CoLAKE: Contextualized Language and Knowledge Embedding

Tianxiang Sun[1,*], Yunfan Shao[1], Xipeng Qiu[1,†]
Qipeng Guo[1], Yaru Hu[1], Xuanjing Huang[1], Zheng Zhang[2]
[1]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[1]School of Computer Science, Fudan University
[2]Amazon Shanghai AI Lab
{txsun19,yfshao19,xpqiu,qpguo16,xjhuang}@fudan.edu.cn
yrhu112358@outlook.com zhaz@amazon.com

Abstract

With the emerging branch of incorporating factual knowledge into pre-trained language models such as BERT, most existing models consider shallow, static, and separately pre-trained entity embeddings, which limits the performance gains of these models. Few works explore the potential of deep contextualized knowledge representation when injecting knowledge. In this paper, we propose the Contextualized Language and Knowledge Embedding (CoLAKE), which jointly learns contextualized representation for both language and knowledge with the extended MLM objective. Instead of injecting only entity embeddings, CoLAKE extracts the knowledge context of an entity from large-scale knowledge bases. To handle the heterogeneity of knowledge context and language context, we integrate them in a unified data structure, word-knowledge graph (WK graph). CoLAKE is pre-trained on large-scale WK graphs with the modified Transformer encoder. We conduct experiments on knowledge-driven tasks, knowledge probing tasks, and language understanding tasks. Experimental results show that CoLAKE outperforms previous counterparts on most of the tasks. Besides, CoLAKE achieves surprisingly high performance on our synthetic task called word-knowledge graph completion, which shows the superiority of simultaneously contextualizing language and knowledge representation.[1]

1 Introduction

Deep contextualized language models pre-trained on large-scale unlabeled corpora have achieved significant improvement on a wide range of NLP tasks (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019). However, they are shown to have difficulty capturing factual knowledge (Logan et al., 2019).

Recently, there is a growing interest in combining pre-trained language models (PLMs) with structured knowledge. A popular approach is to inject pre-trained entity embeddings into PLMs to better capture factual knowledge, such as ERNIE (Zhang et al., 2019) and KnowBERT (Peters et al., 2019). The shortcomings of these models can be summarized as follows: (1) The entity embeddings are separately pre-trained with some knowledge embedding (KE) models (e.g., TransE (Bordes et al., 2013)), and fixed during training PLMs. Thus they are not real joint models to learn the knowledge embedding and language embedding simultaneously. (2) The previous models only take entity embeddings to enhance PLMs, which are hard to fully capture the rich contextual information of an entity in the knowledge graph (KG). Thus their performance gains are limited by the quality of pre-trained entity embeddings. (3) The pre-trained entity embeddings are static and need to be re-trained when the KG is slightly changed.

In this paper, we propose the **Co**ntextualized **L**anguage **A**nd **K**nowledge **E**mbedding (**CoLAKE**), which jointly learns language representation and knowledge representation in a common representation space. Different from the previous models, CoLAKE dynamically represents an entity according to its *knowledge context* and *language context*. For each entity, CoLAKE considers a sub-graph surrounding

[*]Work done during internship at Amazon Shanghai AI Lab.
[†]Corresponding author.
[1]Our code is available at https://github.com/txsun1997/CoLAKE.

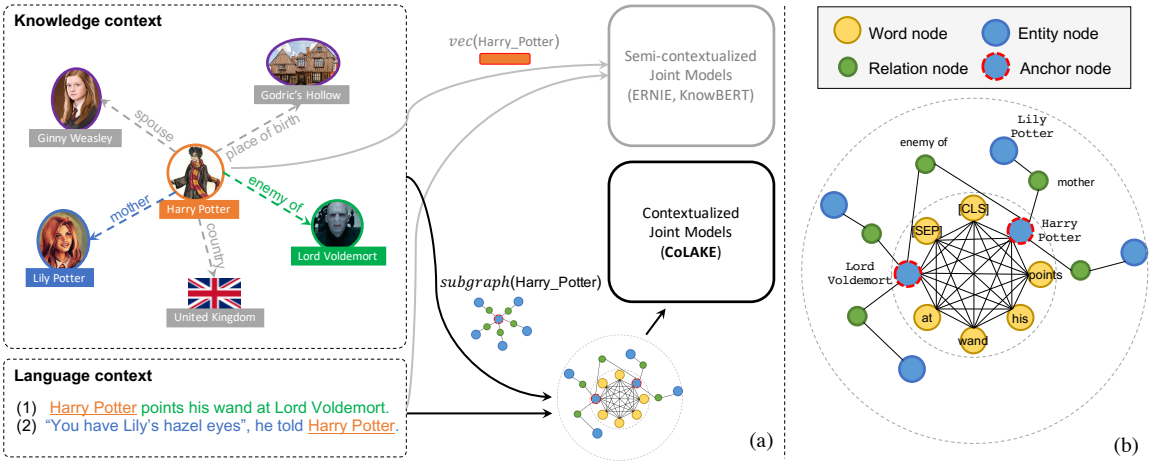


Figure 1: (a) When injecting knowledge, CoLAKE considers the *knowledge context* about the entity while previous semi-contextualized joint models only consider a single entity embedding. By contextualizing the entity, CoLAKE is able to directly access (`Harry_Potter`, *enemy of*, `Lord_Voldemort`) to help understand sentence (1) and access (`Harry_Potter`, *mother*, `Lily_Poter`) to help understand sentence (2). (b) The word-knowledge graph (WK graph) is a unified structure to represent both language context and knowledge context, which is composed of two parts: the fully-connected word graph and knowledge sub-graphs extracted from large KGs.

it as its knowledge context that contains the facts (triplets) about the entity. In this way, CoLAKE can dynamically access different facts as background knowledge to help understand the current text. As shown in Figure 1(a), to understand different sentences, CoLAKE can utilize different facts about the linked entity `Harry_Potter`. The *knowledge context* of `Harry_Potter` is a sub-graph containing the triplets about it. According to whether or not to utilize entities' knowledge context, our proposed CoLAKE can be distinguished from previous models, which we call *semi-contextualized joint models* since they only contextualize language representation.

To deal with the heterogeneous structure of language and KG, we build a graph to integrate them into a unified data structure, called *word-knowledge graph (WK graph)*. Most recent successful PLMs use Transformer architecture (Vaswani et al., 2017), which treats input sequences as fully-connected word graphs. WK graph is knowledge-augmented word graph. Using entities mentioned in the sentence, we extract sub-graphs centered on those mentioned entities from KGs. Then we mosaic such sub-graphs and the word graph in a unified heterogeneous graph, i.e. WK graph. An instance of the WK graph can be found in Figure 1(b). The constructed WK graph is fed into CoLAKE along with its adjacency matrix to control the information flow to reflect the graph structure. CoLAKE is based on the Transformer encoder, with the embedding layer and the encoder layers slightly modified to adapt to input in the form of WK graph. Besides, we extend the masked language model (MLM) objective (Devlin et al., 2019) to the whole input graph. That is, apply the same masking strategy to word, entity, and relation nodes and training the model to predict the masked nodes based on the rest of the graph.

We evaluate CoLAKE on several knowledge-required tasks and GLUE (Wang et al., 2019a). Experimental results demonstrate that CoLAKE outperforms previous semi-contextualized counterparts on most of the tasks. To explore potential applications of CoLAKE, we design a synthetic task called word-knowledge graph completion. Our evaluation on this task shows that CoLAKE outperforms several KE models by a large margin, in transductive setting and inductive setting.

In summary, CoLAKE can be characterized in three-fold: (1) CoLAKE learns contextualized language representation and contextualized knowledge representation simultaneously with the extended MLM objective. (2) CoLAKE adopts the WK graph to integrate the heterogeneous input for language and knowledge. (3) CoLAKE is essentially a pre-trained graph neural network (GNN), thereby being structure-aware and easy to extend.

2 Related Work

Language Representation Learning. The past decade has witnessed the great success of pre-trained language representation. Initially, word representation pre-trained using multi-task objectives (Collobert and Weston, 2008) or co-occurrence statistics (Mikolov et al., 2013; Pennington et al., 2014) are static and non-contextual. Recently, contextualized word representation pre-trained on large-scale unlabeled corpora with deep neural networks has dominated across a wide range of NLP tasks (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019; Qiu et al., 2020).

Knowledge Representation Learning. Knowledge Representation Learning (KRL) is also termed as Knowledge Embedding (KE), which is to map entities and relations into low-dimensional continuous vectors. Most existing methods use triplets as training samples to learn static, non-contextual embeddings for entities and relations (Bordes et al., 2013; Yang et al., 2015; Lin et al., 2015). Recent advances focusing on contextualized representation, which use subgraphs or paths as training samples, have achieved new state-of-the-art results on KG tasks (Wang et al., 2019b; Wang et al., 2020a).

Joint Language and Knowledge Models. Due to the mutual information existing in language and KGs, joint models often benefit both sides. Besides, tasks such as entity linking also require entity embeddings that are compatible with word embeddings. Combining the success of Mikolov et al. (2013) and Bordes et al. (2013), Wang et al. (2014) jointly learn embeddings for language and KG. Targeting mention-entity matching in entity linking, Yamada et al. (2016), Ganea and Hofmann (2017) also proposed joint methods to map entities and words into the same vector space. Inspired by the recent success of contextualized language representation, much effort has been devoted to injecting entity embeddings into PLMs (Zhang et al., 2019; Peters et al., 2019). Despite their success, the knowledge gains are limited by the expressivity of their used pre-trained entity embeddings, which is static and inflexible. In contrast, KEPLER (Wang et al., 2019c) aims to benefit both sides so jointly learn language model and knowledge embedding. However, KEPLER does not directly learn embeddings for each entity but learns to generate entity embeddings with PLMs from entity descriptions. Besides, none of these work exploits the potential of contextualized knowledge representation, which makes them different from our proposed CoLAKE. A brief comparison can be found in Table 1. CoLAKE is conceptually similar to K-BERT (Liu et al., 2020) and BERT-MK (He et al., 2019). CoLAKE differs from K-BERT in that, instead of injecting triplets during fine-tuning, CoLAKE jointly learns embeddings for entities and relations during pre-training LMs. Besides, CoLAKE places language and knowledge representation learning into a unified pre-training task, masked language model, which makes it more concise than BERT-MK. In addition, CoLAKE is a general-purpose joint model while BERT-MK mainly focuses on medical domain.

	Joint Models	Language		Knowledge	
		Objective	Contextualized?	Objective	Contextualized?
Non-contextual	Wang et al. (2014)	Skip-Gram	✗	TransE	✗
	Yamada et al. (2016)	Skip-Gram	✗	Skip-Gram	✗
Semi-contextualized	ERNIE (Zhang et al., 2019)	MLM	✓	TransE*	✗
	KnowBERT (Peters et al., 2019)	MLM	✓	-	✗
	KEPLER (Wang et al., 2019c)	MLM	✓	TransE	✗
Contextualized	CoLAKE (Ours)	MLM	✓	MLM	✓

Table 1: Comparison of several joint models based on whether the representation is contextualized. *The entity embeddings are fixed during pre-training ERNIE. KnowBERT does not have restrictions on the entity embeddings. For ERNIE and KnowBERT, we omit the next sentence prediction (NSP) objective.

3 CoLAKE

CoLAKE jointly learns contextualized representation for language and knowledge by pre-training on structured, unlabeled *word-knowledge graphs (WK graphs)*. We first introduce how to construct such WK graphs, then we describe the model architecture and the implementation details.

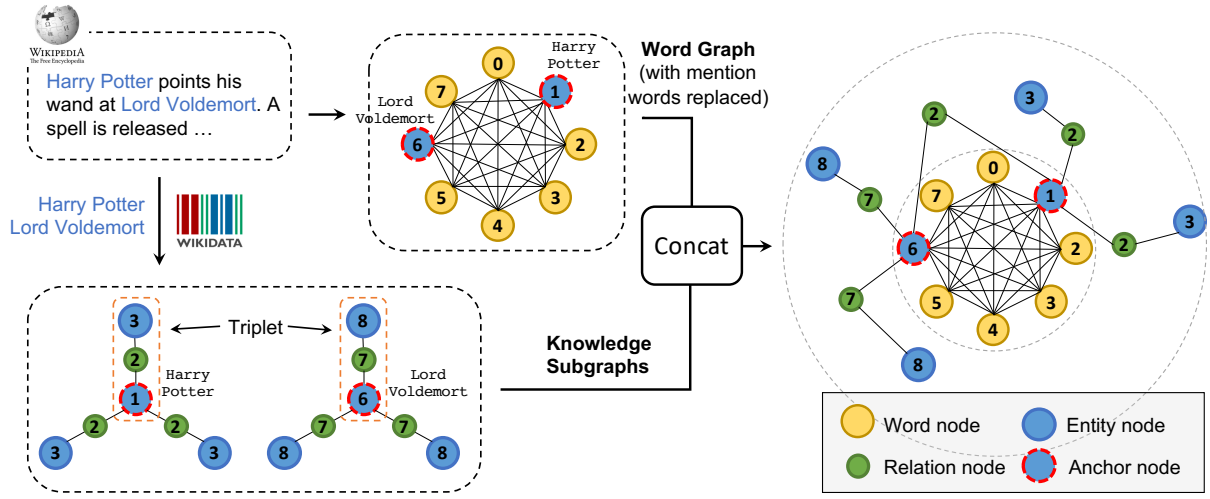


Figure 2: Illustration of WK graph construction. The WK graph is an undirected heterogeneous graph. The numbers marked on graph nodes indicate the position index introduced in Section 3.2.

3.1 Graph Construction

Typically, language embedding models take sequences as input while knowledge embedding (KE) models take triplets or knowledge sub-graphs as input. Recent successful PLMs take Transformer (Vaswani et al., 2017) as their backbone architecture, which actually processes sequences as fully-connected word graphs. Thus, graph is a common data structure to represent language and knowledge. In this section, we show how to integrate word graph and knowledge sub-graphs into the unified WK graph.

We first tokenize a sentence into a sequence of tokens and fully connect them as a word graph. Then we recognize the mentions in the sentence and use an entity linker to find the corresponding entities in a certain KG. The mention nodes are then replaced by their linked entities, which are called *anchor nodes*. By this replacement, the model is encouraged to map the injected entities and mention words near one another in the vector space. Centered on the anchor nodes $\{e_i\}_i$, we can extract their knowledge contexts $\{\{e_i, r_{ij}, e_{ij}\}_j\}_i$ to form sub-graphs, in which relations are also transformed into graph nodes. The extracted sub-graphs and the word graph are then concatenated with anchor nodes to obtain the WK graph. Figure 2 shows the process of constructing a WK graph. In practice, for each anchor node we randomly select up to 15 neighboring relations and entities to construct a sub-graph to be injected into the WK graph. We only consider triplets in which anchor node is head (subject) instead of tail (object). In the WK graph, entities are unique but relations are allowed to repeat.

3.2 Model Architecture

The constructed WK graphs are then fed into the Transformer (Vaswani et al., 2017) encoder. We modify the embedding and encoder layers of vanilla Transformer to adapt to input in the form of WK graph.

Embedding Layer. The input embedding is the sum of token embedding, type embedding, and position embedding. For token embedding, we maintain three lookup tables for words, entities, and relations respectively. For word embedding, we follow RoBERTa (Liu et al., 2019) which uses Byte-Pair Encoding (BPE) (Sennrich et al., 2016) to transform sequence into subwords units to handle the large vocabulary. In contrast, we directly learn embeddings for each unique entity and relation as common knowledge embedding methods do. The token embeddings are obtained by concatenating word, entity, and relation embeddings, which are of the same dimensionality. There are different types of nodes so the WK graph is heterogeneous. To handle this, we simply use type embedding to indicate the node types, i.e. word, entity, and relation. For position embedding, we need to assign each injected entity and relation a position index. Inspired by Liu et al. (2020), we adopt the soft-position index which allows repeated position indices and keeps tokens in the same triplet continuous. Figure 2 shows an intuitive example of how to assign position index to graph nodes.

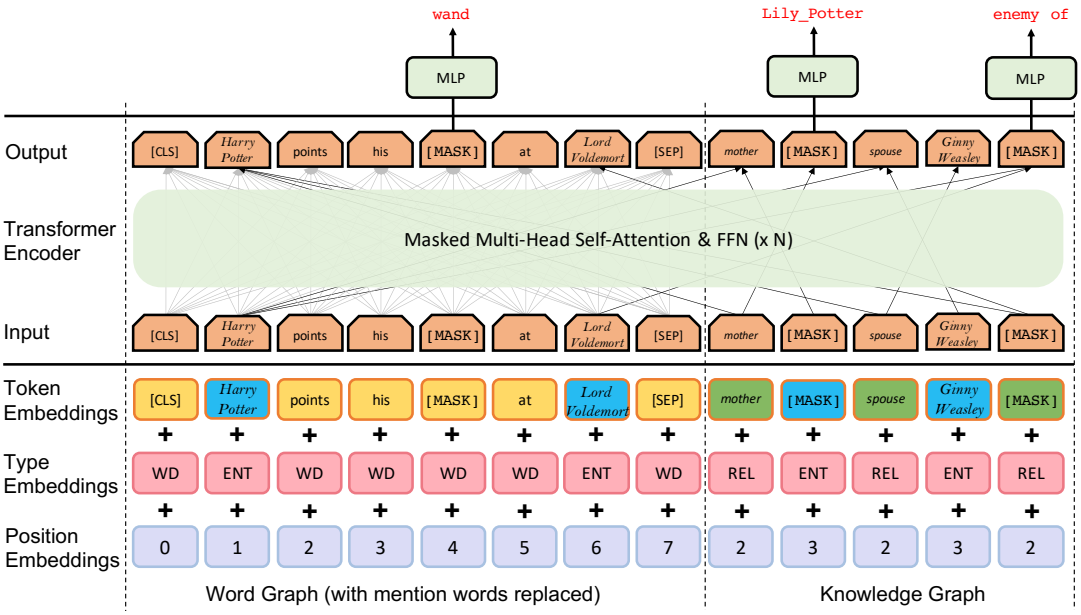


Figure 3: Overall architecture of CoLAKE. In this case, three triplets, (`Harry_Potter`, *mother*, `Lily_Potter`), (`Harry_Potter`, *spouse*, `Ginny_Weasley`), and (`Harry_Potter`, *enemy of*, `Lord_Voldemort`) are injected into the raw sequence. The model is asked to predict the masked word *wand*, the masked entity `Lily_Potter`, and the masked relation *enemy of*.

Masked Transformer Encoder. We use masked multi-head self-attention to control the information flow to reflect the structure of WK graph. Given the representation of graph nodes $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the dimension for each node, the representation after masked self-attention is obtained by

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{X}\mathbf{W}^Q, \mathbf{X}\mathbf{W}^K, \mathbf{X}\mathbf{W}^V, \tag{1}$$

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}, \tag{2}$$

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\mathbf{A} + \mathbf{M})\mathbf{V}, \tag{3}$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_k}$ are learnable parameters. $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mask matrix given by

$$\mathbf{M}_{ij} = \begin{cases} 0 & \text{if } x_i \text{ and } x_j \text{ are connected,} \\ -\inf & \text{if } x_i \text{ and } x_j \text{ are disconnected.} \end{cases} \tag{4}$$

With the masked Transformer encoder, each node can only gather information from its 1-hop neighbor at each layer. Masked Transformer encoder works similar to GAT (Velickovic et al., 2018).

3.3 Pre-Training Objective

The Masked Language Model (MLM) objective is to randomly mask some of tokens from the input and train the model to predict the original vocabulary id of the masked tokens based on their contexts. In this section, we extend the MLM from word sequences to WK graphs.

In particular, we mask 15% of graph nodes at random. When a node is masked, we replace it with (1) the `[MASK]` token 80% of time, (2) a randomly sampled node with the same type as the original node 10% of time, (3) the unchanged node 10% of time. As different types of nodes are masked, we encourage CoLAKE to learn different aspects of capabilities:

- **Masking word nodes.** When words are masked, the objective is similar to traditional MLM. The difference is, CoLAKE can predict masked words based on not only the context words but also the entities and relations in the WK graph. Masking words helps CoLAKE learn linguistic knowledge.

- **Masking entity nodes.** If the masked entity is an anchor node, the objective, which is to predict the anchor node based on its context, helps to align the representation spaces of language and knowledge. Take the instance in Figure 1(b), the embedding of entity `Harry_Potter` will be learned to be similar to its textual form, *Harry Potter*. If the masked entity is not an anchor node, the MLM objective is similar to that used in semantic matching-based KE methods such as ConvE (Dettmers et al., 2018) and CoKE (Wang et al., 2019b), which enables CoLAKE to learn a large number of entity embeddings. Masking entity nodes helps CoLAKE (a) map words and entities into a common representation space, and (b) learn contextualized representation for entities.

- **Masking relation nodes.** If the masked relation is between two unique anchor nodes, the objective is similar to distantly supervised relation extraction (Craven and Kumlien, 1999), which requires the model to classify the relationship between two entities mentioned in the text. Otherwise, the objective is to predict the relationship between its two neighboring entities, which is similar to traditional KE methods. Masking relation nodes helps CoLAKE (a) learn to do relation extraction, and (b) learn contextualized representation for relations.

However, the pre-training task of predicting masked anchor nodes could be trivial because the model is easy to accomplish this task only based on the knowledge context instead of the language context, which is more varied than knowledge context. To mitigate this, we discard neighbors of anchor nodes in 50% of time during pre-training.

3.4 Model Training

CoLAKE is trained with cross-entropy loss. We use three classification heads to predict three types of nodes. In practice, however, the large number of entities brings challenges to training and predicting.

Mixed CPU-GPU Training. Due to the large number of entities in KG, training the whole model in GPU is intractable. To handle this, we asynchronously update entity embeddings in CPU memory while keeping the rest of our model updated in GPU. In particular, we store and update entity embeddings in CPU memory which is shared among multiple trainer processes. During pre-training, the trainer processes read the entity embeddings from the shared CPU memory and write the gradients back to CPU. Our implementation is based on the distributed key-value store (KVStore) from Zheng et al. (2020).

Negative Sampling. Applying the Softmax function to the huge number of entities is very time-consuming. CoLAKE uses negative sampling to conduct prediction for each entity over one positive entity and $k(k \ll n)$ negative entities instead of all n entities in KG. Following Mikolov et al. (2013), we sample negative entities from the 3/4 powered entity frequency distribution.

4 Experiments

In this section, we present the details of pre-training and fine-tuning CoLAKE, and its experimental results on knowledge-driven tasks, knowledge probing tasks, and language understanding tasks.

4.1 Pre-Training Data and Implementation Details

CoLAKE uses English Wikipedia (2020/03/01)[2] as pre-training data and uses WikiExtractor[3] to process the downloaded Wikipedia dump. We use Wikipedia anchors to align text to Wikidata5M (Wang et al., 2019c), which is a newly proposed large-scale KG containing 21M fact triplets. We construct WK graphs as training samples and filter out graph samples without entity nodes and relation nodes. Finally, CoLAKE pre-trained the Transformer encoder along with 3,085,345 entity embeddings and 822 relation embeddings on 26M training samples.

The Transformer encoder of CoLAKE is initialized with RoBERTa$_{\text{BASE}}$ (Liu et al., 2019). We use the implementation from HuggingFace's Transformer (Wolf et al., 2019). The entity embeddings and relation embeddings are initialized with the average of the RoBERTa$_{\text{BASE}}$ BPE embeddings of entity

[2] `https://dumps.wikimedia.org/enwiki/`

[3] `https://github.com/attardi/wikiextractor`

and relation aliases provided by Wang et al. (2019c). AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.98$ is used in pre-training. We train CoLAKE with the batch size of 2048 and the learning rate of 1e-4 for 1 epoch. For each anchor node, we sample $k = 200$ negative entities. CoLAKE is trained on 8 32G NVIDIA V100 GPUs for 38 hours.

4.2 Knowledge-Driven Tasks

We first fine-tune and evaluate CoLAKE on knowledge-driven tasks. To annotate entities in the sentence, we use TAGME (Ferragina and Scaiella, 2010) to link mentions to entities in KGs. Instead of replacing the textual mention with its symbolic entity, we follow Pörner et al. (2019) and concatenate the two forms of tokens, e.g. Jean Mara ##is `Jean_Marais`. Concretely, we conduct experiments on two knowledge-driven tasks: entity typing and relation extraction.

Entity Typing. The entity typing task is to classify the semantic type of a given entity mention based on its surface form and context. We add two special tokens, `[ENT]` and `[/ENT]`, before and after the entity mentions to be classified and use the final representation of the `[CLS]` token as the feature to conduct classification[4]. We evaluate CoLAKE on Open Entity (Choi et al., 2018). To compare with ERNIE, KnowBERT, and KEPLER, we adopt the same experiment setting which considers nine general types. To be consistent with previous work, we adopt micro precision, recall, and F1 score as evaluation metrics. The experimental results are shown in Table 2.

Relation Extraction. The relation extraction task is to classify the relationship between two entities mentioned in a given sentence. During fine-tuning, we add four special tokens, `[HD]`, `[/HD]`, `[TL]` and `[/TL]` to identify the head entity and the tail entity. Also, we use the final representation of the `[CLS]` token as the feature to be fed into the classifier. We evaluate CoLAKE on FewRel (Han et al., 2018) that is rearranged by Zhang et al. (2019). Since FewRel is built with Wikidata, we discard triplets in the FewRel test set from pre-training data to avoid information leakage. Following previous work, we report macro precision, recall and F1 score on FewRel. The experimental results can be found in Table 2.

Model	Open Entity			FewRel		
	P	R	F	P	R	F
BERT (Devlin et al., 2019)	76.4	71.0	73.6	85.0	85.1	84.9
RoBERTa (Liu et al., 2019)	77.4	73.6	75.4	85.4	85.4	85.3
ERNIE (Zhang et al., 2019)	78.4	72.9	75.6	88.5	88.4	88.3
KnowBERT (Peters et al., 2019)	**78.6**	73.7	76.1	-	-	-
KEPLER (Wang et al., 2019c)	77.8	74.6	76.2	-	-	-
E-BERT (Pörner et al., 2019)	-	-	-	88.6	88.5	88.5
CoLAKE (Ours)	77.0	**75.7**	**76.4**	**90.6**	**90.6**	**90.5**

Table 2: Experimental results on Open Entity and FewRel.

4.3 Knowledge Probing

LAMA (LAnguage Model Analysis) probe (Petroni et al., 2019) aims to measure factual knowledge stored in language models via cloze-style statement like: *Dante was born in [MASK]*. Subsequently, a more "factual" subset of LAMA, LAMA-UHN (Pörner et al., 2019), is constructed by filtering out easy-to-answer samples. We evaluate CoLAKE on these two probes and report the mean precision at one (P@1) macro-averaged over relations.

For fair comparision, we use the intersection of the vocabularies for all considered models and construct a common vocabulary of ∼18K case-sensitive tokens. In this experiment, considered models include ELMo (Peters et al., 2018), ELMo5.5B (Peters et al., 2018), $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019), $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019), and K-Adapter (Wang et al., 2020b).

The results of LAMA and LAMA-UHN are shown in Table 3. It is worth noticing that BERT outperforms RoBERTa by a large margin. Wang et al. (2020b) reported the same phenomenon in their

[4]In the implementation of RoBERTa, the `[CLS]` token is replaced with `<s>`.

Corpus	Pre-trained Models					
	ELMo	ELMo5.5B	BERT	RoBERTa	CoLAKE	K-Adapter*
LAMA-Google-RE	2.2	3.1	11.4	5.3	9.5	7.0
LAMA-UHN-Google-RE	2.3	2.7	5.7	2.2	4.9	3.7
LAMA-T-REx	0.2	0.3	32.5	24.7	28.8	29.1
LAMA-UHN-T-REx	0.2	0.2	23.3	17.0	20.4	23.0

Table 3: P@1 on LAMA and LAMA-UHN. *K-Adapter is based on RoBERTa$_{\mathrm{LARGE}}$ while other Transformer-based LMs are of BASE size. Besides, K-Adapter uses a subset of T-REx as its training data, which may contribute to its superiority over CoLAKE on LAMA-T-REx and LAMA-UHN-T-REx.

paper. We conjecture that the main reason behind this is the larger and byte-level BPE vocabulary used by RoBERTa. Though, CoLAKE outperforms its baseline, RoBERTa$_{\mathrm{BASE}}$, by a significant margin. Besides, CoLAKE even improves over the LARGE size of model, K-Adapter, by 2.5% and 1.2% on LAMA-Google-RE and LAMA-UHN-Google-RE respectively.

4.4 Language Understanding Tasks

We also evaluate CoLAKE on the General Language Understanding Evaluation (GLUE) (Wang et al., 2019a), which provides a collection of diverse NLU tasks. Since these tasks require little factual knowledge, we attempt to explore whether CoLAKE degenerates the performance on these NLU tasks.

The experimental results on GLUE dev set are shown in Table 4. CoLAKE is slightly degraded from RoBERTa but improves over KEPLER by 1.4% on average. We conclude that CoLAKE is able to simultaneously model text and knowledge via the heterogeneous WK graph. In summary, our experiments demonstrate that CoLAKE significantly improves the performance on knowledge-required tasks and at the same time achieves comparable results on language understanding tasks.

Model	MNLI (m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	AVG.
RoBERTa	87.5 / 87.3	91.9	92.8	94.8	63.6	91.2	90.2	78.7	86.4
KEPLER	87.2 / 86.5	91.5	92.4	94.4	62.3	89.4	89.3	70.8	84.9
CoLAKE	87.4 / 87.2	92.0	92.4	94.6	63.4	90.8	90.9	77.9	86.3

Table 4: GLUE results on dev set. Both of KEPLER and CoLAKE are initialized with RoBERTa$_{\mathrm{BASE}}$.

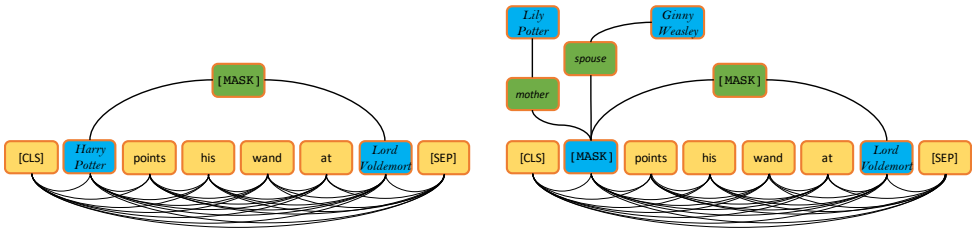
4.5 Word-Knowledge Graph Completion

Note that CoLAKE is essentially a GNN pre-trained on large-scale WK graphs, which makes it structure-aware and easy to generalize to unseen entities.

To probe CoLAKE's capability of modeling both structural and semantic features, we design a task named *word-knowledge graph completion*. In particular, we use the FewRel test set to construct two experimental settings: transductive setting and inductive setting. In both settings, each sample provides a triplet (h, r, t) and a sentence that expresses the triplet. The considered models are required to predict the relation r. For each sample in **transductive setting**, the two entities, h and t, and their relation r are seen in training phase. But the triplet (h, r, t) has not appeared in the training data. We collect 10K samples from the FewRel test set to construct the transductive setting. For each sample in **inductive setting**, at least one entity is unseen during training. This setting requires the model to be inductive so that it can generalize to unseen entities. We collect 1K samples from the FewRel test set to construct the inductive setting. We directly evaluate CoLAKE in the two settings without further training on the FewRel training set. The forms of word-knowledge graph are depicted in Figure 4. For inductive setting, we encourage CoLAKE to infer the unseen entity by aggregating messages from its neighbors.

We take several well-known models for link prediction as our baselines[5]. For transductive setting, we compare CoLAKE with four widely-used models, i.e. TransE (Bordes et al., 2013), DistMult (Yang et

[5]We did not compare with KEPLER (Wang et al., 2019c) since the authors did not release their data split and model yet.



(a) Transductive setting. (b) Inductive setting.

Figure 4: Illustration of the input word-knowledge graph for transductive setting and inductive setting. In transductive setting, `Harry_Potter` and `Lord_Voldemort` are seen during training. In inductive setting, `Harry_Potter` is unknown but its neighboring entities are seen in training data.

al., 2015), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019). We use DGL-KE (Zheng et al., 2020) to train the four baseline models on Wikidata5M[6]. For inductive setting, we take DKRL (Xie et al., 2016) as our baseline. As shown in Table 5, CoLAKE outperforms other models by a large margin thanks to its capability of simultaneously utilizing structural knowledge and rich text semantics while traditional KE models can only handle structural knowledge. Besides, the inductive ability of CoLAKE is more realistic. Unlike DKRL and KEPLER, which generate entity embeddings from descriptions, CoLAKE generates entity embeddings based on their neighbors.

Model	MR ↓	MRR	HITS@1	HITS@3	HITS@10
Transductive setting					
TransE (Bordes et al., 2013)	15.97	67.30	60.28	70.96	79.75
DistMult (Yang et al., 2015)	27.09	60.56	48.66	69.69	79.61
ComplEx (Trouillon et al., 2016)	26.73	61.09	49.80	70.64	79.78
RotatE (Sun et al., 2019)	30.36	70.90	64.74	74.89	81.05
CoLAKE	2.03	82.48	72.14	92.19	98.58
Inductive setting					
DKRL (Xie et al., 2016)	168.21	8.18	5.03	7.28	14.13
CoLAKE	31.01	28.10	15.69	30.28	58.05

Table 5: The experimental results on word-knowledge graph completion task.

5 Conclusion

In this paper, we propose CoLAKE to jointly learn contextualized representation for language and knowledge. We integrate the language context and knowledge context in a unified data structure, word-knowledge graph. The experimental results show the effectiveness of CoLAKE on knowledge-required tasks. Besides, to explore the potential application of the WK graph, we design a task named WK graph completion, which shows that CoLAKE is essentially a powerful GNN that is structure-aware and inductive. The surprisingly high performance on WK graph completion inspires the potential applications of WK graph, for example, (a) CoLAKE may help to denoise distantly annotated samples of relation extraction (Craven and Kumlien, 1999; Mintz et al., 2009), (b) CoLAKE can be used to measure the quality of graph-to-text templates (Davison et al., 2019; Bouraoui et al., 2020) due to its capability of preserving the original graph structure. We leave these applications as future work.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2020AAA0106700) and National Natural Science Foundation of China (No. 62022027 and 61976056).

[6]The triplets in FewRel test set are removed from Wikidata5M to avoid information leakage.

References

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *AAAI*, pages 7456–7463.

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

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

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, pages 77–86.

Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. In *EMNLP-IJCNLP*, pages 1173–1178.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186.

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

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*, pages 2619–2629.

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 *EMNLP*, pages 4803–4809.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2019. Integrating graph contextualized knowledge into pre-trained language models. *CoRR*, abs/1912.00147.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

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

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph.

Robert L. Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, pages 5962–5971.

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

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011.

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

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

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, pages 43–54.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473.

Nina Pörner, Ulli Waltinger, and Hinrich Schütze. 2019. BERT is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised QA. *CoRR*, abs/1911.03681.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019b. Coke: Contextualized knowledge graph embedding. *CoRR*, abs/1911.02168.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2019c. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *CoRR*, abs/1911.06136.

Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2020a. Entity context and relational paths for knowledge graph completion. *CoRR*, abs/2002.06757.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020b. K-adapter: Infusing knowledge into pre-trained models with adapters. *CoRR*, abs/2002.01808.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*, pages 250–259.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. 2020. DGL-KE: training knowledge graph embeddings at scale. *CoRR*, abs/2004.08532.