Ered: Enhanced Text Representations with Entities and Descriptions

Anonymous ACL submission

Abstract

External knowledge, e.g., entities and entity descriptions, can help humans understand texts. Many works have been explored to include external knowledge in the pre-trained models. These methods, generally, design pre-training tasks and implicitly introduce knowledge by updating model weights, alternatively, use it straightforwardly together with the original text. Though effective, there are some limitations. On the one hand, it is implicit and only model weights are paid attention to, the pre-trained entity embeddings are ignored. On the other hand, entity descriptions may be lengthy, and inputting into the model together with the original text may distract the model's attention. This paper aims to explicitly include both entities and entity descriptions in the fine-tuning stage. First, the pre-trained entity embeddings are fused with the original text representation and updated by the backbone model layer by layer. Second, descriptions are represented by the knowledge module outside the backbone model, and each knowledge layer is selectively connected to one backbone layer for fusing. Third, two knowledge-related auxiliary tasks, i.e., entity/description enhancement and entity enhancement/pollution task, are designed to smooth the semantic gaps among evolved representations. We conducted experiments on four knowledge-oriented tasks and two common tasks, and the results achieved a new state-of-the-art on several datasets. Besides, we conduct an ablation study to show that each module in our method is necessary.

1 Introduction

Pre-trained language models (PLMs), including BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b), have achieved state-of-the-art (SOTA) performances on various natural language processing (NLP) tasks. These PLMs can learn rich linguistic knowledge from unlabeled text (Liu et al., 2019a). However, they capture some kinds of statistical co-occurrence and cannot sufficiently capture fact or commonsense knowledge (Petroni et al., 2019; Liétard et al., 2021). They always have better representation on popular token instead of tail token (Orr et al., 2020a).

Text	The British Information Commissioner 's Office invites Web users to locate its address using Google Maps .
Mention	Information Commissioner 's Office
Span	(12, 46)
Entity	Information Commissioner's Office
Description	British data protection authority

Table 1: An example of a text and its associated entities and descriptions, extracted from Open Entity dataset.

Entities and its associated descriptions in knowledge graphs (KGs), e.g., ConceptNet (Speer et al., 2017), WordNet (Miller, 1995), Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Brümmer et al., 2016), just to name a few, contain extensive information. Table 1 shows an example of a given text and its associated entities and entity descriptions (only one is shown in the table), obviously, the description can help understand. Some works have focused on incorporating entities or entity descriptions into PLMs (Xiong et al., 2019; Peters et al., 2019; Levine et al., 2020; Zhao et al., 2022; Zhang et al., 2019; Yamada et al., 2020; Wang et al., 2021b; Xu et al., 2021b; Wang et al., 2021a; Xu et al., 2021a). Usually, they design knowledge-related pre-training tasks, e.g., entity prediction and entity relation prediction tasks, to continue pre-training the models on a large-scale corpus. External knowledge is therefore implicitly introduced by updating the models' parameters. Alternatively, they directly append the text of entities or descriptions to the original input text, treating entities or descriptions as additional text to enrich the original entry. Although these methods have yielded promising results, we argue that they have the following shortcomings. Firstly, when entities and de-

scriptions are involved in continuing pre-training, the knowledge is only implicitly injected by updating the model parameters. Moreover, during this process, the entity embeddings pre-trained by these pre-training tasks, which cost many computation resources and are of great value, are wasted. Secondly, when the entities and descriptions are appended directly to the original text, it will leads to huge costs of computing resources and a diversion of the model's attention, as the descriptions are always long texts.

To alleviate the above issues, we propose Ered, where both entities and entity descriptions are explicitly included to enhance the representation of the original text. Firstly, the pre-trained entity embeddings are explicitly fused with the original input representations, and then updated during the training, that is, the output of the current layer is fed to the next layer. Secondly, description texts are represented by the knowledge module, which is a light model outside the backbone model, and aims to represent the long description text separately. Moreover, each knowledge layer is selectively connected to one backbone layer, to enhance corresponding text representation. Note that, the description representations are updated by the knowledge module layer by layer, but kept fixed when feed to the backbone layer. Finally, two entity/description-related auxiliary tasks, namely entity/description enhancement and entity enhancement/pollution task, are designed to narrow the semantic gaps among the representations of texts, entities and descriptions. We conduct experiments on two entity-related tasks, i.e., entity typing and relation classification, and two common NLP tasks, i.e., sentiment analysis and extended exact match. The experimental results show that Ered significantly outperforms the baseline models and gets SOTA on several datasets.

2 Related Work

Some works have explored injecting entity or entity description into the pre-trained language models. Some of them include knowledge in the pre-training stage by designing pre-training tasks, while others introduce knowledge directly in the fine-tuning stage.

In the pre-training stage. ERNIE-THU (Zhang et al., 2019) uses static entity embeddings separately learned from KGs. It first obtains all entity embeddings by TransE (Bordes et al., 2013), links the named entity mentions in the text to entities in KGs, and adds the linked entity embeddings to the corresponding mention positions. Besides, it designs pre-training objectives by randomly masking some of the named entity alignments and asking the model to select appropriate entities from KGs to complete the alignments. Same to ERNIE-THU, KnowBert (Peters et al., 2019) incorporates an integrated entity linker in their model and adopts end-to-end training. KEPLER (Wang et al., 2021b) encodes entity descriptions by PLMs as the representations of entities and trains these entity representations by conventional knowledge embedding methods. The model is pre-trained by MLM and this KE objective. In addition to the masked language model (MLM) (Devlin et al., 2019), LUKE (Yamada et al., 2020) randomly masks tokens and entities and then recover them by training the RoBERTa to predict the tokens and the original form of the masked entities in KGs. It provides entity identifier "[MASK]" as additional input, and designs entity-aware self-attention to better use the entity identifier embedding. WKLM (Xiong et al., 2019) designs a pre-training task, which randomly replaces some of the entity names in the input text and asks the model to predict whether an entity name is replaced. K-Adapter (Wang et al., 2021a) designs two adapters as a plug-in, which is pre-trained by relation classification and dependency relation prediction task.

In the fine-tuning stage KT-attn (Xu et al., 2021a) appends entities and entity descriptions directly to the original input text in the fine-tuning stage and designs an attention matrix to avoid computation resource costs induced by descriptions. It also compares with knowledge as text and knowledge as embedding methods.

Our work is different from the works mentioned above. Firstly, both entities and entity descriptions are explicitly introduced to the fine-tuning stage. Secondly, descriptions are processed by the knowledge module, a lighter model, to avoid the impacts induced by these long texts. Besides, the backbone and knowledge module is connected layer-to-layer. Although it appears similar but is different from (Wang et al., 2021a), where hidden states flow from the backbone to the pre-trained adapters. It is naturally a method of knowledge introduction by updating the weights of the models, whereas, the hidden states of Ered flow from the knowledge module to the backbone model for enhancement.

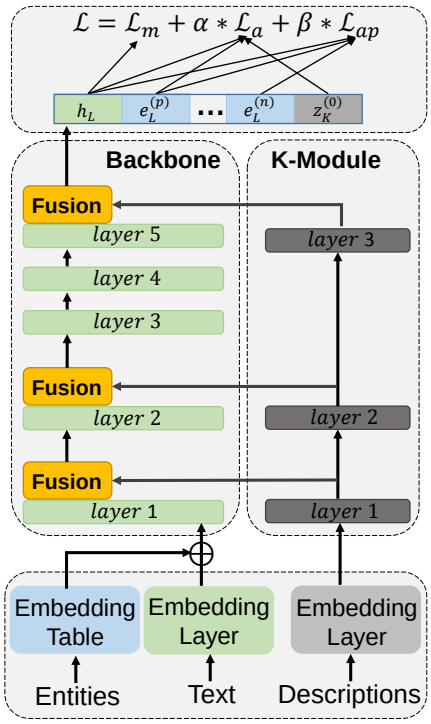


Figure 1: Overview of Ered ($L = 5, K = 3$, the connected layer is $1, 2, 5$).

3 Our Method

In this section, we present the overall framework of Ered, as shown in Figure 1. It is composed of an input layer converting input items to vectors (Section 3.1), a backbone model processing text (Section 3.2), a knowledge module processing descriptions (Section 3.3), a fusion module builds layer-wise connections between the layers of the backbone and knowledge module (Section 3.4), and a prediction layer computing the probability distribution of target classes (Section 3.5).

3.1 Input Layer

The input of Ered includes the original text, entities, and descriptions. The text is fed into the embedding layer of the backbone model, where token embedding, position embedding, and segment embedding are added together. The embeddings of entities are lookup from the entity embeddings table, which is pre-trained by entity-related pre-training tasks. The description is tokenized and then fed into the embedding layer of the knowledge module (K-module). To be specific, given the input sentence S, we recognize all the entities by entity linker, it will output the mention span in the input and the entities in the Wikidata, and then we associate each entity with its description. After that, we tokenize S into subword sequence $X = \{x_1, \ldots, x_m\}$, where m is the maximum sequence length of the text[1]. Then, we get its embeddings $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ by the backbone embedding layer. For each description D, we tokenize it into subword sequence $U = \{u_1, \ldots u_n\}$, where n is the maximum sequence length of the descriptions. Then, we get its embedding $\mathbf{U} = \{\mathbf{u}_1, \ldots \mathbf{u}_n\}$ by the knowledge module embedding layer. Besides, entity embeddings $\mathbf{e}$ are obtained from the entity embedding table $\mathbf{v} \in \mathbb{R}^{|V| \times d_1}$, where $|V|$ is the entity vocabulary size, d_1 is the dimension size of entity embeddings.

3.2 Backbone Model

The backbone model is responsible for capturing semantic representation from the original input tokens. It is a prevalent PLMs, e.g., BERT and RoBERTa, stacking L backbone layers, and we exclude a comprehensive description of this module and refer readers to (Devlin et al., 2019) and (Liu et al., 2019b) for details. In our setting, Transformer (Vaswani et al., 2017) encoder is used, it takes the embeddings $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ as input and computes layer-wise representation. The output of current layer is fed into the next layer,

$$\mathbf{h}_i = \text{Transformer}_i(\mathbf{h}_{i-1}), \quad (1)$$

where $\mathbf{h}_i \in \mathbb{R}^{m \times d_2}, i \in \mathbb{N}^+, i \in [1, L]$ is the representation of the text in the i-th backbone layer, and $\mathbf{h}_0 = \mathbf{X}$. Transformer_i refers to the i-th layer, d_2 is the dimension size of the backbone model.

3.3 Knowledge Module

The knowledge module is responsible for capturing the knowledge-related representations of entity descriptions. It is a light PLMs, that stacks K knowledge layers, outside the backbone model as an external plugin to process the long text. In our setting, the pre-trained DistilBERT (Sanh et al., 2019) is used, and its parameters are frozen. The knowledge module takes the embeddings of entity descriptions as input, and it updates the hidden states of the descriptions layer by layer,

$$\mathbf{z}_k = \text{Knowledge}_k(\mathbf{z}_{k-1}), \quad (2)$$

where $\mathbf{z}_k \in \mathbb{R}^{n \times d_3}, k \in \mathbb{N}^+, k \in [1, K]$ is the representation of the description text in the k-th knowledge layer, and $\mathbf{z}_0 = \mathbf{U}$. Knowledge_k refers to the k-th layer of the knowledge module, d_3 is the dimension size of the knowledge module.

[1]We pad zeros to keep the dimension.

3.4 Fusion Module

Since different models produce the text, entities and descriptions embeddings with different semantic spaces. The fusion module is responsible for narrowing the semantic gaps, fusing the knowledge-related information into the input representation, and outputting an entity/description enhanced text representation. Motivated by (Yang et al., 2021), instead of fusing the final hidden state, we build a layer-wise connection between the backbone and knowledge layers, to achieve deeper integration.

It takes the text representation $\mathbf{h}$, entity embedding $\mathbf{e}$ and description representation $\mathbf{z}$ as input. Since the number layer K of the knowledge module is always less than the number layer L of the backbone layer, some of the backbone layers are connected while others are not. Therefore, an alignment is needed to determine which backbone layer is connected. For connected layers, both entity embedding and corresponding layer-wise representation of descriptions are concatenated to the text representation, and then fed to the next backbone layer for enhanced text representation. Note that, $\mathbf{z}_k$ is only used to enhance $\mathbf{h}$. Formally,

$$\begin{aligned} \mathbf{h}'_{i-1} &= \mathbf{h}_{i-1} \,||\, \mathbf{e}_{i-1} \,||\, f(\mathbf{z}^{(0)}_{k-1}), \\ \mathbf{h}_i, \mathbf{e}_i, \_ &= \text{Transformer}_i(\mathbf{h}'_{i-1}), \\ \mathbf{z}_k &= \text{Knowledge}_k(\mathbf{z}_{k-1}), \end{aligned} \tag{3}$$

where f is a linear function to align dimension, $\mathbf{z}^{(0)}_k$ is the vector in the first position of description, i.e., "[CLS]", output by the k-th knowledge layer. For layers without connection, the entity embedding is concatenated to the text representation, and then fed to the next backbone layer for enhancement,

$$\begin{aligned} \mathbf{h}'_{i-1} &= \mathbf{h}_{i-1} \,||\, \mathbf{e}_{i-1}, \\ \mathbf{h}_i, \mathbf{e}_i &= \text{Transformer}_i(\mathbf{h}'_{i-1}). \end{aligned} \tag{4}$$

For example, as depicted in Figure 1, the backbone model has five layers and the knowledge module has three layers. The shown alignment is that the knowledge layer is connected to the first, second, and fifth backbone layer.

3.5 Prediction Layer

The prediction layer comprises linear layers to map the representation over probability distributions.

Main task. The vector of entity identifier $\mathbf{h}^{(I)}_L$ (detailed in Section 4) is used as the final representation to compute the probability distribution, $\hat{p} = W_1 \cdot \mathbf{h}^{(I)}_L + b_1$. With the given probabilities, cross-entropy loss function is adopted to compute the loss of the main task,

$$\mathcal{L}_m = -\frac{1}{Y} \sum_{y \in Y} y \cdot \log(\hat{p}). \tag{5}$$

Auxiliary tasks. Since the vectors of entities, texts and descriptions are obtained from different models, there have different semantic gaps. To shrink these gaps, motivated by (Zhao et al., 2022), where sentiment words are used to construct enhanced and polluted sentence representation, we design two auxiliary tasks. The first auxiliary task is **entity/description enhancement task**, which is pretty similar to the main task, except that the vector $\mathbf{h}^{(E)}_L$ of target entity or sentence is enhanced with the knowledge representations, $\mathbf{h}_{(a)} = \mathbf{h}^{(E)}_L + \mathbf{e}^{(p)}_L + \mathbf{z}^{(0)}_K$. Then, is is used as the final representation to compute the probability distribution over the target classes, $\hat{p} = W_2 \cdot h_{(a)} + b_2$. Therefore, the loss of the first auxiliary task is,

$$\mathcal{L}_a = -\frac{1}{Y} \sum_{y \in Y} y \cdot \log(\hat{p}). \tag{6}$$

The second auxiliary task is **entity enhancement/pollution task**, where the text representation is enhanced by the representation of its associated entity, i.e., $\mathbf{g}_{(a)} = \mathbf{e}^{(p)}_L + \mathbf{h}^{(E)}_L$ or polluted by randomly sampled ones, i.e., $\mathbf{g}_{(p)} = \mathbf{e}^{(n)}_L + \mathbf{h}^{(E)}_L$, and the model is asked to distinguish them $\hat{c} = W_3 \cdot (\mathbf{g}_{(a)} \,||\, \mathbf{g}_{(p)}) + b_3$. Therefore, the loss of the second auxiliary task is,

$$\mathcal{L}_{ap} = -\frac{1}{C} \sum_{c \in C} c \cdot \log(\hat{c}), \tag{7}$$

where $||$ refers to concatenation operation. Y is the label set of the main task, and C is the label set indicating the position index of the positive entity. $W_1, b_1, W_2, b_2, W_3, b_3$ are model parameters, $\mathbf{e}^{(p)}_L, \mathbf{e}^{(n)}_L$ refer to the representation of the positive and negative entities, respectively. $\mathbf{z}^{(0)}_K$ is the vector in the first position of the last knowledge layer. $\mathbf{h}^{(I)}_L$ and $\mathbf{h}^{(E)}_L$ is the vector in the position $(I), (E)$ of the last backbone layer, and $(I), (E)$ index the position of the entity identifier and entity special token, respectively. It will be detailed in Section 4.

The total loss is a weighted sum of the above three losses, $\mathcal{L} = \mathcal{L}_m + \alpha * \mathcal{L}_a + \beta * \mathcal{L}_{ap}$, where $\alpha > 0$ and $\beta > 0$ are loss coefficients.

4 Experiments

This section presents the implementation details and the results of several NLP tasks. The statistics of these datasets are shown in Table 2. We use LUKE (Yamada et al., 2020) as the backbone model and DistilBERT as the knowledge module. LUKE is based on the large version of RoBERTa ($L = 24, d_2 = 1024$) and DistilBERT is a distilled BERT with $K = 6, d_1 = 768$. We extract descriptions from Wikidata[2], and entity embedding table from the pre-trained LUKE checkpoints[3], which contains 500,000 entities. Positive entities are recognized by the entity linker, while negative entities are randomly sampled from entity vocabulary. Both baselines and our method share the same training parameter for fairness. Note that, we run the experiments several times and report the average results except for FIGER dataset. Please refer to the Appendix A.1 for more implementation details. The source code is available at XXX (we will release all the code when the paper is accepted).

Dataset	Train	Dev	Test	#Types
Open Entity	1,998	1,998	1,998	9
FIGER	2,000,000	10,000	563	113
FewRel	8,000	16,000	16,000	80
TACRED	68,124	22631	15,509	42
SST	67,349	872	–	2
EEM	405,482	101,370	–	2

Table 2: The statistics of Open Entity, FIGER, FewRel, TACRED, SST and EEM datasets.

4.1 Knowledge-orientated Tasks

We first conduct experiments on knowledge-oriented tasks, i.e., entity typing and relation classification. Baselines are described in section 2.

4.1.1 Entity Typing

Entity typing is the task of predicting the types of an entity given its sentence context. Here we use Open Entity (Choi et al., 2018) and FIGER (Ling et al., 2015) datasets, following the split setting as (Zhang et al., 2019; Wang et al., 2021a). To fine-tune our models for entity typing, following the setting of (Yamada et al., 2020), we modify the input token sequence by adding the special token "[ENTITY]" before and after a certain entity, and providing entity identifier "[MASK]" along with the input. The representation of entity identifier "[MASK]" is adopted to perform classification, and the first "[ENTITY]" special token representation is used as text representation. It is treated as a multiple labels classification problem, and binary cross-entropy loss is used to optimize the model. Following the same evaluation criteria used in the previous works, for Open Entity, we evaluate the models using micro precision, recall and F1, and adopt the micro F1 score as the final metric. For FIGER, we adopt accuracy, macro F1, and micro F1 scores for evaluation.

[2] https://www.wikidata.org/w/api.php

[3] https://huggingface.co/studio-ousia/luke-large

Model	Prec.	Rec.	Mi-F1
$\text{BERT}_{\text{base}}$	76.4	71.0	73.6
ERNIE-THU (Zhang et al., 2019)	78.4	72.9	75.6
KnowBERT (Peters et al., 2019)	78.6	73.7	76.1
$\text{RoBERTa}_{\text{large}}$	77.6	75.0	76.2
K-Adapter (Wang et al., 2021a)	79.0	76.3	77.6
LUKE (Yamada et al., 2020)	79.9	76.6	78.2
$\text{RoBERTa}^{*}_{\text{large}}$	78.3	74.4	76.3
K-Adapter*	78.0	76.3	77.0
LUKE*	80.8	74.7	77.6
LUKE+Adapter	78.3	76.1	77.4
Ered	80.3	75.9	**78.1**

Table 3: Results of entity typing on the Open Entity dataset. * refers to reproduced results.

Model	Acc	Ma-F1	Mi-F1
$\text{BERT}_{\text{base}}$	52.0	75.2	71.6
ERNIE-THU (Zhang et al., 2019)	57.2	75.6	73.4
WKLM (Xiong et al., 2019)	60.2	82.0	77.00
$\text{RoBERTa}_{\text{large}}$	56.3	82.4	77.8
K-Adapter (Wang et al., 2021a)	61.8	84.9	80.5
$\text{RoBERTa}^{*}_{\text{large}}$	54.9	81.6	77.1
LUKE*	57.4	**82.1**	78.1
Ered	**60.6**	77.7	**78.8**

Table 4: Results of entity typing on the FIGER dataset (maximum sequence length is reduced from 256 to 128).

Results The results on Open Entity and FIGER dataset are presented in Table 3 and 4, respectively. We can see that Ered outperforms the previous SOTA by 0.5 F1 points on Open Entity. On FIGER dataset, it outperforms the reproduced RoBERTa and LUKE by 1.7 and 0.7 micor F1 points, respectively. Besides, to demonstrate the effectiveness of our proposed model, we also reimplement LUKE+Adapter, where the two adapters pre-trained by K-Adapter (Wang et al., 2021a) are transferred to the LUKE model. We find that, with the plugin of the two adapters, there are no expected gains, but drops of 0.2 F1 points on the Open Entity dataset. We attribute these results to the semantic spaces of the two, namely LUKE and

Model	Prec.	Rec.	Mi-F1
$\text{BERT}_{\text{base}}$	85.1	85.1	84.9
ERNIE-THU (Zhang et al., 2019)	88.5	88.4	88.3
$\text{RoBERTa}^{*}_{\text{large}}$	88.8	88.8	88.8
LUKE*	89.4	89.4	89.4
Ered	90.3	90.3	**90.3**

Table 5: Results of entity typing on the FewRel dataset.

Model	Prec.	Rec.	Mi-F1
$\text{BERT}_{\text{base}}$	67.2	64.8	66.0
ERNIE-THU (Zhang et al., 2019)	70.0	66.1	67.97
KnowBERT (Peters et al., 2019)	71.6	71.4	71.5
$\text{RoBERTa}_{\text{base}}$	70.4	71.1	70.7
KEPLER-Wiki (Wang et al., 2021b)	71.5	72.5	72.0
$\text{RoBERTa}_{\text{large}}$	70.2	72.4	71.3
K-Adapter (Wang et al., 2021a)	70.1	74.0	72.0
LUKE (Yamada et al., 2020)	70.4	75.1	72.7
LUKE*	71.2	72.2	71.7
Ered	71.3	73.7	**72.5**

Table 6: Results of relation classification on TACRED.

adapters, being different. We think it is necessary to narrow this semantic gap, and the ablation study in Section 4.3 confirms our view.

4.1.2 Relation Classification

Relation classification is the task of determining the relation between the given head and tail entities in a sentence. Here we use TACRED (Zhang et al., 2017) and FewRel (Han et al., 2018) datasets, following the split setting as (Zhang et al., 2019; Wang et al., 2021a). Following (Yamada et al., 2020), we modify the input token sequence by adding the special token "[HEAD]" before and after the first entity, adding "[TAIL]" before and after the second entity, and adding two entity identifiers "[MASK]" as additional input. The representations of entity identifiers are concatenated to perform relation classification, and the token representations of the first special token "[HEAD]" and "[TAIL]" are concatenated to represent the original text. We evaluate the models using micro precision, recall and F1, and adopt micro F1 score as the final metric to represent the model performance as in previous works.

Results The results on FewRel and TACRED are shown in Table 5 and 6, respectively. Notably, the gap between the original reported results in LUKE and the reproduced results may probably be because of different maximum sequence lengths, i.e., from 512 to 256, open-source library, i.e., from AllenNLP to HuggingFace, and reports, i.e., from the best to average results. Compared with the previous best-published models, Ered achieves an improvement of 0.9 and 0.8 F1 points, respectively, demonstrating the usefulness of the representations of entities and entity descriptions and the effectiveness of our designed framework.

Model	ACC
$\text{BERT}_{\text{base}}$	93.00
ERNIE-THU (Zhang et al., 2019)	93.50
$\text{KT-attn}^{*}_{\text{bert-base}}$	93.33
$\text{RoBERTa}_{\text{base}}$	94.72
KEPLER-Wiki (Wang et al., 2021b)	94.50
$\text{KT-attn}_{\text{roberta-base}}$ (Xu et al., 2021a)	94.84
$\text{KT-attn}^{*}_{\text{roberta-base}}$	94.72
$\text{RoBERTa}_{\text{large}}$	96.22
$\text{RoBERTa}^{*}_{\text{large}}$	96.10
$\text{KT-attn}_{\text{roberta-large}}$ (Xu et al., 2021a)	96.44
$\text{KT-attn}^{*}_{\text{roberta-large}}$	96.44
Ered	**96.90**

Table 7: Results of sentiment analysis on the SST dataset. * refers to reproduced results.

Model	ROC AUC	PR AUC
$\text{BERT}^{*}_{\text{base}}$	85.60	90.64
$\text{KT-attn}^{*}_{\text{bert-base}}$	86.27	91.02
$\text{RoBERTa}^{*}_{\text{base}}$	86.08	90.94
$\text{KT-attn}^{*}_{\text{roberta-base}}$	86.87	91.38
$\text{RoBERTa}^{*}_{\text{large}}$	87.36	91.82
$\text{KT-attn}^{*}_{\text{roberta-large}}$	**88.29**	92.46
Ered	87.90	92.27

Table 8: Results on the EEM dataset.

4.2 Common Tasks

(Zhang et al., 2019; Wang et al., 2021b; Xu et al., 2021a) show that common tasks may not require external knowledge, which may harm the language model's representation to some extent. To test Ered, we conduct experiments on two common tasks, including sentence-level sentiment analysis and extended exact match tasks.

Sentence-level sentiment analysis aims to predict the sentiment polarity of the given sentence. We use SST dataset, obtained from General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which collects several popular NLP tasks (Rajpurkar et al., 2016; Socher et al., 2013). The entity identifier "[MASK]" is inserted and its representation is used for prediction. While, the vector in the first position of the last backbone layer is adopted to compute the enhanced or polluted text representation, we evaluate it by accuracy (ACC).

Dataset	Open Entity			FIGER			FewRel			TACRED			SST	EEM	
Model	Prec.	Rec.	Mi-F1	Acc	Ma-F1	Mi-F1	Prec.	Rec.	Mi-F1	Prec.	Rec.	Mi-F1	ACC	ROC AUC	PR AUC
Baseline	80.8	74.7	77.6	57.4	**82.1**	78.1	89.4	89.4	89.4	71.2	72.2	71.7	96.4	**88.3**	92.5
Ered	80.3	75.9	**78.0**	**60.6**	77.7	**78.8**	90.3	90.3	**90.3**	71.3	73.7	**72.5**	**96.9**	87.9	92.3
w/o a	80.4	74.6	77.4	59.9	81.1	78.4	90.2	90.2	90.2	70.8	72.5	71.7	96.3	87.7	92.1
w/o b	80.1	75.3	77.6	-	-	-	89.9	89.9	89.9	72.3	72.3	72.1	96.1	87.7	92.1
w/o a, b	79.9	75.3	77.5	-	-	-	89.9	89.9	89.9	72.7	71.1	71.8	96.4	87.8	92.2

Table 9: Ablation results of each auxiliary task.

Extended exact match is a kind of matching mode of search advertising in the search advertisements scene, which requires the user's search term, i.e., query, must exactly match the bid term, i.e., keyword. Therefore, the task is to determine whether the given query and keyword are exactly matched or not. It is a binary classification problem. A private EEM dataset from the Bing ads group is used, where an entity identifier is provided as an extra input and the vector in the first position is used to perform all classifications. We evaluate it by ROC AUC and PR AUC.

Results Table 7 shows the results on SST. We can see that Ered outperforms all the baselines, and increases the accuracy of RoBERTa$_{\text{large}}$ and KT-attn by 0.8 and 0.45 accuracy points, respectively. Table 8 shows the results on EEM, it shows that Ered outperforms the RoBERTa$_{\text{large}}$ about 0.6 ROC AUC points and is comparable to KT-attn. We claim that common tasks are not knowledge-intensive, but the reasonable use of knowledge can promote the representation of the language model, just as our human beings do.

4.3 Ablation Study

In this subsection, we analyze the impacts of external knowledge and auxiliary tasks, where **w/o a** refers to fine-tuning Ered without entity/description enhancement task, **w/o b** refers to removing entity enhancement/pollution task, **w/o a, b** refers to no auxiliary is adopted except entity and description representation. As shown in Table 9, **w/o a** is better than **w/o b** is some cases but worser in other cases. Ered is better than **w/o**, demonstrating the necessity of the auxiliary tasks and two auxiliary tasks can mutually enhance each other. Moreover, when no auxiliary task is adopted, the ablation models suffer significant drops, about 0.3 to 0.8 points, demonstrating that the straightforward introduction of external representation may not be helpful or even harm the performance. In summary, according to the results, we claim that when external representation is introduced, which may have a different semantic space from the backbone, auxiliary tasks that aim to **narrow semantic gap** are necessary. These results also explain that combining pre-trained adapters from K-Adapter with LUKE does not boost the performance.

5 Analysis

5.1 Effects of Layer Alignment

As described in subsection 3.4, each knowledge layer is selectively connected with one backbone layer. Therefore, in this section, we analyze the impacts of different layer alignment, the results are shown in Table 10. For the test four datasets, the differences between the best and worst results are 0.7, 0.1, 0.6, and 0.5, respectively, indicating that different layer alignment has significant impacts. "last" can achieve the top results on Open Entity, FewRel and TACREDdataset, but obtain the worst results on SST, demonstrating different layer fusion impacts different datasets. In most cases, "first & last" can get a relatively solid result.

Layers	Open Entity	FewRel	TACRED	SST	avg.
last	78.1	90.4	72.6	95.6	84.18
first	77.6	90.3	72.5	96.1	84.13
middle	77.5	90.3	72.0	96.1	83.98
first & last	78.1	90.3	72.4	96.1	**84.23**
uniform	78.2	90.4	72.0	96.0	84.15

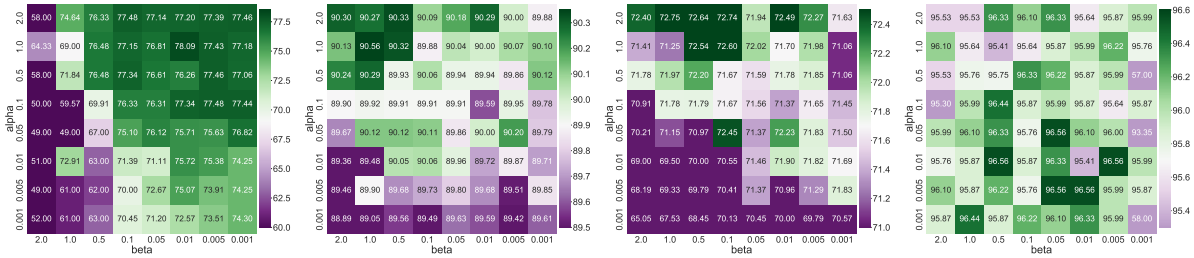
Table 10: Results under different layer alignments. "last", "first", "middle" refers to the last/first/middle K backbone layers are connected, and "first & last" refers to the first and last $K/2$ backbone layers are connected. "uniform" means that the backbone layers are connected to knowledge layer in a uniform interval. Underline indicates top ranked results.

5.2 Effects of Loss Coefficients

In Section 3.5, we use α and β coefficients to weight the two auxiliary task losses and then add it with the main loss. As reported in previous works (Zhao et al., 2022; Chuang et al., 2022), the auxiliary loss should have smaller weights to avoid domain the model's attention. Therefore, in this section, we search α and β from {2.0, 1.0, 0.5,

α	Open Entity	FewRel	TACRED	SST	β	Open Entity	FewRel	TACRED	SST
2.0	76.8	89.7	71.7	96.0	2.0	68.0	90.0	70.8	95.7
1.0	77.5	89.8	71.9	95.6	1.0	68.0	90.1	71.0	95.5
0.5	77.5	89.6	72.0	96.1	0.5	68.0	90.4	72.0	95.9
0.1	77.5	89.7	71.7	95.9	0.1	77.1	90.2	71.6	95.9
0.05	76.5	89.9	72.0	96.2	0.05	77.4	90.0	72.5	95.8
0.01	75.3	89.6	71.1	96.1	0.01	78.4	90.0	72.3	96.0
0.005	73.6	89.7	71.5	96.3	0.005	77.7	89.7	71.8	95.5
0.001	72.2	89.5	70.1	95.9	0.001	77.2	89.9	72.2	96.4

Table 11: Results under different values of α and β. Underline indicates top ranked results.



(a) Open Entity (b) FewRel (c) TACRED (d) SST

Figure 2: Impacts of loss coefficients.

0.1, 0.05, 0.01, 0.005, 0.001} to demonstrate this parameter's impacts, the results are shown in Table 11. We can find that when we solely adopt the first task, i.e., the entity/description enhancement task, 1.0 to 0.05 is better for the three knowledge-related datasets, and 0.05 to 0.005 is better for SST. When we solely adopt the second task, namely the entity enhancement/pollution task, 0.01 to 0.005 is a better choice. In summary, a relatively larger α and smaller β are recommended in most cases. Figer 2 shows the mutual impacts of the two auxiliary tasks. It shows that top results concentrate on the top of Figer 2(a), 2(b) and 2(c), whereas on the contrary for SST. For FewRel and TACRED, the results in the top left corner are better, whereas for Open Entity, that in the top right corner are better.

5.3 Limitations

As show in Eq. 3.5, Ered takes a multi-task loss, these losses introduce parameters, i.e., W_2, b_2, W_3, b_3, to linearly transform the representation to the distribution of target classes. Besides, the dimension size and the semantic space of the backbone and knowledge model are different, and the map from the former to the latter introduces parameters in each fusion module, i.e., $W_{(k)} \in \mathbb{R}^{d_3 \times d_2}, 1 \leq k \in \mathbb{N}^+ \leq K$. Moreover, though the parameters of the knowledge module are frozen, it induces computations and time costs in the inference phase, and this problem can be solved by pre-computation (Borgeaud et al., 2021). Specifically, we pre-compute each knowledge layer representation of all entity descriptions and cache the pre-computed representations for later use.

6 Conclusion

This paper presents a novel architecture Ered for enhancing text representation with entities and entity descriptions. Long description text is represented separately by a lighter knowledge module and then injected to the backbone for knowledge enhancement. On top of the architecture, two entity/description-related auxiliary tasks are introduced to narrow the semantic gap between involved different representations. Empirical results on knowledge-related and common tasks show the effectiveness of Ered compared to current state-of-the-art knowledge enhanced methods. We also conduct extensive ablation studies to demonstrate the impacts of each design choice in Ered. One limitation of our work is that the knowledge module costs computation resources and increases inference time, and it can be easily solved by pre-computation, we leave this for future work. We believe that Ered can provide the NLP community with a new way to utilize knowledge for natural language and thus produce better representations.

7 REFERENCES

References

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

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*.

Martin Brümmer, Milan Dojchinovski, and Sebastian Hellmann. 2016. Dbpedia abstracts: A large-scale, open, multilingual nlp training corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3339–3343.

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.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628.

Yuqing Gao, Jisheng Liang, Benjamin Han, Mohamed Yakout, and Ahmed Mohamed. 2018. Building a large-scale, accurate and fresh knowledge graph. *KDD-2018, Tutorial*, 39:1939–1374.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. Sensebert: Driving some sense into bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667.

Bastien Liétard, Mostafa Abdou, and Anders Søgaard. 2021. Do language models know the way to rome? In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 510–517.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *North American Chapter of the Association for Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2018. Mixed precision training. In *International Conference on Learning Representations*.

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

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Ré. 2020a. Bootleg: Chasing the tail with self-supervised named entity disambiguation. In *Conference on Innovative Data Systems Research*.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2020b. Bootleg: chasing the tail with self-supervised named entity disambiguation. *arXiv preprint arXiv:2010.10363*.

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.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *empirical methods in natural language processing*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *empirical methods in natural language processing*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Johannes M van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P de Vries. 2020. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2197–2200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

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.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418.

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. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Ruochen Xu, Yuwei Fang, Chenguang Zhu, and Michael Zeng. 2021a. Does knowledge help general nlu? an empirical study. *arXiv preprint arXiv:2109.00563*.

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2021b. Fusing context into knowledge graph for commonsense question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1201–1207.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34.

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.

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.

Qinghua Zhao, Shuai Ma, and Shuo Ren. 2022. Kesa: A knowledge enhanced approach for sentiment analysis. *arXiv preprint arXiv:2202.12093*.

A Appendix

A.1 Implementation Details

For most parameters, we adopt the value recommended in LUKE, and the parameters we use are

Name	Open Entity	FIGER	FewRel	TACRED	SST	EEM
Batch size	4	2048	32	32	128	128
Maximum text length	256	128	256	256	128	32
Maximum description length	64	64	64	64	64	32
Learning rate	1e-5	2e-5	1e-5	1e-5	{1e-5, 2e-5, 3e-5, 5e-5}	
Epoch	3	2	10	5	3	3
Evaluation steps	per epoch	50	per epoch	500	500	500
Warmup ratio	0.06	0.06	0.1	0.1	0.1	0.1
Number of entities	4	2	4	4	4	2
Number of descriptions	1	1	1	1	1	1
α	1.0	1.0	1.0	1.0	1.0	1.0
β	0.01	0.01	1.0	0.1	0.1	1.0
Alignment	the top six backbone layers are connected to knowledge layers					
Recognized entities	ERNIE-THU				TAGME	Satori
Times of experiments	20	1	20	5	4	4
Reported results	average	-	average	average	best	best

Table 12: Hyper-parameters and other details of our experiments. "average" and "best" refer to that the averaged/best results are reported.

listed in Table 12. We optimized the model by AdamW (Loshchilov and Hutter, 2018), and a linear learning rate decay is adopted. Besides, mixed precision (Micikevicius et al., 2018) is adopted to accelerate computation. The number of associated entities and descriptions is searched from {1, 2, 4, 6, 8}, and the α and β are searched from {1.0, 0.1, 0.01}. In our experiments, four entities, one description and $\alpha = 1.0$ are used as default. And the knowledge layer is aligned to the last K backbone layers. Since entities are used, we need entity linker (Wu et al., 2020; Orr et al., 2020b; van Hulst et al., 2020; Ferragina and Scaiella, 2010) to recognize the entities included in the text. In our experiment, we adopt the linked datasets provided by (Zhang et al., 2019), and for SST, TAGME (Ferragina and Scaiella, 2010) is used to perform entity linking. For EEM, entity and entity description are given, which is extracted from Microsoft knowledge graph Satori (Gao et al., 2018). Positive entities are recognized by the entity linker, while negative entities are randomly sampled from entity vocabulary. When no entity is included in one sentence, an entity identifier "[MASK]" is used as a positive entity. For EEM and FIGER dataset, considering its large training samples, we run it just for one time. Note that, considering the large-scale training set of FIGER dataset, to accelerate the training process, we reduce the maximum sequence length from 256 to 128. Besides, with 2 million training samples and only 500 test samples, it is easy to overfit, and we used the parameters recommended in (Zhang et al., 2019; Wang et al., 2021a) and did not do a grid-search. Specifically, the batch size per GPU is 64, the gradient accumulation step is set to 8, four NVIDIA V100 of 32G are used, and then it takes about two hours per epoch and the best results are always obtained in step 300, the learning rate is set to 2e-5, the warmup step is set to 6%, $\alpha = 1.0, \beta = 0.01$, the number of entities and descriptions are set to 2 and 1, respectively. We run training on FIGER for two epochs and evaluate it every 50 steps.