JAKET: Joint Pre-training of Knowledge Graph and Language Understanding

Donghan Yu[1*], **Chenguang Zhu**[2*], **Yiming Yang**[1], **Michael Zeng**[2]
[1] Carnegie Mellon University
[2] Microsoft Cognitive Services Research Group
dyu2@cs.cmu.edu, chezhu@microsoft.com

Abstract

Knowledge graphs (KGs) contain rich information about world knowledge, entities, and relations. Thus, they can be great supplements to existing pre-trained language models. However, it remains a challenge to efficiently integrate information from KG into language modeling. And the understanding of a knowledge graph requires related context. We propose a novel joint pre-training framework, JAKET, to model both the knowledge graph and language. The knowledge module and language module provide essential information to mutually assist each other: the knowledge module produces embeddings for entities in text while the language module generates context-aware initial embeddings for entities and relations in the graph. Our design enables the pre-trained model to easily adapt to unseen knowledge graphs in new domains. Experiment results on several knowledge-aware NLP tasks show that our proposed framework achieves superior performance by effectively leveraging knowledge in language understanding.

Introduction

Pre-trained language models (PLM) leverage large-scale unlabeled corpora to conduct self-supervised training. They have achieved remarkable performance in various NLP tasks, exemplified by BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019b), XLNet (Yang et al. 2019), and GPT series (Radford et al. 2018, 2019; Brown et al. 2020). It has been shown that PLMs can effectively characterize linguistic patterns in text and generate high-quality context-aware representations (Liu et al. 2019a). However, these models struggle to grasp world knowledge about entities and relations (Poerner, Waltinger, and Schütze 2019; Talmor et al. 2019), which are clearly important for language understanding.

Knowledge graphs (KGs) represent entities and relations in a structural way. They can also solve the sparsity problem in text modeling. For instance, a language model may require many instances of the phrase "labrador is a kind of dog" in its training corpus before it implicitly learns this fact. In comparison, a knowledge graph can use two entity nodes "labrador", "dog" and a relation edge "is_a" between these nodes to precisely represent this fact.

Recently, some efforts have been made to integrate knowledge graphs into PLM. Most of them combine the token representations in PLM with embeddings of aligned KG entities. The entity embeddings in those methods are either pre-computed based on an external source by a separate model (Zhang et al. 2019; Peters et al. 2019), which may not be easily aligned with the PLM embeddings, or directly learned as model parameters (Févry et al. 2020; Verga et al. 2020), which often have an over-parameterization issue due to the large number of entities. Moreover, all the previous works share a common challenge: when the pre-trained model is fine-tuned in a new domain with a previously unseen knowledge graph, it struggles to adapt to the new entities, relations and structure.

Therefore, we propose JAKET, a Joint pre-trAining framework for KnowledgE graph and Text. Our framework contains a knowledge module and a language module, which mutually assist each other by providing required information to achieve more effective semantic analysis. The knowledge module is based on a graph attention network (Velickovic et al. 2018) to provide structure-aware entity embeddings for language modeling. And the language module produces contextual representations as initial embeddings for KG entities and relations given their descriptive text. Thus, in both modules, content understanding is based on related knowledge and rich context. On one hand, the joint pre-training effectively projects entities/relations and text into a shared semantic latent space, which eases the semantic matching between them. On the other hand, as the knowledge module produces representations from descriptive text, it solves the over-parameterization issue since entity embeddings are no longer part of the model's parameters.

In order to solve the cyclic dependency between the two modules, we propose a novel two-step language module LM_1 and LM_2, respectively. LM_1 provides embeddings for both LM_2 and KG. The entity embeddings from KG are also fed into LM_2, which produces the final representation. LM_1 and LM_2 can be easily established as the first several transformer layers and the rest layers of a pre-trained language model such as BERT and RoBERTa. Furthermore, we design an entity context embedding memory with periodic update which speeds up the pre-training by about 15x.

The pre-training tasks are all self-supervised, including entity category classification and relation prediction for the

[*]Equal contribution. Work done while Donghan Yu was an intern at Microsoft.

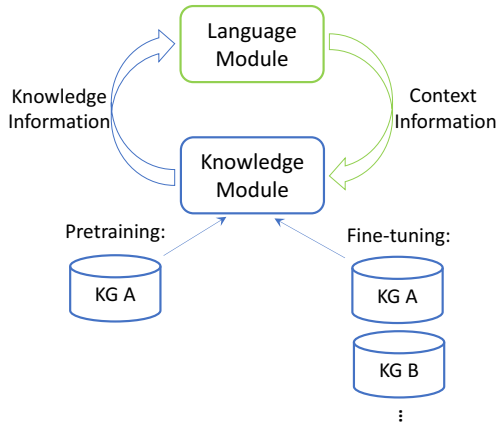

Figure 1: A simple illustration on the novelty of our proposed model JAKET: (1) The language module and knowledge module mutually assist each other; (2) JAKET can adapt to unseen knowledge graphs during fine-tuning.

knowledge module, and masked token prediction and masked entity prediction for the language module.

A great benefit of our framework is that it can easily adapt to unseen knowledge graphs in the fine-tuning phase. As the initial embeddings of entities and relations come from their descriptive text, JAKET is not confined to any fixed KG. With the learned ability to integrate structural information during pre-training, the framework is extensible to novel knowledge graphs with previously unseen entities and relations, as illustrated in Fig.1.

We conduct empirical studies on several knowledge-aware natural language understanding (NLU) tasks, including few-shot relation classification, question answering over KG and entity classification. The results show that JAKET achieves the best performance compared with strong baseline methods on all the tasks, including those with a previously unseen knowledge graph.

Related Work

Pre-trained language models have been shown to be very effective in various NLP tasks, including ELMo (Peters et al. 2018), GPT (Radford et al. 2018), BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019b) and XLNet (Yang et al. 2019). Built upon large-scale corpora, these pretrained models learn effective representations for various semantic structures and linguistic relationships. They are trained on self-supervised tasks like masked language modeling and next sentence prediction.

Recently, a lot of efforts have been made on investigating how to integrate knowledge into PLMs (Levine et al. 2020; Baldini Soares et al. 2019; Liu et al. 2020; Guu et al. 2020; Xu et al. 2021; Wang et al. 2021; Yu et al. 2020). These approaches can be grouped into two categories:

a. Explicitly injecting entity representation into the language model, where the representations are either pre-computed from external sources (Yu et al. 2021; Peters et al. 2019) or directly learned as model parameters (Fang et al. 2021; Verga et al. 2020). For example, ERNIE (THU) (Zhang et al. 2019) pre-trains the entity embeddings on a knowledge graph using TransE (Bordes et al. 2013), while EAE (Févry et al. 2020) learns the representation from pre-training objectives with all the other model parameters. K-BERT (Liu et al. 2020) represents the entities by the embeddings of surface form tokens (i.e. entity names), which contains much less semantic information compared with description text. Moreover, it only injects KG during fine-tuning phase instead of joint-pretraining KG and text. The most related work is CoLAKE (Sun et al. 2020a). However, there are two main differences between JAKET and CoLAKE: (1) The entity embeddings of CoLAKE are initialized by a fixed language model, while JAKET initializes entity embedding by a dynamically updated language model, which allows better adaptation on unseen entities; (2) CoLAKE uses a shared Transformer to model both text and KG, while JAKET contains two separate modules, with more model flexibility.

b. Implicitly modeling knowledge information, including entity-level masked language modeling (Sun et al. 2019; Shen et al. 2020), entity-based replacement prediction (Xiong et al. 2020) and knowledge embedding loss as regularization (Wang et al. 2019b). For example, besides token-level masked language modeling, ERNIE (Baidu) (Sun et al. 2019) uses phrase-level and entity-level masking to predict all the masked slots. KEPLER (Wang et al. 2019b) calculates entity embeddings using a pre-trained language model based on the description text, which is similar to our work. However, they use the entity embeddings for the knowledge graph completion task instead of injecting them into the language model.

Some works (Ding et al. 2019; Lv et al. 2020) investigated the combination of graph neural network (GNN) and PLM. For example, (Lv et al. 2020) uses XLNet to generate initial node representation based on node context and feeds them into a GNN. However, these approaches do not integrate knowledge into language modeling, and they are designed for specific NLP tasks such as reading comprehension or commonsense reasoning. In comparison, we jointly pre-train both the knowledge graph representation and language modeling and target for general knowledge-aware NLU tasks.

Method

In this section, we introduce the JAKET framework of joint pre-training knowledge graph and language understanding. We begin by defining the mathematical notations, and then present our model architecture with the knowledge module and language module. Finally, we introduce how to pre-train our model and fine-tune it for downstream tasks. The framework is illustrated in Fig.2.

Definition

A knowledge graph is denoted by $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E} = \{e_1 \dots e_N\}$ is the set of entities and $\mathcal{R} = \{r_1 \dots r_P\}$ is the set of relations. $\mathcal{T} = \{(e_{t_i^1}, r_{t_i^2}, e_{t_i^3}) | 1 \leq i \leq T, e_{t_i^1}, e_{t_i^3} \in \mathcal{E}, r_{t_i^2} \in \mathcal{R}\}$ stands for the set of head-relation-tail triplets. $N_v = \{(r, u) | (v, r, u) \in \mathcal{T}\}$ represents the set of neighboring relations and entities of an entity v.

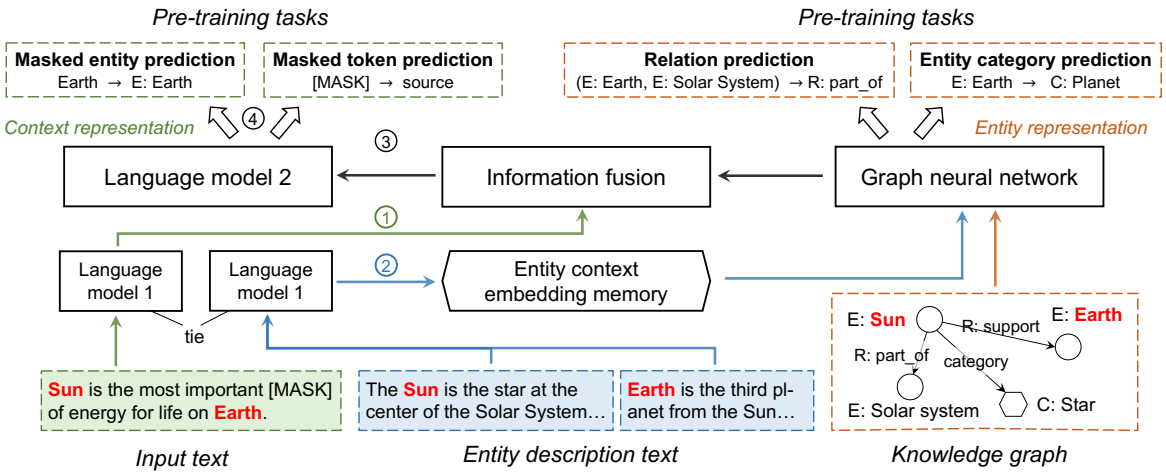

Figure 2: A demonstration for the structure of JAKET, where the language module is on the left side while the knowledge module is on the right side. Symbol ⓘ indicates the steps to compute context representations. "E:", "R:" and "C:" stand for Entities, Relations and Categories in KG respectively. Entity mentions in text are marked red and bold such as **Sun**.

We define $\mathcal{V} = \{[\text{MASK}], [\text{CLS}], [\text{EOS}], w_1 \ldots w_V\}$ as a vocabulary of tokens and the contextual text $\mathbf{x} = [x_1, x_2, \ldots, x_L]$ as a sequence of tokens where $x_i \in \mathcal{V}$. In the vocabulary, [MASK] is the special token for masked language modeling (Devlin et al. 2019) and [CLS], [EOS] are the special tokens indicating the beginning and end of the sequence. We define F as the dimension of token embeddings, which is equal to the dimension of entity/relation embeddings from the KG.

The text $\mathbf{x}$ has a list of entity mentions $\mathbf{m} = [m_1, \ldots, m_M]$, where each mention $m_i = (e_{m_i}, s_{m_i}, o_{m_i})$: e_{m_i} is the corresponding entity and s_{m_i}, o_{m_i} are the start and end index of this mention in the context. In other words, $[x_{s_{m_i}}, \ldots, x_{o_{m_i}}]$ is linked with entity e_{m_i}[1]. We assume the span of mentions are disjoint for a given text sequence.

As entities in the knowledge graph are represented by nodes without context, we use *entity description text* to describe the concept and meaning of entities. For each entity e_i, its description text $\mathbf{x}^{e_i}$ describes this entity. The mention of e_i in $\mathbf{x}^{e_i}$ is denoted as $m^{e_i} = (e_i, s_i^e, o_i^e)$, similarly defined as above. For instance, the description text for the entity "sun" can be "[CLS] The Sun is the star at the center of the Solar System [EOS]". Then the mention is $m^{Sun} = (Sun, 3, 3)$. If there are multiple mentions of e_i in its description text, we choose the first one. If there's no mention of e_i in its description text, we set $s_i^e = o_i^e = 1$. Similarly, we define *relation description text* as the text that can describe each relation.

Knowledge Module

The goal of the knowledge module (KM) is to model the knowledge graph to generate knowledge-based entity representations.

To compute entity node embeddings, we employ the Graph Attention Network (GAT) (Velickovic et al. 2018)[2], which uses the self-attention mechanism to specify different weights for different neighboring nodes. However, the vanilla GAT is designed for homogeneous graphs with single-relation edges. To leverage the multi-relational information, we adopt the idea of composition operator (Vashishth et al. 2020) to compose entity embeddings and relation embeddings. In detail, in the l-th layer of KM, we update the embedding $E_v^{(l)}$ of entity v as follows: First for each relation entity pair $(r, u) \in \mathcal{N}_v$, we combine the embedding of entity u with the embedding of relation r:

$$M_{u,r}^{(l-1)} = f(E_u^{(l-1)}, R_r) \tag{1}$$

Note that the relation embedding R_r is shared across different layers. The function $f(\cdot,\cdot) : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^F$ merges a pair of entity and relation embeddings into one representation. Here, we set $f(x, y) = x + y$ inspired by TransE (Bordes et al. 2013). More complicated functions like MLP network can also be applied. Then the combined embeddings are aggregated by graph attention mechanism:

$$m_v^k = \sigma\left(\sum_{(r,u)\in\mathcal{N}_v} \alpha_{v,r,u}^k W^k M_{u,r}^{(l-1)}\right) \tag{2}$$

where k is the index of attention head and W^k is the model parameter. The attention score $\alpha_{v,r,u}^k$ is calculated by:

$$S_{u,r} = \mathbf{a}^T \left[W^k E_v^{(l-1)} \oplus W^k M_{u,r}^{(l-1)}\right] \tag{3}$$

$$\alpha_{v,r,u}^k = \frac{\exp\left(\text{LeakyReLU}\ (S_{u,r})\right)}{\sum_{(r',u')\in\mathcal{N}_v} \exp\left(\text{LeakyReLU}\ (S_{u',r'})\right)} \tag{4}$$

[1] We do not consider discontinous entity mentions in this work.

[2] We also tried Graph Convolutional Network (GCN) (Kipf and Welling 2017) but it performs worse than GAT on the pre-training tasks.

Finally the embedding of entity v is updated through combining the message representation m_v^k and its embedding in layer $(l-1)$:

$$E_v^{(l)} = \text{LayerNorm}\left(\bigoplus_{k=1}^{K} m_v^k + E_v^{(l-1)}\right) \quad (5)$$

where LayerNorm stands for layer normalization (Ba, Kiros, and Hinton 2016). $\bigoplus$ means concatenation and K is the number of attention heads.

The initial entity embeddings $E^{(0)}$ and relation embeddings R are generated from our language module, which will be introduced in Section "Solving the Cyclic Dependency". Then, the output entity embeddings from the last GAT layer are used as the final entity representations E^{KM}. Note that the knowledge graph can be very large, making the embedding update over all the entities not tractable. Thus we follow the minibatch setting (Hamilton, Ying, and Leskovec 2017): given a set of input entities, we perform neighborhood sampling to generate their multi-hop neighbor sets and we compute representations only on the entities and relations that are necessary for the embedding update.

Language Module

The goal of the language module (LM) is to model text data and learn context-aware representations. The language module can be any model for language understanding, e.g. BERT (Devlin et al. 2019). In this work, we use the pre-trained model RoBERTa-base (Liu et al. 2019b) as the language module.

Solving the Cyclic Dependency

In our framework, the knowledge and language modules mutually benefit each other: the language module LM outputs context-aware embedding to initialize the embeddings of entities and relations in the knowledge graph given the description text; the knowledge module (KM) outputs knowledge-based entity embeddings for the language module.

However, there exists a cyclic dependency which prevents computation and optimization in this design. To solve this problem, we propose a decomposed language module which includes two language models: LM_1 and LM_2. We employ the first 6 layers of RoBERTa as LM_1 and the remaining 6 layers as LM_2. The computation proceeds as follows:

1. LM_1 operates on the input text $\mathbf{x}$ and generates contextual embeddings Z.
2. LM_1 generates initial entity and relation embeddings for KM given description text.
3. KM produces its output entity embeddings to be combined with Z and sent into LM_2.
4. LM_2 produces the final embeddings of $\mathbf{x}$, which includes both contextual and knowledge information.

In detail, in step 1, suppose the context $\mathbf{x}$ is embedded as X^{embed}. LM_1 takes X^{embed} as input and outputs hidden representations:

$$Z = \text{LM}_1(X^{embed}) \quad (6)$$

In step 2, suppose $\mathbf{x}^{e_j}$ is the *entity description text* for entity e_j, and the corresponding mention is $m^{e_j} = (e_j, s_j^e, o_j^e)$. LM_1 takes the embedding of $\mathbf{x}^{e_j}$ and produces the contextual embedding Z^{e_j}. Then, the average of embeddings at position s_j^e and o_j^e is used as the initial entity embedding of e_j, i.e.

$$E_j^{(0)} = (Z_{s_j^e}^{e_j} + Z_{o_j^e}^{e_j})/2 \quad (7)$$

The knowledge graph relation embeddings R are generated in a similar way using its description text.

In step 3, KM computes the final entity embeddings E^{KM}, which is then combined with the output Z from LM_1. In detail, suppose the mentions in $\mathbf{x}$ are $\mathbf{m} = [m_1, \ldots, m_M]$. Z and E^{KM} are combined at positions of mentions: for each position index k, if $\exists i \in \{1, 2, \ldots, M\}$ s.t. $s_{m_i} \leq k \leq o_{m_i}$,

$$Z_k^{merge} = Z_k + E_{e_{m_i}}^{\text{KM}} \quad (8)$$

where $E_{e_{m_i}}^{\text{KM}}$ is the output embedding of entity e_{m_i} from KM. For other positions which do not have corresponding mentions, we keep the original embeddings: $Z_k^{merge} = Z_k$. Then we apply layer normalization (Ba, Kiros, and Hinton 2016) on Z^{merge}:

$$Z' = \text{LayerNorm}(Z^{merge}) \quad (9)$$

Finally, Z' is fed into LM_2.

In step 4, LM_2 operates on the input Z' and obtains the final embeddings:

$$Z^{\text{LM}} = \text{LM}_2(Z') \quad (10)$$

The four steps are marked by the symbol ⊗ in Fig.2 for better illustration.

Entity Context Embedding Memory

Many knowledge graphs contain a large number of entities. Thus, even for one sentence, the number of entities plus their multi-hop neighbors can grow exponentially with the number of layers in the graph neural network. As a result, it's very time-consuming for the language module to compute context embeddings based on the description text of all involved entities in a batch on the fly.

To solve this problem, we construct an entity context embedding memory, $E^{context}$, to store the initial embeddings of all KG entities. Firstly, the language module pre-computes the context embeddings for all entities and places them into the memory. The knowledge module only needs to retrieve required embeddings from the memory instead of computing them, i.e. $E^{(0)} \leftarrow E^{context}$.

However, as embeddings in the memory are computed from the "old" (initial) language module while the token embeddings during training are computed from the updated language module, there will be an undesired discrepancy. Thus, we propose to update the whole embedding memory $E^{context}$ with the current language module every $T(i)$ steps, where i is the number of times that the memory has been updated (starting from 0). $T(i)$ is set as follows:

$$T(i) = \min(I_{init} * a^{\lfloor i/r \rfloor}, I_{max}) \quad (11)$$

where I_{init} is the initial number of steps before the first update and a is the increasing ratio of updating intervals. r is the number of repeated times of the current updating interval. I_{max} is the maximum number of steps between updates. $\lfloor \cdot \rfloor$ means the operation of rounding down. In our experiments, we set $I_{init} = 10, a = 2, r = 3, I_{max} = 500$, and the corresponding sequence of T is $[10, 10, 10, 20, 20, 20, 40, 40, 40, \ldots, 500, 500]$. Note that we choose $a > 1$ because the model parameters usually change less as training proceeds.

Moreover, we propose a momentum update to make $E^{context}$ evolve more smoothly. Suppose the newly calculated embedding memory by LM is $E^{context}_{new}$, then the updating rule is:

$$E^{context} \leftarrow mE^{context} + (1-m)E^{context}_{new} \tag{12}$$

where $m \in [0, 1)$ is a momentum coefficient which is set as 0.8 in experiment.

This memory design speeds up our model by about 15x during pre-training while keeping the effectiveness of entity context embeddings. For consideration of efficiency, we use relation embeddings only during fine-tuning.

Pre-training

During pre-training, both the knowledge module and language module are optimized based on several self-supervised learning tasks listed below. The examples of all the training tasks are shown in Fig.2.

At each pre-training step, we first sample a batch of root entities and perform random-walk sampling on each root entity. The sampled entities are fed into KM for the following two tasks.

Entity category prediction. The knowledge module is trained to predict the category label of entities based on the output entity embeddings E^{KM}. This task has been demonstrated to be effective in pre-training graph neural networks (Hu et al. 2020). The loss function is cross-entropy for multi-class classification, denoted as $\mathcal{L}_c$.

Relation prediction. KM is also trained to predict the relation between a given entity pair based on E^{KM}. The loss function is cross-entropy for multi-class classification, denoted as $\mathcal{L}_r$.

Then, we uniformly sample a batch of text sequences and their entities for the following two tasks.

Masked token prediction. Similar to BERT, We randomly mask tokens in the sequence and predict the original tokens based on the output Z^{LM} of the language module. We denote the loss as $\mathcal{L}_t$.

Masked entity prediction. The language module is also trained to predict the corresponding entity of a given mention. For the input text, we randomly remove 15% of the mentions $\mathbf{m}$. Then for each removed mention $m_r = (e_r, s_r, o_r)$, the model predicts the masked entity e_r based on the mention's embedding. In detail, it predicts the entity whose embedding in $E^{context}$ is closest to $q = g((Z^{\text{LM}}_{s_r} + Z^{\text{LM}}_{o_r})/2)$, where $g(x) = \text{GELU}(xW_1)W_2$ is a transformation function. GELU is an activation function proposed by (Hendrycks and Gimpel 2016). Since the number of entities can be very large, we use e_r's neighbours and other randomly sampled entities as negative samples. The loss function $\mathcal{L}_e$ is cross entropy based on the inner product between q and each candidate entity's embedding. Fig.2 shows an concrete example, where the mention "Earth" is not marked in the input text since it's masked and the task is to link the mention "Earth" to entity "Q2: Earth".

Fine-tuning

During fine-tuning, our model supports using either the knowledge graph employed during pre-training or a novel custom knowledge graph with previously unseen entities[3]. If a custom KG is used, the entity context embedding memory is recomputed by the pre-trained language module using the new entity description text.

Our model also supports KG-only tasks such as entity classification or link prediction where the input data are entity description text and KG without context corpus. In this case, the Language Model 1 takes entity description text as input and output entity embeddings into the knowledge module (i.e. graph neural network) for downstream tasks. The Language Model 2 will not be used.

In this work, we do not update the entity context memory during fine-tuning for consideration of efficiency. We also compute the relation context embedding memory using the pre-trained language model.

Experiment

Basic Settings

Data for Pre-training. We use the English Wikipedia as the text corpus, Wikidata (Vrandečić and Krötzsch 2014) as the knowledge graph, and SLING (Ringgaard, Gupta, and Pereira 2017) to identify entity mentions. For each entity, we use the first 64 consecutive tokens of its Wikipedia page as its description text and we filter out entities without a corresponding Wikipedia page. We also remove entities that have fewer than 5 neighbors in the Wikidata KG and fewer than 5 mentions in the Wikipedia corpus. The final knowledge graph contains 3,657,658 entities, 799 relations and 20,113,978 triplets. We use the *instance of* relation to find the category of each entity. In total, 3,039,909 entities have category labels of 19,901 types. The text corpus contains about 4 billion tokens.

Implementation Details. We initialize the language module with the pre-trained RoBERTa-base (Liu et al. 2019b) model. The knowledge module is initialized randomly. Our implementation is based on the HuggingFace framework (Wolf et al. 2019) and DGL (Wang et al. 2019a). For the knowledge module, we grid-search the number of layers within [1, 2] (we do not consider more than 2 layers due to model efficiency), the number of attention heads in GAT in [1, 4, 8, 12]. We choose the best performing hyper-parameters based on the validation loss of pre-training tasks after training for 1 epoch: the number of layers is 2 and the number of attention heads is 8. The number of sampled neighbors in each

[3]We assume the custom domain comes with NER and entity linking tools which can annotate entity mentions in text. The training of these systems is beyond the scope of this work.

hop is 10. The dimension of hidden states in the knowledge module is 768, the same as the language module. The number of parameters of the whole model is 111M, which is almost the same as RoBERTa-base.

During pre-training, the batch size and length of text sequences are 1024 and 512 respectively. The batch size of KG entities is 16,384. The number of training epochs is 8. JAKET is optimized by AdamW (Loshchilov and Hutter 2019) using the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon =$ 1e-8, and weight decay of 0.01. The learning rate of the language module is warmed up over the first 3,000 steps to a peak value of 1e-5, and then linearly decayed. The learning rate of our knowledge module starts from 1e-4 and then linearly decayed. The computing infrastructure we use is the NVIDIA V100 GPU in all the experiments.

Baselines. We compare our proposed model JAKET with the pre-trained RoBERTa-base (Liu et al. 2019b) and four knowledge-enhanced pre-trained model ERNIE (THU) (Zhang et al. 2019), KnowBERT (Peters et al. 2019), KEPLER (Wang et al. 2019b) and CoLAKE (Sun et al. 2020a) using their officially released models which are also pre-trained on English Wikipedia corpus and Wikidata KG. We also test two variants of our model: RoBERTa+GNN and RoBERTa+GNN+M. The two models have the same model structure as JAKET, but they are not pre-trained on our data. Moreover, the entity and relation context embedding memories of RoBERTa+GNN are randomly generated while the memories of RoBERTa+GNN+M are computed by RoBERTa.

Downstream Tasks

Few-shot Relation Classification Relation classification requires the model to predict the relation between two entities in text. Few-shot relation classification takes the N-way K-shot setting. For each query instance, N relations with K supporting examples for each relation are given. The model is required to classify the instance into one of the N relations. In this paper, we evaluate our model on a widely used benchmark dataset FewRel 1.0 (Han et al. 2018).

We use the pre-trained knowledge graph for FewRel as it comes with entity mentions from Wikidata knowledge graph. To predict the relation label, we build a sequence classification layer on top of the output of LM. More specifically, we use the PAIR framework proposed by (Gao et al. 2019), which pairs each query instance with all the supporting instances, concatenate each pair as one sequence, and send the concatenated sequence to our sequence classification model to get the score of the two instances expressing the same relation. We do not use relation embeddings in this task to avoid information leakage.

As shown in Table 1, in all three few-shot settings, our model consistently outperforms both ERNIE and KnowBERT, and performs on par with KEPLER. We didn't compare with CoLAKE since its original paper tests on this dataset in a different setting. Comparing the results between RoBERTa and RoBERTa+GNN, we see that adding GNN with randomly generated entity features does not improve the performance. The difference between RoBERTa+GNN+M and RoBERTa+GNN demonstrates the importance of generating context embedding memory by the language module, while JAKET can further improve the performance by pre-training.

Question Answering over KG The Question Answering over KG (KGQA) task is to answer natural language questions related to a knowledge graph. The answer to each question is an entity in the KG. This task requires an understanding over the question and reasoning over multiple entities and relations.

We use the vanilla version of the MetaQA (Zhang et al. 2018) dataset, which contains questions requiring multi-hop reasoning over a novel movie-domain knowledge graph. Each question is provided with one entity mention and the question is named as a k-hop question if the answer entity is a k-hop neighbor of the question entity. We define all the k-hop neighbor entities of the question entity as the candidate entities for the question. We also consider a more realistic setting where we simulate an incomplete KG by randomly dropping a triplet with a probability 50%. This setting is called *KG-50%*, compared with the full KG setting *KG-Full*. For each entity, we randomly sample one question containing it as the entity's description context. We manually write the description for each relation since the number of relations is very small. We use the output embedding of [CLS] token from LM as the question embedding, and then find the entity with the closest context embedding.

As shown in Table 2, RoBERTa+GNN+M outperforms RoBERTa, demonstrating the effectiveness of KM+LM structure. JAKET further improves the accuracy by 0.6% to 2.5% under both KG settings, showing the benefits of our proposed joint pre-training. Note that the baseline models ERNIE, KnowBERT and KEPLER even performs worse than RoBERTa. We argue that this is caused by the discrepancy between the KG used in pre-training and the downstream KG. In comparison, our model has learned the capability to quickly adapt to unseen knowledge graphs in target tasks. JAKET performs consistently better than CoLAKE, which demonstrates its stronger adaptation ability.

Entity Classification To further evaluate our model's capability to reason over unseen knowledge graphs, we design an entity classification task. Here, the model is given a portion of the Wikidata knowledge graph unseen during pre-training, denoted as $\mathcal{KG}'$. It needs to predict the category labels of these unseen entities. Our entity classification dataset contains a KG with 23,046 entities and 316 relations. The number of triplets is 38,060. Among all the entities, 16,529 of them have category labels and the total number of distinct labels is 1,291. We conduct experiments under a semi-supervised transductive setting by splitting the entities in $\mathcal{KG}'$ into train/dev/test splits of 20%, 20% and 60%. To further test the robustness of models to the size of training data, we also evaluate models when using 20% and 5% of the original training set.

In this task, RoBERTa takes the entity description text as input for label prediction while neglecting the structure information of KG. JAKET and RoBERTa+GNN+M make predictions based on the entity representation output from the GNN of the knowledge module. We also include vanilla GNN as a baseline, which uses the same GAT-based structure as

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot
BERT (Devlin et al. 2019)	85.7	89.5	76.8
ERNIE (Zhang et al. 2019)	86.9	91.4	78.6
KnowBERT (Peters et al. 2019)	86.2	90.3	77.0
KEPLER (Peters et al. 2019)	87.3	90.5	**79.4**
RoBERTa (Liu et al. 2019b)	86.4	90.3	77.3
RoBERTa+GNN	86.3	-	-
RoBERTa+GNN+M	86.9	-	-
JAKET	**87.4**	**92.1**	78.9

Table 1: Accuracy results (mean across 5 different runs) on the dev set of FewRel 1.0. All the models are equipped with the same state-of-the-art few-shot framework PAIR (Gao et al. 2019).

Model	KG-Full		KG-50%	
	1-hop	2-hop	1-hop	2-hop
ERNIE	89.8	70.1	61.2	38.7
KnowBERT	89.5	69.3	61.1	38.3
KEPLER	90.1	70.3	60.7	38.1
CoLAKE	92.4	72.1	62.5	40.6
RoBERTa	90.2	70.8	61.5	39.3
RoB+G+M	91.4	72.6	62.5	40.8
JAKET	**93.9**	**73.2**	**63.1**	**41.9**

Table 2: Hits@1 results (mean across 5 different runs) on the MetaQA dataset over 1-hop and 2-hop questions under different KG settings. RoB+G+M is the abbreviation for the baseline model RoBERTa+GNN+M.

Model	Training Size		
	100%	20%	5%
GNN	48.2	-	-
RoBERTa	33.4	-	-
RoB+G+M	79.1	66.7	53.5
JAKET	**81.6**	**70.6**	**58.4**

Table 3: Accuracy results (mean across 5 different runs) on the entity classification task over an unseen Wikidata knowledge graph. RoB+G+M is the abbreviation for the baseline model RoBERTa+GNN+M.

our knowledge module, but with randomly initialized model parameters and context embedding memory.

As shown in Table 3, our model achieves the best performance under all the settings. The performance of GNN or RoBERTa alone is significantly lower than RoBERTa+GNN+M, which demonstrates the importance of integrating both context and knowledge information using our proposed framework. Also, the gap between JAKET and RoBERTa+GNN+M increases when there's less training data, showing that the joint pre-training can reduce the model's dependence on downstream training data.

Computation Analysis

The computation of the KG module is much less than the LM module. For the RoBERTa-base model, the number of inference computation flops (#flops) over each sequence (length 128) is over 22 billion (Sun et al. 2020b). Here, we theoretically compute the number of flops of the KG module as follows: The sequence length $N = 128$, and hidden dimension $H = 768$. The number of entities in a sequence is usually less than $N/5$. The number of sampled neighbors per entity $r = 10$. And the number of layers of the GNN based KG module $L = 2$. It follows that the #flops of KG module is about $N/5 \times r^L \times 2H^2 \approx 3$ billion, less than $1/7$ of the language module computation. If we set $r = 5$, the #flops can be further reduced to about $1/30$ of LM computation.

During pre-training, another computation overhead is entity context embedding memory update (Section 3.5): Firstly, the number of entities is about 3 million and the update step interval is about 500. Thus for each step on average the model processes the description text of $3 \times 10^6/500 = 6000$ entities. Secondly, the length of description text is 64, much smaller than the length of input text 512, and we only use LM1 (the first half of LM module) for entity context embedding generation, which saves half of the computation time compared to using the whole LM module. Thirdly, the embedding update only requires forward propagation, costing only half of computation compared to training process which requires both forward and backward propagation. Thus, generating context embedding of 6k entities consumes about the same number of flops as training $6000 \times 64/(512 \times 2 \times 2) \approx 200$ input texts, much smaller than the batch size 1024. In short, the entity context embedding memory update only costs $200/1024 \approx 1/5$ additional computation. Note this computation overhead only exists during pre-training, since entity embedding memory will not be updated during fine-tuning.

Conclusion

This paper presents a novel framework, JAKET, to jointly pre-train models for knowledge graph and language understanding. Under our framework, the knowledge module and language module both provide essential information for each other. After pre-training, JAKET can quickly adapt to unseen knowledge graphs in new domains. Moreover, we design the entity context embedding memory which speeds up the pre-training by 15x. Experiments show that JAKET outperforms baseline methods in several knowledge-aware NLU tasks: few-shot relation classification, KGQA and entity classification. In the future, we plan to extend our framework to natural language generation tasks.

Acknowledgments

Donghan Yu and Yiming Yang are supported in part by the National Science Foundation (NSF) under grant IIS-1546329, and by the United States Department of Energy via the Brookhaven National Laboratory under Contract No. 384608.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baldini Soares, L.; FitzGerald, N.; Ling, J.; and Kwiatkowski, T. 2019. Matching the Blanks: Distributional Similarity for Relation Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2895–2905. Florence, Italy: Association for Computational Linguistics.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In Burges, C. J. C.; Bottou, L.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2787–2795.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ding, M.; Zhou, C.; Chen, Q.; Yang, H.; and Tang, J. 2019. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2694–2703. Florence, Italy: Association for Computational Linguistics.

Fang, Y.; Wang, S.; Xu, Y.; Xu, R.; Sun, S.; Zhu, C.; and Zeng, M. 2021. Leveraging Knowledge in Multilingual Commonsense Reasoning. *arXiv preprint arXiv:2110.08462*.

Févry, T.; Baldini Soares, L.; FitzGerald, N.; Choi, E.; and Kwiatkowski, T. 2020. Entities as Experts: Sparse Memory Access with Entity Supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4937–4951. Online: Association for Computational Linguistics.

Gao, T.; Han, X.; Zhu, H.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2019. FewRel 2.0: Towards More Challenging Few-Shot Relation Classification. 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)*, 6250–6255. Hong Kong, China: Association for Computational Linguistics.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M.-W. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 1024–1034.

Han, X.; Zhu, H.; Yu, P.; Wang, Z.; Yao, Y.; Liu, Z.; and Sun, M. 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*, 4803–4809. Brussels, Belgium: Association for Computational Linguistics.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020. Strategies for Pre-training Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Levine, Y.; Lenz, B.; Dagan, O.; Ram, O.; Padnos, D.; Sharir, O.; Shalev-Shwartz, S.; Shashua, A.; and Shoham, Y. 2020. SenseBERT: Driving Some Sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4656–4667. Online: Association for Computational Linguistics.

Liu, N. F.; Gardner, M.; Belinkov, Y.; Peters, M. E.; and Smith, N. A. 2019a. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; and Wang, P. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2901–2908.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lv, S.; Guo, D.; Xu, J.; Tang, D.; Duan, N.; Gong, M.; Shou, L.; Jiang, D.; Cao, G.; and Hu, S. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8449–8456.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized

Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Peters, M. E.; Neumann, M.; Logan, R.; Schwartz, R.; Joshi, V.; Singh, S.; and Smith, N. A. 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)*, 43–54. Hong Kong, China: Association for Computational Linguistics.

Poerner, N.; Waltinger, U.; and Schütze, H. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. *https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8): 9.

Ringgaard, M.; Gupta, R.; and Pereira, F. C. 2017. SLING: A framework for frame semantic parsing. *arXiv preprint arXiv:1710.07032*.

Shen, T.; Mao, Y.; He, P.; Long, G.; Trischler, A.; and Chen, W. 2020. Exploiting Structured Knowledge in Text via Graph-Guided Representation Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8980–8994. Online: Association for Computational Linguistics.

Sun, T.; Shao, Y.; Qiu, X.; Guo, Q.; Hu, Y.; Huang, X.; and Zhang, Z. 2020a. Colake: Contextualized language and knowledge embedding. *arXiv preprint arXiv:2010.00309*.

Sun, Y.; Wang, S.; Li, Y.; Feng, S.; Chen, X.; Zhang, H.; Tian, X.; Zhu, D.; Tian, H.; and Wu, H. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020b. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Talmor, A.; Elazar, Y.; Goldberg, Y.; and Berant, J. 2019. oLMpics–On what Language Model Pre-training Captures. *arXiv preprint arXiv:1912.13283*.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. P. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Verga, P.; Sun, H.; Soares, L. B.; and Cohen, W. W. 2020. Facts as Experts: Adaptable and Interpretable Neural Memory over Symbolic Knowledge. *arXiv preprint arXiv:2007.00849*.

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

Wang, H.; Liu, Y.; Zhu, C.; Shou, L.; Gong, M.; Xu, Y.; and Zeng, M. 2021. Retrieval Enhanced Model for Commonsense Generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 3056–3062. Online: Association for Computational Linguistics.

Wang, M.; Yu, L.; Zheng, D.; Gan, Q.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; et al. 2019a. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315*.

Wang, X.; Gao, T.; Zhu, Z.; Liu, Z.; Li, J.; and Tang, J. 2019b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *arXiv preprint arXiv:1911.06136*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771.

Xiong, W.; Du, J.; Wang, W. Y.; and Stoyanov, V. 2020. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xu, Y.; Zhu, C.; Xu, R.; Liu, Y.; Zeng, M.; and Huang, X. 2021. Fusing Context Into Knowledge Graph for Commonsense Question Answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1201–1207. Online: Association for Computational Linguistics.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yu, W.; Zhu, C.; Fang, Y.; Yu, D.; Wang, S.; Xu, Y.; Zeng, M.; and Jiang, M. 2021. Dict-BERT: Enhancing Language Model Pre-training with Dictionary. *arXiv preprint arXiv:2110.06490*.

Yu, W.; Zhu, C.; Li, Z.; Hu, Z.; Wang, Q.; Ji, H.; and Jiang, M. 2020. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

Zhang, Y.; Dai, H.; Kozareva, Z.; Smola, A. J.; and Song, L. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1441–1451. Florence, Italy: Association for Computational Linguistics.