ERNIE: Enhanced Language Representation with Informative Entities

Zhengyan Zhang[1,2,3*], **Xu Han**[1,2,3*], **Zhiyuan Liu**[1,2,3†], **Xin Jiang**[4], **Maosong Sun**[1,2,3], **Qun Liu**[4]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
[4]Huawei Noahs Ark Lab, Huawei Technologies

Abstract

Neural language representation models such as BERT pre-trained on large-scale corpora can well capture rich semantic patterns from plain text, and be fine-tuned to consistently improve the performance of various NLP tasks. However, the existing pre-trained language models rarely consider incorporating knowledge graphs (KGs), which can provide rich structured knowledge facts for better language understanding. We argue that informative entities in KGs can enhance language representation with external knowledge. In this paper, we utilize both large-scale textual corpora and KGs to train an enhanced language representation model (ERNIE), which can take full advantage of lexical, syntactic, and knowledge information simultaneously. The experimental results have demonstrated that ERNIE achieves significant improvements on various knowledge-driven tasks, and meanwhile is comparable with the state-of-the-art model BERT on other common NLP tasks. The source code of this paper can be obtained from `https://github.com/thunlp/ERNIE`.

1 Introduction

Pre-trained language representation models, including feature-based (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2017, 2018) and fine-tuning (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2018) approaches, can capture rich language information from text and then benefit many NLP applications. BERT (Devlin et al., 2018), as one of the most recently proposed models, obtains the state-of-the-art results on various NLP applications by simple fine-tuning, including named entity recognition (Sang and De Meulder, 2003), question answering (Rajpurkar et al., 2016; Zellers et al., 2018), natural language inference (Bowman et al., 2015), and text classification (Wang et al., 2018).

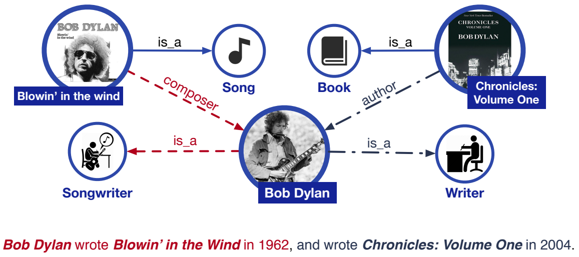

is_a
is_a
Blowin' in the wind
Song
Book
Chronicles: Volume One
composer
author
is_a
is_a
Songwriter
Bob Dylan
Writer
Bob Dylan wrote ***Blowin' in the Wind*** in 1962, and wrote ***Chronicles: Volume One*** in 2004.

Figure 1: An example of incorporating extra knowledge information for language understanding. The solid lines present the existing knowledge facts. The red dotted lines present the facts extracted from the sentence in red. The blue dot-dash lines present the facts extracted from the sentence in blue.

Although pre-trained language representation models have achieved promising results and worked as a routine component in many NLP tasks, they neglect to incorporate knowledge information for language understanding. As shown in Figure 1, without knowing *Blowin' in the Wind* and *Chronicles: Volume One* are *song* and *book* respectively, it is difficult to recognize the two occupations of *Bob Dylan*, i.e., `songwriter` and `writer`, in the entity typing task. Furthermore, it is nearly impossible to extract the fine-grained relations, such as `composer` and `author` in the relation classification task. For the existing pre-trained language representation models, these two sentences are syntactically ambiguous, like "*UNK* wrote *UNK* in *UNK*". Hence, considering rich knowledge information can lead to better language understanding and accordingly benefits various knowledge-driven applications, e.g. entity typing and relation classification.

[*] indicates equal contribution

[†] Corresponding author: Z.Liu(liuzy@tsinghua.edu.cn)

For incorporating external knowledge into language representation models, there are two main challenges. (1) **Structured Knowledge Encoding**: regarding to the given text, how to effectively extract and encode its related informative facts in KGs for language representation models is an important problem; (2) **Heterogeneous Information Fusion**: the pre-training procedure for language representation is quite different from the knowledge representation procedure, which leads to two individual vector spaces. Hence, how to design a special pre-training objective to fuse the lexical, syntactic, and knowledge information is another challenge.

To overcome the challenges mentioned above, we propose **E**nhanced Language **R**epresentatio**N** with **I**nformative **E**ntities (ERNIE), which pre-trains a language representation model on both large-scale textual corpora and knowledge graphs:

(1) For extracting and encoding knowledge information, we firstly recognize named entity mentions in text and then align these mentions to their corresponding entities in KGs. Instead of directly using the graph-based facts in KGs, we encode the graph structure of KGs with knowledge embedding algorithms like TransE (Bordes et al., 2013), and then take the informative entity embeddings as input for ERNIE. Based on the alignments between text and KGs, ERNIE integrates entity representations in the knowledge module into the underlying layers of the semantic module.

(2) Similar to BERT, we adopt the masked language model and the next sentence prediction as the pre-training objectives. Besides, for the better fusion of textual and knowledge features, we design a new pre-training objective by randomly masking some of the named entity alignments in the input text and asking the model to select appropriate entities from KGs to complete the alignments. Unlike the existing pre-trained language representation models only utilizing local context to predict tokens, our objectives require models to aggregate both context and knowledge facts for predicting both tokens and entities, and lead to a knowledgeable language representation model.

We conduct experiments on two knowledge-driven NLP tasks, i.e., entity typing and relation classification. The experimental results show that ERNIE significantly outperforms the state-of-the-art model BERT on these knowledge-driven tasks, by taking full advantage of lexical, syntactic, and knowledge information. We also evaluate ERNIE on other common NLP tasks, and ERNIE still achieves comparable results.

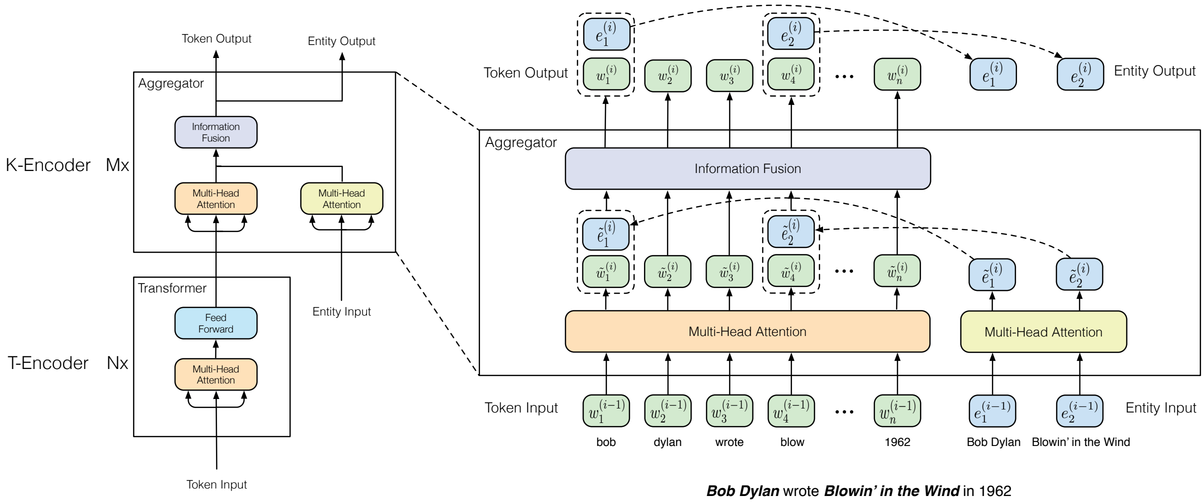
2 Related Work

Many efforts are devoted to pre-training language representation models for capturing language information from text and then utilizing the information for specific NLP tasks. These pre-training approaches can be divided into two classes, i.e., feature-based approaches and fine-tuning approaches.

The early work (Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014) focuses on adopting feature-based approaches to transform words into distributed representations. As these pre-trained word representations capture syntactic and semantic information in the textual corpora, they are often used as input embeddings and initialization parameters for various NLP models, and offer significant improvements over random initialization parameters (Turian et al., 2010). Since these word-level models often suffer from the word polysemy, Peters et al. (2018) further adopt the sequence-level model (ELMo) to capture complex word features across different linguistic contexts and use ELMo to generate context-aware word embeddings.

Different from the above-mentioned feature-based language approaches only using the pre-trained language representations as input features, Dai and Le (2015) train auto-encoders on unlabeled text, and then use the pre-trained model architecture and parameters as a starting point for other specific NLP models. Inspired by Dai and Le (2015), more pre-trained language representation models for fine-tuning have been proposed. Howard and Ruder (2018) present AWD-LSTM (Merity et al., 2017) to build a universal language model (ULMFiT). Radford et al. (2018) propose a generative pre-trained Transformer (Vaswani et al., 2017) (GPT) to learn language representations. Devlin et al. (2018) propose a deep bidirectional model with multiple-layer Transformers (BERT), which achieves the state-of-the-art results for various NLP tasks.

Though both feature-based and fine-tuning language representation models achieve great success, they ignore the incorporation of knowledge information. As demonstrated in recent work, injecting extra knowledge information can



(a) Model Achitecture

(b) Aggregator

Figure 2: The left part is the architecture of ERNIE. The right part is the aggregator for the mutual integration of the input of tokens and entities. Information fusion layer takes two kinds of input: one is the token embedding, and the other one is the concatenation of token embedding and entity embedding. After information fusion, it outputs new token embeddings and entity embeddings for the next layer.

significantly enhance original models, such as reading comprehension (Mihaylov and Frank, 2018; Zhong et al., 2018), machine translation (Zaremoodi et al., 2018), natural language inference (Chen et al., 2018), knowledge acquisition (Han et al., 2018a), and dialog systems (Madotto et al., 2018). Hence, we argue that extra knowledge information can effectively benefit existing pre-training models. In this paper, we utilize both corpora and KGs to train an enhanced language representation model based on BERT.

3 Methodology

In this section, we present the overall framework of ERNIE and its detailed implementation, including the model architecture in Section 3.2, the novel pre-training task designed for encoding informative entities and the fusion of heterogeneous information in Section 3.4, and the details of fine-tuning procedure in Section 3.5.

3.1 Notations

We denote a token sequence as $\{w_1, \ldots, w_n\}$, where n is the length of the token sequence. Meanwhile, we denote the entity sequence aligning to the given tokens as $\{e_1, \ldots, e_m\}$, where m is the length of entity sequence. Note that m is not equal to n in most cases, as not every token can be aligned to an entity in KGs. Furthermore, we denote the whole vocabulary containing all tokens as $\mathcal{V}$, and the entity list containing all entities in KGs as $\mathcal{E}$. If a token $w \in \mathcal{V}$ has a corresponding entity $e \in \mathcal{E}$, their alignment is defined as $f(w) = e$. In this paper, we align an entity to the first token in its named entity phrase, as shown in Figure 2.

3.2 Model Architecture

As shown in Figure 2, the whole model architecture of ERNIE consists of two stacked modules: (1) the underlying textual encoder (`T-Encoder`) responsible to capture basic lexical and syntactic information from the input tokens, and (2) the upper knowledgeable encoder (`K-Encoder`) responsible to integrate extra token-oriented knowledge information into textual information from the underlying layer, so that we can represent heterogeneous information of tokens and entities into a united feature space. Besides, we denote the number of `T-Encoder` layers as N, and the number of `K-Encoder` layers as M.

To be specific, given a token sequence $\{w_1, \ldots, w_n\}$ and its corresponding entity sequence $\{e_1, \ldots, e_m\}$, the textual encoder firstly sums the token embedding, segment embedding, positional embedding for each token to compute its input embedding, and then computes lexical and syntactic features $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$ as follows,

$$\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\} = \texttt{T-Encoder}(\{w_1, \ldots, w_n\}), \quad (1)$$

In this paper, tokens are at subword level.

where T-Encoder$(\cdot)$ is a multi-layer bidirectional Transformer encoder. As T-Encoder$(\cdot)$ is identical to its implementation in BERT and BERT is prevalent, we exclude a comprehensive description of this module and refer readers to Devlin et al. (2018) and Vaswani et al. (2017).

After computing $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_n\}$, ERNIE adopts a knowledgeable encoder K-Encoder to inject the knowledge information into language representation. To be specific, we represent $\{e_1, \dots, e_m\}$ with their entity embeddings $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_m\}$, which are pre-trained by the effective knowledge embedding model TransE (Bordes et al., 2013). Then, both $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_n\}$ and $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_m\}$ are fed into K-Encoder for fusing heterogeneous information and computing final output embeddings,

$$\begin{aligned} \{\boldsymbol{w}_1^o, \dots, \boldsymbol{w}_n^o\}, \{\boldsymbol{e}_1^o, \dots, \boldsymbol{e}_n^o\} = \texttt{K-Encoder}(\\ \{\boldsymbol{w}_1, \dots, \boldsymbol{w}_n\}, \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_m\}). \end{aligned} \tag{2}$$

$\{\boldsymbol{w}_1^o, \dots, \boldsymbol{w}_n^o\}$ and $\{\boldsymbol{e}_1^o, \dots, \boldsymbol{e}_n^o\}$ will be used as features for specific tasks. More details of the knowledgeable encoder K-Encoder will be introduced in Section 3.3.

3.3 Knowledgeable Encoder

As shown in Figure 2, the knowledgeable encoder K-Encoder consists of stacked aggregators, which are designed for encoding both tokens and entities as well as fusing their heterogeneous features. In the i-th aggregator, the input token embeddings $\{\boldsymbol{w}_1^{(i-1)}, \dots, \boldsymbol{w}_n^{(i-1)}\}$ and entity embeddings $\{\boldsymbol{e}_1^{(i-1)}, \dots, \boldsymbol{e}_m^{(i-1)}\}$ from the preceding aggregator are firstly fed into two multi-head self-attentions (MH-ATTs) (Vaswani et al., 2017) respectively,

$$\begin{aligned} \{\tilde{\boldsymbol{w}}_1^{(i)}, \dots, \tilde{\boldsymbol{w}}_n^{(i)}\} &= \texttt{MH-ATT}(\{\boldsymbol{w}_1^{(i-1)}, \dots, \boldsymbol{w}_n^{(i-1)}\}), \\ \{\tilde{\boldsymbol{e}}_1^{(i)}, \dots, \tilde{\boldsymbol{e}}_m^{(i)}\} &= \texttt{MH-ATT}(\{\boldsymbol{e}_1^{(i-1)}, \dots, \boldsymbol{e}_m^{(i-1)}\}). \end{aligned} \tag{3}$$

Then, the i-th aggregator adopts an information fusion layer for the mutual integration of the token and entity sequence, and computes the output embedding for each token and entity. For a token w_j and its aligned entity $e_k = f(w_j)$, the information fusion process is as follows,

$$\begin{aligned} \boldsymbol{h}_j &= \sigma(\tilde{\boldsymbol{W}}_t^{(i)} \tilde{\boldsymbol{w}}_j^{(i)} + \tilde{\boldsymbol{W}}_e^{(i)} \tilde{\boldsymbol{e}}_k^{(i)} + \tilde{\boldsymbol{b}}^{(i)}), \\ \boldsymbol{w}_j^{(i)} &= \sigma(\boldsymbol{W}_t^{(i)} \boldsymbol{h}_j + \boldsymbol{b}_t^{(i)}), \\ \boldsymbol{e}_k^{(i)} &= \sigma(\boldsymbol{W}_e^{(i)} \boldsymbol{h}_j + \boldsymbol{b}_e^{(i)}). \end{aligned} \tag{4}$$

where $\boldsymbol{h}_j$ is the inner hidden state integrating the information of both the token and the entity. $\sigma(\cdot)$ is the non-linear activation function, which usually is the GELU function (Hendrycks and Gimpel, 2016). For the tokens without corresponding entities, the information fusion layer computes the output embeddings without integration as follows,

$$\begin{aligned} \boldsymbol{h}_j &= \sigma(\tilde{\boldsymbol{W}}_t^{(i)} \tilde{\boldsymbol{w}}_j^{(i)} + \tilde{\boldsymbol{b}}^{(i)}), \\ \boldsymbol{w}_j^{(i)} &= \sigma(\boldsymbol{W}_t^{(i)} \boldsymbol{h}_j + \boldsymbol{b}_t^{(i)}). \end{aligned} \tag{5}$$

For simplicity, the i-th aggregator operation is denoted as follows,

$$\begin{aligned} \{\boldsymbol{w}_1^{(i)}, \dots, \boldsymbol{w}_n^{(i)}\}, \{\boldsymbol{e}_1^{(i)}, \dots, \boldsymbol{e}_m^{(i)}\} = \texttt{Aggregator}(\\ \{\boldsymbol{w}_1^{(i-1)}, \dots, \boldsymbol{w}_n^{(i-1)}\}, \{\boldsymbol{e}_1^{(i-1)}, \dots, \boldsymbol{e}_m^{(i-1)}\}). \end{aligned} \tag{6}$$

The output embeddings of both tokens and entities computed by the top aggregator will be used as the final output embeddings of the knowledgeable encoder K-Encoder.

3.4 Pre-training for Injecting Knowledge

In order to inject knowledge into language representation by informative entities, we propose a new pre-training task for ERNIE, which randomly masks some token-entity alignments and then requires the system to predict all corresponding entities based on aligned tokens. As our task is similar to training a denoising auto-encoder (Vincent et al., 2008), we refer to this procedure as a denoising entity auto-encoder (dEA). Considering that the size of $\mathcal{E}$ is quite large for the softmax layer, we thus only require the system to predict entities based on the given entity sequence instead of all entities in KGs. Given the token sequence $\{w_1, \dots, w_n\}$ and its corresponding entity sequence $\{e_1, \dots, e_m\}$, we define the aligned entity distribution for the token w_i as follows,

$$p(e_j|w_i) = \frac{\exp(\texttt{linear}(\boldsymbol{w}_i^o) \cdot \boldsymbol{e}_j)}{\sum_{k=1}^{m} \exp(\texttt{linear}(\boldsymbol{w}_i^o) \cdot \boldsymbol{e}_k)}, \tag{7}$$

where linear$(\cdot)$ is a linear layer. Eq. 7 will be used to compute the cross-entropy loss function for dEA.

Considering that there are some errors in token-entity alignments, we perform the following operations for dEA: (1) In 5% of the time, for a given token-entity alignment, we replace the entity with another random entity, which aims to train our model to correct the errors that the token is aligned with a wrong entity; (2) In 15% of the time, we mask token-entity alignments, which aims to train our model to correct the errors that entity alignment system doesn't extract all existing alignments; (3) In the rest of the time, we keep

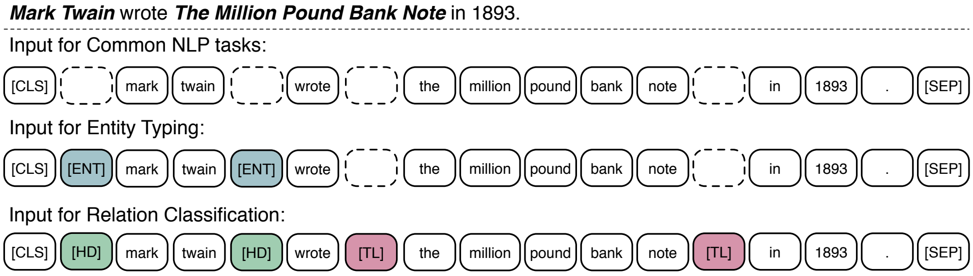

Mark Twain wrote *The Million Pound Bank Note* in 1893.
Input for Common NLP tasks:
[CLS] mark twain wrote the million pound bank note in 1893 . [SEP]
Input for Entity Typing:
[CLS] [ENT] mark twain [ENT] wrote the million pound bank note in 1893 . [SEP]
Input for Relation Classification:
[CLS] [HD] mark twain [HD] wrote [TL] the million pound bank note [TL] in 1893 . [SEP]

Figure 3: Modifying the input sequence for the specific tasks. To align tokens among different types of input, we use dotted rectangles as placeholder. The colorful rectangles present the specific mark tokens.

token-entity alignments unchanged, which aims to encourage our model to integrate the entity information into token representations for better language understanding.

Similar to BERT, ERNIE also adopts masked language model (MLM) and next sentence prediction (NSP) as pre-training tasks to enable ERNIE to capture lexical and syntactic information from tokens in text. More details of these pre-training tasks can be found from Devlin et al. (2018).

3.5 Fine-tuning for Specific Tasks

As shown in Figure 3, for various common NLP tasks, ERNIE can adopt the fine-tuning procedure similar to BERT. We can take the final output embedding for the first token, which corresponds to the special [CLS] token, as the representation of the input sequence for specific tasks. For some knowledge-driven tasks (e.g., relation classification and entity typing), we design special fine-tuning procedure:

For relation classification, the task requires systems to classify relation labels of given entity pairs based on context. The most straightforward way to fine-tune ERNIE for relation classification is to apply the pooling layer to the final output embeddings of the given entity mentions, and represent the given entity pair with the concatenation of their mention embeddings for classification. In this paper, we design another method, which modifies the input token sequence by adding two mark tokens to highlight entity mentions. These extra mark tokens play a similar role like position embeddings in the conventional relation classification models (Zeng et al., 2015). Then, we also take the [CLS] token embedding for classification. Note that we design different tokens **[HD]** and **[TL]** for head entities and tail entities respectively.

The specific fine-tuning procedure for entity typing is a simplified version of relation classification. As previous typing models make full use of both context embeddings and entity mention embeddings (Shimaoka et al., 2016), we argue that the modified input sequence with the mention mark token **[ENT]** can guide ERNIE to combine both context information and entity mention information attentively.

4 Experiments

In this section, we present the detail of pre-training ERNIE and the fine-tuning results on five NLP datasets, which contain both knowledge-driven tasks and the common NLP tasks.

4.1 Pre-training Dataset

The pre-training procedure primarily acts in accordance with the existing literature on pre-training language models. For the large cost of training ERNIE from scratch, we adopt the parameters of BERT released by Google to initialize the Transformer blocks for encoding tokens. Since pre-training is a multi-task procedure consisting of NSP, MLM, and dEA, we use English Wikipedia as our pre-training corpus and align text to Wikidata. After converting the corpus into the formatted data for pre-training, the annotated input has nearly $4,500$M subwords and 140M entities, and discards the sentences having less than 3 entities.

Before pre-training ERNIE, we adopt the knowledge embeddings trained on Wikidata by TransE as the input embeddings for entities. To be specific, we sample part of Wikidata which contains $5,040,986$ entities and $24,267,796$ fact triples. The entity embeddings are fixed during training and the parameters of the entity encoding modules are all initialized randomly.

https://github.com/google-research/bert
https://www.wikidata.org/

Dataset	Train	Develop	Test	Type
FIGER	2,000,000	10,000	563	113
Open Entity	2,000	2,000	2,000	6

Table 1: The statistics of the entity typing datasets FIGER and Open Entity.

Model	Acc.	Macro	Micro
NFGEC (Attentive)	54.53	74.76	71.58
NFGEC (LSTM)	55.60	75.15	71.73
BERT	52.04	75.16	71.63
ERNIE	**57.19**	**76.51**	**73.39**

Table 2: Results of various models on FIGER.

4.2 Parameter Settings and Training Details

In this work, we denote the hidden dimension of token embeddings and entity embeddings as H_w, H_e respectively, and the number of self-attention heads as A_w, A_e respectively. In detail, we have the following model size: $N = 6, M = 6, H_w = 768, H_e = 100, A_w = 12, A_e = 4$, total parameters = 114M.

The total amount of parameters of $\text{BERT}_{\text{BASE}}$ is about 110M, which means the knowledgeable module of ERNIE is much smaller than the language module. Hence, we only pre-train ERNIE on the annotated corpus for one epoch. To accelerate the training process, we reduce the max sequence length from 512 to 256 as the computation of self-attention is a quadratic function of the length. To keep the number of tokens in a batch as same as BERT, we double the batch size to 512. Except for setting the learning rate as $5e^{-5}$, we largely follow the pre-training hyper-parameters used in BERT. For fine-tuning, most hyper-parameters are the same as pre-training, except batch size, learning rate, and number of training epochs. We find the following ranges of possible values work well on the training datasets with gold annotations, i.e., batch size: 32, learning rate (Adam): $5e^{-5}$, $3e^{-5}$, $2e^{-5}$, number of epochs ranging from 3 to 10.

We also fine-tune ERNIE on the distant supervised dataset, i.e., FIGER (Ling et al., 2015). As the powerful expression ability of deeply stacked Transformer blocks, we found small batch size would lead the model to overfit the training data. Hence, we use lager batch size and less training epochs to avoid overfitting, and keep the range of learning rate unchanged, i.e., batch size: 2048, number of epochs: 2, 3.

As most datasets do not have entity annotations, we use TAGME (Ferragina and Scaiella, 2010) to extract the entity mentions in the sentences and link them to their corresponding entities in KGs.

Model	P	R	F1
NFGEC (LSTM)	68.80	53.30	60.10
UFET	77.40	60.60	68.00
BERT	76.37	70.96	73.56
ERNIE	**78.42**	**72.90**	**75.56**

Table 3: Results of various models on Open Entity.

4.3 Entity Typing

Given an entity mention and its context, entity typing requires systems to label the entity mention with its respective semantic types. To evaluate performance on this task, we fine-tune ERNIE on two well-established datasets FIGER (Ling et al., 2015) and Open Entity (Choi et al., 2018). The training set of FIGER is labeled with distant supervision, and its test set is annotated by human. Open Entity is a completely manually-annotated dataset. The statistics of these two datasets are shown in Table 1. We compare our model with the following baseline models for entity typing:

NFGEC. NFGEC is a hybrid model proposed by Shimaoka et al. (2016). NFGEC combines the representations of entity mention, context and extra hand-craft features as input, and is the state-of-the-art model on FIGER. As this paper focuses on comparing the general language representation abilities of various neural models, we thus do not use the hand-craft features in this work.

UFET. For Open Entity, we add a new hybrid model UFET (Choi et al., 2018) for comparison. UFET is proposed with the Open Entity dataset, which uses a Bi-LSTM for context representation instead of two Bi-LSTMs separated by entity mentions in NFGEC.

Besides NFGEC and UFET, we also report the result of fine-tuning BERT with the same input format introduced in Section 3.5 for fair comparison. Following the same evaluation criteria used in the previous work, we compare NFGEC, BERT, ERNIE on FIGER, and adopt strict accuracy, loose macro, loose micro scores for evaluation. We compare NFGEC, BERT, UFET, ERNIE on Open Entity, and adopt precision, recall, micro-F1 scores for evaluation.

The results on FIGER are shown in Table 2. From the results, we observe that: (1) BERT

Dataset	Train	Develop	Test	Relation
FewRel	8,000	16,000	16,000	80
TACRED	68,124	22,631	15,509	42

Table 4: The statistics of the relation classification datasets FewRel and TACRED.

achieves comparable results with NFGEC on the macro and micro metrics. However, BERT has lower accuracy than the best NFGEC model. As strict accuracy is the ratio of instances whose predictions are identical to human annotations, it illustrates some wrong labels from distant supervision are learned by BERT due to its powerful fitting ability. (2) Compared with BERT, ERNIE significantly improves the strict accuracy, indicating the external knowledge regularizes ERNIE to avoid fitting the noisy labels and accordingly benefits entity typing.

The results on Open Entity are shown in Table 3. From the table, we observe that: (1) BERT and ERNIE achieve much higher recall than the previous entity typing models, which means pre-training language models make full use of both the unsupervised pre-training and manually-annotated training data for better entity typing. (2) Compared to BERT, ERNIE improves the precision by 2% and the recall by 2%, which means the informative entities help ERNIE predict the labels more precisely.

In summary, **ERNIE effectively reduces the noisy label challenge in FIGER, which is a widely-used distantly supervised entity typing dataset, by injecting the information from KGs.** Besides, ERNIE also outperforms baselines on Open Entity which has gold annotations.

4.4 Relation Classification

Relation classification aims to determine the correct relation between two entities in a given sentence, which is an important knowledge-driven NLP task. To evaluate performance on this task, we fine-tune ERNIE on two well-established datasets FewRel (Han et al., 2018b) and TACRED (Zhang et al., 2017). The statistics of these two datasets are shown in Table 4. As the original experimental setting of FewRel is few-shot learning, we rearrange the FewRel dataset for the common relation classification setting. Specifically, we sample 100 instances from each class for the training set, and sample 200 instances for the development and testing respectively. There are 80 classes in FewRel, and there are 42 classes (including a special relation *"no relation"*) in TACRED. We compare our model with the following baseline models for relation classification:

Model	FewRel			TACRED		
	P	R	F1	P	R	F1
CNN	69.51	69.64	69.35	70.30	54.20	61.20
PA-LSTM	-	-	-	65.70	64.50	65.10
C-GCN	-	-	-	69.90	63.30	66.40
BERT	85.05	85.11	84.89	67.23	64.81	66.00
ERNIE	88.49	88.44	**88.32**	69.97	66.08	**67.97**

Table 5: Results of various models on FewRel and TACRED.

CNN. With a convolution layer, a max-pooling layer, and a non-linear activation layer, CNN gets the output sentence embedding, and then feeds it into a relation classifier. To better capture the position of head and tail entities, position embeddings are introduced into CNN (Zeng et al., 2015).

PA-LSTM. Zhang et al. (2017) propose PA-LSTM introducing a position-aware attention mechanism over an LSTM network, which evaluates the relative contribution of each word in the sequence for the final sentence representation.

C-GCN. Zhang et al. (2018) adopt the graph convolution operations to model dependency trees for relation classification. To encode the word order and reduce the side effect of errors in dependency parsing, Contextualized GCN (C-GCN) firstly uses Bi-LSTM to generate contextualized representations as input for GCN models.

In addition to these three baselines, we also fine-tune BERT with the same input format introduced in Section 3.5 for fair comparison.

As FewRel does not have any null instance where there isn't any relation between entities, we adopt macro averaged metrics to present the model performances. Since FewRel is built by checking whether the sentences contain facts in Wikidata, we drop the related facts in KGs before pre-training for fair comparison. From Table 5, we have two observations: (1) As the training data does not have enough instances to train the CNN encoder from scratch, CNN just achieves an F1 score of 69.35%. However, the pre-training models including BERT and ERNIE increase the F1 score by at least 15%. (2) ERNIE achieves an absolute F1 increase of 3.4% over BERT, which means fusing external knowledge is very effective.

Model	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
	392k	363k	104k	67k	8.5k	5.7k	3.5k	2.5k
BERT$_{\text{BASE}}$	84.6/83.4	71.2	-	93.5	52.1	85.8	88.9	66.4
ERNIE	84.0/83.2	71.2	91.3	93.5	52.3	83.2	88.2	68.8

Table 6: Results of BERT and ERNIE on different tasks of GLUE.

Model	P	R	F1
BERT	85.05	85.11	84.89
ERNIE	88.49	88.44	**88.32**
w/o entities	85.89	85.89	85.79
w/o dEA	85.85	85.75	85.62

Table 7: Ablation study on FewRel.

In TACRED, there are nearly 80% null instances so that we follow the previous work (Zhang et al., 2017) to adopt micro averaged metrics to represent the model performances instead of the macro. The results of CNN, PA-LSTM, and C-GCN come from the paper by Zhang et al. (2018), which are the best results of CNN, RNN, and GCN respectively. From Table 5, we observe that: (1) The C-GCN model outperforms the strong BERT model by an F1 increase of 0.4%, as C-GCN utilizes the dependency trees and the entity mask strategy. The entity mask strategy refers to replacing each subject (and object similarly) entity with a special NER token, which is similar to our proposed pre-training task dEA. (2) ERNIE achieves the best recall and F1 scores, and increases the F1 of BERT by nearly 2.0%, which proves the effectiveness of the knowledgeable module for relation classification.

In conclusion, we find that the pre-trained language models can provide more information for relation classification than the vanilla encoder CNN and RNN. And ERNIE outperforms BERT on both of the relation classification datasets, especially on the FewRel which has a much smaller training set. **It demonstrates extra knowledge helps the model make full use of small training data, which is important for most NLP tasks as large-scale annotated data are unavailable.**

4.5 GLUE

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of diverse natural language understanding tasks, which is the main benchmark used in Devlin et al. (2018). To explore whether our knowledgeable module degenerates the performance on common NLP tasks, we evaluate ERNIE on 8 datasets of GLUE and compare it with BERT.

In Table 6, we report the results of our evaluation submissions and those of BERT from the leaderboard. We notice that ERNIE is consistent with BERT$_{\text{BASE}}$ on big datasets like MNLI, QQP, QNLI, and SST-2. The results become more unstable on small datasets, that is, ERNIE is better on CoLA and RTE, but worse on STS-B and MRPC.

In short, ERNIE achieves comparable results with BERT$_{\text{BASE}}$ on GLUE. On the one hand, it means GLUE does not require external knowledge for language representation. On the other hand, it illustrates ERNIE does not lose the textual information after heterogeneous information fusion.

4.6 Ablation Study

In this subsection, we explore the effects of the informative entities and the knowledgeable pre-training task (dEA) for ERNIE using FewRel dataset. **w/o entities** refers to fine-tuning ERNIE without entity sequence input. **w/o dEA** refers to fine-tuning the ERNIE without the pre-training task dEA. As shown in Table 7, we have the following observations: (1) Without entity sequence input, dEA still injects knowledge information into language representation during pre-training, which increases the F1 score of BERT by 0.9%. (2) Although the informative entities bring much knowledge information which intuitively benefits relation classification, ERNIE without dEA takes little advantage of these, leading to the F1 increase of 0.7%.

5 Conclusion

In this paper, we propose ERNIE to incorporate knowledge information into language representation models. Accordingly, we propose the knowledgeable aggregator and pre-training task dEA for better fusion of heterogeneous information from both text and KGs. The experimental results demonstrate that ERNIE has better abilities of both denoising distantly supervised data and fine-tuning on limited data than BERT. There are three important directions remain for future research: (1) inject knowledge into feature-based pre-training models such as ELMo (Peters et al., 2018); (2) introduce diverse structured knowledge into language representation models such as ConceptNet (Speer and Havasi, 2012) which is different from world knowledge database Wikidata; (3) annotate more real-world corpora heuristically for larger pre-training data.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of ACL*, pages 2406–2417.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of ACL*, pages 87–96.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Proceedings of NIPS*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM*, pages 1625–1628.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018a. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

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

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL*, pages 328–339.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *TACL*, 3:315–328.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of ACL*, pages 1468–1478.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of ACL*, pages 821–832.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL*, pages 1756–1765.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Proceedings of Technical report, OpenAI*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of NAACL*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of AKBC 2016*, pages 69–74.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *Proceedings of LREC*, pages 3679–3686.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of ICML*, pages 1096–1103.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Poorya Zaremoodi, Wray Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: Improving low-resource neural machine translation. In *Proceedings of ACL*, pages 656–661.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of EMNLP*, pages 93–104.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*, pages 2205–2215.

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 EMNLP*, pages 35–45.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018. Improving question answering by commonsense-based pre-training. *arXiv preprint arXiv:1809.03568*.