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 BERT\textsc{Large} and “PLM+KG” represents the model using BERT\textsc{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.
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 improvement 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., \((\text{car}, \text{IsA}, \text{vehicle})\)), where \(\mathcal{V}\) is the vocabulary and \(\mathcal{R}\) is the set of relations. \(h\) is a \textit{head entity}, \(r\) is a relation, and \(t\) is a \textit{tail entity}.

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:

\[
\text{v}_{\text{KG}}(t) \approx \phi_h(\text{v}_{\text{KG}}(h), \text{v}_r(t)),
\]

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

<table>
<thead>
<tr>
<th>Approximate formula</th>
<th>Approximate formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE (\text{(Bordes et al., 2013)})</td>
<td>(\text{v}(t) \approx \text{v}(h) + \text{v}(r))</td>
</tr>
<tr>
<td>TransR (\text{(Lin et al., 2015)})</td>
<td>(\text{v}_r(t) \approx \text{v}(h) + \text{v}(r))</td>
</tr>
<tr>
<td>RotatE (\text{(Sun et al., 2019)})</td>
<td>(\text{v}(t) \approx \text{v}(h) \circ \text{v}(r))</td>
</tr>
<tr>
<td>PairRE (\text{(Chao et al., 2021)})</td>
<td>(\text{v}(t) \approx \text{v}(h) \circ \text{v}^H(r) \circ \text{v}^T(r))</td>
</tr>
</tbody>
</table>

Table 1: List of approximate formulas of KG embedding models. \(\text{v}(\cdot)\) is the embedding function specific to the relation \(r\). \(\circ\) is the Hadamard product. \(\oplus\) denotes the element-wise division. \(\text{v}^H(r)\) and \(\text{v}^T(r)\) are embeddings of the relation \(r\) for head and tail entities, respectively.
entities 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-neighboring embeddings as $\phi_i(v_{KG}(w), v_i(r))$, 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 $\phi_i(v_{KG}(w), v_i(\text{PartOf}))$, such as $\phi_i(v_{KG}(\text{car}), v_i(\text{PartOf})) = v_{KG}(\text{car}) - v_i(\text{PartOf})$ in TransE. In the following, $\phi$ refers to either $\phi_i$ or $\phi_t$.

**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; 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., \text{IsA}, \text{HasA}, \text{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 \text{car\_hasA}, \text{car\_hasA}, and \text{car\_AtLocation}. For example, the attention between \text{engine} and \text{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 $\phi$ 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 $w = [w^{(1)}, \ldots, w^{(L)}]$, where $w$ denotes a subword and $L$ denotes the length of the sequence, KG-attention integrates $\phi$ into attention mechanism as follows:

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

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

$Q^{(i)} = v_{KG}(w^{(i)}) \oplus v_{PLM}(w^{(i)})$ \hspace{1cm} (4)
We can simply compute and concatenate the KG between entities even if the relation between the entities is not connected in the KG. Among the compared methods, only ILKA achieves this functionality.

### 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>PLM +KG</th>
<th>Relation between entities connected</th>
<th>Relation between entities not connected</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERMINE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>KnowBERT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CoLAKE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>KEPLER</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ILKA (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2: Comparison of ILKA with existing methods.
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 $v_{KG}(t) \approx v_{KG}(h) + v_r(r)$. Our KG-attention is designed to integrate the head $\rightarrow$ tail relation with the PLM embeddings as follows:

$$K_{head}^{(i,j)} = (v_{KG}(w^{(i)}) + v_r(r^{(j)})) \oplus v_{PLM}(w^{(i)}).$$  

$^1$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
\[
K_{\text{tail}}^{(i,j)} = (v_{\text{KG}}(w^{(i)}) - v_{r^{(i,j)}}) \oplus v_{\text{PLM}}(w^{(i)}). \tag{7}
\]

3) ILKA-both: The two relations above can be combined by concatenating the head and tail vectors from $v_{\text{KG}}(w^{(i)})$:
\[
K_{\text{both}}^{(i,j)} = \begin{cases} 
K_{\text{head}}^{(i,j)} & \text{if } j \in \{1, \ldots, |R|\} \\
K_{\text{tail}}^{(i,j-|R|)} & \text{if } j \in \{|R| + 1, \ldots, 2|R|\}. 
\end{cases} \tag{8}
\]

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

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: BERTBASE for comparison with the BERT-based models ERNIE and KnowBERT and RoBERTaBASE 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. 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). 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
Table 3: Micro precision, recall and F1 scores on TACRED (%). The KnowBERT results shown in Table 3 were reevaluated in Wang et al. (2021).

<table>
<thead>
<tr>
<th>Base-model</th>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BERT</td>
<td>67.2</td>
<td>64.8</td>
<td>66.0</td>
</tr>
<tr>
<td></td>
<td>ERNIE</td>
<td>70.0</td>
<td>66.1</td>
<td>68.0</td>
</tr>
<tr>
<td></td>
<td>KnowBERT</td>
<td>73.5</td>
<td>64.1</td>
<td>68.5</td>
</tr>
<tr>
<td></td>
<td>ILKA-head</td>
<td>72.6</td>
<td>68.9</td>
<td>70.7</td>
</tr>
<tr>
<td></td>
<td>RoBERTa</td>
<td>70.8</td>
<td>69.6</td>
<td>70.2</td>
</tr>
<tr>
<td></td>
<td>KEPLER</td>
<td>71.5</td>
<td>72.5</td>
<td>72.0</td>
</tr>
<tr>
<td></td>
<td>ILKA-head</td>
<td>73.9</td>
<td>71.9</td>
<td>72.9</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Base-model</th>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BERT</td>
<td>76.4</td>
<td>71.0</td>
<td>73.6</td>
</tr>
<tr>
<td></td>
<td>ERNIE</td>
<td>78.4</td>
<td>72.9</td>
<td>75.6</td>
</tr>
<tr>
<td></td>
<td>KnowBERT</td>
<td>77.9</td>
<td>71.2</td>
<td>74.4</td>
</tr>
<tr>
<td></td>
<td>ILKA-head</td>
<td>75.5</td>
<td>74.4</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td>RoBERTa</td>
<td>75.1</td>
<td>73.4</td>
<td>74.3</td>
</tr>
<tr>
<td></td>
<td>KEPLER</td>
<td>77.0</td>
<td>75.7</td>
<td>76.4</td>
</tr>
<tr>
<td></td>
<td>ILKA-head</td>
<td>78.1</td>
<td>76.7</td>
<td></td>
</tr>
</tbody>
</table>

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-headBERT outperformed BERT-original and ERNIE in the following tasks: MNLI-mm, QQP, CoLA, STS-B, MRPC, and RTE. Moreover, ILKA-headRoBERTa outperformed RoBERTa-original, CoLAKE, and KEPLER in the following tasks: MNLI-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$_{RoBERTa}$ on the TACRED and OpenEntity tasks. In the “ILKA$_{w/o-KGneighbors}$” 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$_{w/o-KGneighbors}$”, suggesting that when
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BERT</strong></td>
<td><strong>84.6/83.6</strong></td>
<td>71.2</td>
<td>90.5</td>
<td>93.5</td>
<td>52.1</td>
<td>83.8</td>
<td>88.9</td>
<td>66.4</td>
</tr>
<tr>
<td><strong>ERNIE</strong></td>
<td>84.0/83.2</td>
<td>71.2</td>
<td><strong>91.3</strong></td>
<td>93.5</td>
<td>52.3</td>
<td>83.2</td>
<td>88.2</td>
<td>68.8</td>
</tr>
<tr>
<td><strong>ILKA-head</strong></td>
<td><strong>83.8/84.4</strong></td>
<td><strong>87.7</strong></td>
<td>90.7</td>
<td>92.2</td>
<td><strong>62.6</strong></td>
<td><strong>89.3</strong></td>
<td><strong>91.1</strong></td>
<td><strong>69.0</strong></td>
</tr>
<tr>
<td><strong>RoBERTa</strong></td>
<td><strong>87.5/87.3</strong></td>
<td>91.9</td>
<td><strong>92.8</strong></td>
<td><strong>94.8</strong></td>
<td>63.6</td>
<td><strong>91.2</strong></td>
<td><strong>90.2</strong></td>
<td><strong>78.7</strong></td>
</tr>
</tbody>
</table>

Table 6: Test results on COPA. The results of BERT\_LARGE and RoBERTa\_LARGE were reported by Kavumba et al. (2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>TACRED P</th>
<th>TACRED R</th>
<th>TACRED F1</th>
<th>OpenEntity P</th>
<th>OpenEntity R</th>
<th>OpenEntity F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ILKA-head</strong></td>
<td><strong>73.9</strong></td>
<td><strong>71.9</strong></td>
<td><strong>72.9</strong></td>
<td><strong>75.3</strong></td>
<td><strong>78.1</strong></td>
<td><strong>76.7</strong></td>
</tr>
<tr>
<td><strong>ILKA-head_Wiki</strong></td>
<td><strong>74.5</strong></td>
<td><strong>69.5</strong></td>
<td><strong>74.1</strong></td>
<td><strong>76.4</strong></td>
<td><strong>79.9</strong></td>
<td><strong>78.1</strong></td>
</tr>
<tr>
<td><strong>ILKA-head_both</strong></td>
<td><strong>76.4</strong></td>
<td><strong>70.7</strong></td>
<td><strong>73.5</strong></td>
<td><strong>74.0</strong></td>
<td><strong>73.2</strong></td>
<td><strong>73.6</strong></td>
</tr>
</tbody>
</table>

Table 7: Results of the ablation study on TACRED and OpenEntity using the proposed ILKA-RoBERTa.

Table 8: Results of the Knowledge Graph Variants on TACRED and OpenEntity using the proposed ILKA-RoBERTa.

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-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.

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


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


### A Details of WN18 and DBpedia50k

Table 9 presents the scale of WN18 and DBpedia50k and Table 10 presents the types of relations in WN18.

<table>
<thead>
<tr>
<th></th>
<th>WN18</th>
<th>DBpedia50k</th>
</tr>
</thead>
<tbody>
<tr>
<td>#(ENTITIES)</td>
<td>40,943</td>
<td>49,900</td>
</tr>
<tr>
<td>#(RELATIONS)</td>
<td>18</td>
<td>351</td>
</tr>
<tr>
<td>#(Triples in Train)</td>
<td>141,442</td>
<td>32,388</td>
</tr>
<tr>
<td>#(Triples in Valid)</td>
<td>5,000</td>
<td>399</td>
</tr>
<tr>
<td>#(Triples in Test)</td>
<td>5,000</td>
<td>10,969</td>
</tr>
</tbody>
</table>

Table 9: Scales of WN18 and DBpedia50k.

<table>
<thead>
<tr>
<th>Relation Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hyponym</em></td>
<td>hypernym</td>
</tr>
<tr>
<td><em>hypernym</em></td>
<td>part_of</td>
</tr>
<tr>
<td><em>part_of</em></td>
<td><em>related_form</em></td>
</tr>
<tr>
<td><em>has_part</em></td>
<td>similar_to</td>
</tr>
<tr>
<td><em>also_see</em></td>
<td>member_meronym_</td>
</tr>
<tr>
<td><em>member_holonym</em></td>
<td>member_of_domain_topic_</td>
</tr>
<tr>
<td><em>member_of_domain_region</em></td>
<td><em>verb_group</em></td>
</tr>
<tr>
<td><em>synset_domain_region_of</em></td>
<td><em>member_of_domain_usage</em></td>
</tr>
<tr>
<td><em>member_of_domain_usage_of</em></td>
<td><em>synset_domain_usage_of</em></td>
</tr>
</tbody>
</table>

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).

#### Code 1 Training TransE in PyKEEN

```python
1: modeldir = "dirname"
2: model = pipeline(
3:    model='TranE',
4:    dataset='wn18',
5:    model_kwargs=dict(
6:        embedding_dim=50,
7:    ),
8:    training_kwargs=dict(
9:        num_epochs=1000,
10:        batch_size=128,
11:        checkpoint_name='checkpoint.pt',
12:        checkpoint_frequency=50,
13:        checkpoint_directory=dirname,
14:    ),
15: )
```

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 ILKA RoBERTa over three epochs of fine-tuning in the CoLA, STS-B, MRPC, and RTE of the GLUE task. Plotted are the averages of five training runs under each condition. The fine-tuning time of ILKA RoBERTa on these tasks was approximately 10 minutes.

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

### 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).