K-BERT: Enabling Language Representation with Knowledge Graph

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng and Ping Wang

Abstract

Pre-trained language representation models, such as BERT, capture a general language representation from large-scale corpora, but lack domain-specific knowledge. When reading a domain text, experts make inferences with relevant knowledge.For machines to achieve this capability, we propose a knowledge-enabled language representation model (K-BERT) with knowledge graphs (KGs). K-BERT can easily inject triples as domain knowledge into the model, as it can load model parameters from pre-trained BERTs without the need to pre-train itself. Experiments reveal promising results in twelve NLP tasks. Especially in domain-specific tasks (including finance, law, and medicine), K-BERT significantly outperforms BERT, which demonstrates that K-BERT is an excellent choice for solving the knowledge-driven problems that require experts.

Keywords：Pre-trained models, knowledge graphs, knowledge-driven, language representations

1 Introduction

Pre-trained Language Representation (LR) models have achieved promising results in multiple NLP tasks. These models are pre-trained over large-scale open-domain corpora to obtain general language representations and then fine-tuned in specific downstream tasks for absorbing specific-domain knowledge. However, due to the domain-discrepancy between pre-training and fine-tuning, these models do not perform well on knowledge-driven tasks.

The paper chosen for this replication work designs a knowledge-enabled language representation model (K-BERT) with knowledge graphs (KGs). In this model, triples are injected into sentences as domain knowledge. However, too much knowledge incorporation may cause the sentence to deviate from its correct meaning, which is referred to as the knowledge noise (KN) problem. To overcome KN, K-BERT introduces softposition and visible matrix to limit the impact of knowledge. k-BERT can be loaded with any pre-trained BERT model and domain knowledge can easily be injected into the model by configuring the knowledge graph so that it has the knowledge of a domain expert, using which domain-specific language representation features can be better captured.

2 Related works

2.1 Traditional methods of optimising the pre-training process

In optimizing pre-training process, SpanBERT extended BERT by masking contiguous random spans and proposed a span boundary objective[1]. RoBERTa optimized the pre-training of BERT in three ways, i.e.,

deleting the target of the next sentence prediction, dynamically changing the masking strategy and using more and longer sentences for training[2]. THU-ERNIE modified the encoder of BERT to an aggregator for the mutual integration of word and entities[3].

2.2 Combination of knowledge graphs and word vectors

Before the emergence of pre-trained LR models, there were several studies that combined KG with word vectors. Toutanova et al. (2015) proposed a model that captures the compositional structure of textual relations, and optimize entity, knowledge base, and textual relation representations in a joint manner[4]. Han, Liu, and Sun (2016) applied a convolutional neural network and a KG completion task to learn the representation of text and knowledge jointly[5]. Cao et al. (2018) carried out cross-lingual representation learning for words and entities via attentive distant supervision[6].

3 Method

3.1 Overview

In this section, we detail the implementation of K-BERT and its overall framework is presented in Figure 1：

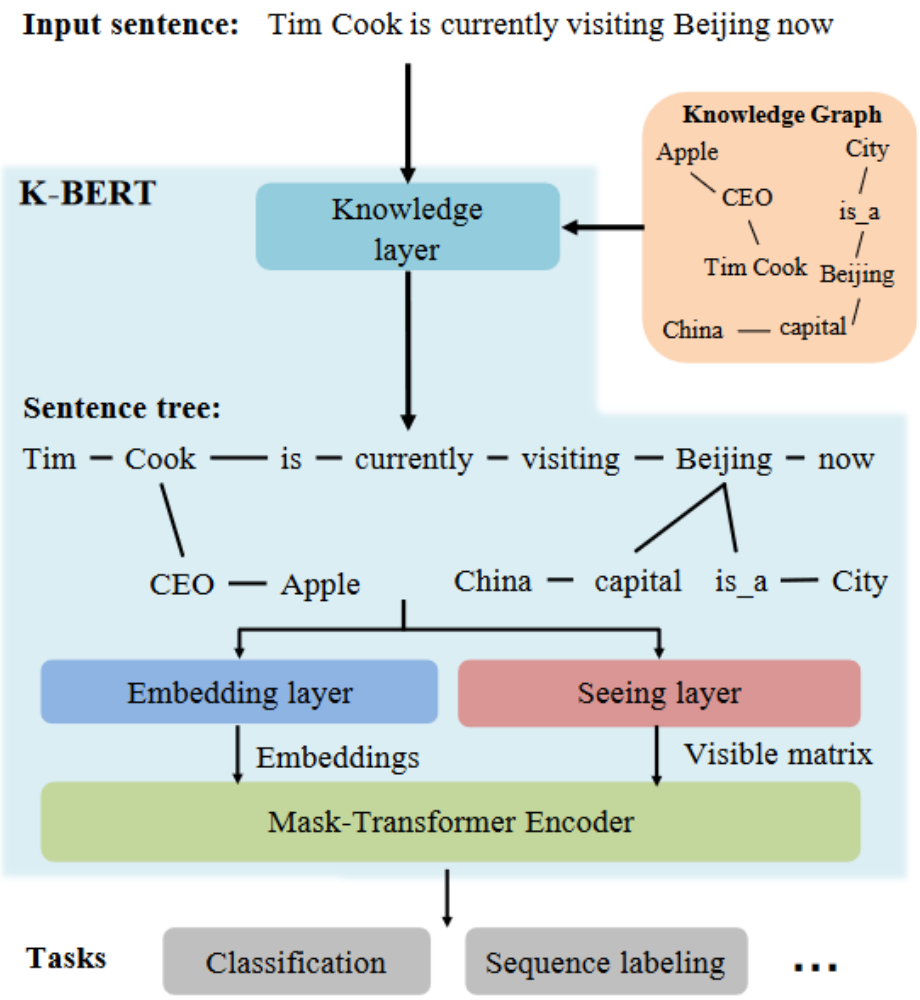

Figure 1: The model structure of K-BERT

3.2 Knowledge layer

The Knowledge Layer (KL) is used for sentence knowledge injection and sentence tree transformation. Specifically, given an input sentence s and a KG K, KL outputs a sentence tree t. This process can be divided

into two steps: knowledge query (K-Query) and knowledge injection (K-Inject). In K-Query, all entity names involved in sentence s are picked out and their corresponding triples are queried from K. Next, K-Inject injects the queried E into sentence s by stitching the three elements in E to their corresponding locations and generating a sentence tree t. The structure of t is illustrated in Figure 2.

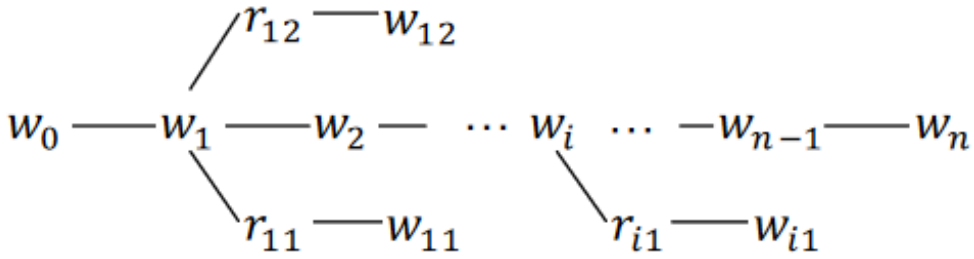

Figure 2: Structure of the sentence tree

3.3 Embedding layer

The function of the Embedding Layer (EL) is to convert the sentence tree into an embedding representation that can be fed into the Mask-Transformer. Similar to BERT, the embedding representation of K-BERT is the sum of three parts: token embedding, position embedding, and segment embedding, but differs in that the input of the K-BERT’s embedding layer is a sentence tree, rather than a token sequence. Therefore, how to transform a sentence tree into a sequence while retaining its structural information is the key to K-BERT.The structure of Embedding layer is illustrated in Figure 3.

3.4 Seeing layer

Seeing layer is the biggest difference between K-BERT and BERT, and also what make this method so effective. The input to K-BERT is a sentence tree, where the branch is the knowledge gained from KG. However, the risk raised with knowledge is that it can lead to changes in the meaning of the original sentence, i.e., KN issue. For example, in the sentence tree in Figure 2, [China] only modifies [Beijing] and has nothing to do with [Apple]. Therefore, the representation of [Apple] should not be affected by [China]. On the other hand, the [CLS] tag used for classification should not bypass the [Cook] to get the information of [Apple], as this would bring the risk of semantic changes. To prevent this from happening, K-BERT’s use a visible matrix M to limit the visible area of each token so that [Apple] and [China], [CLS] and [Apple] are not visible to each other.The structure of Seeing layer is illustrated in Figure 3.

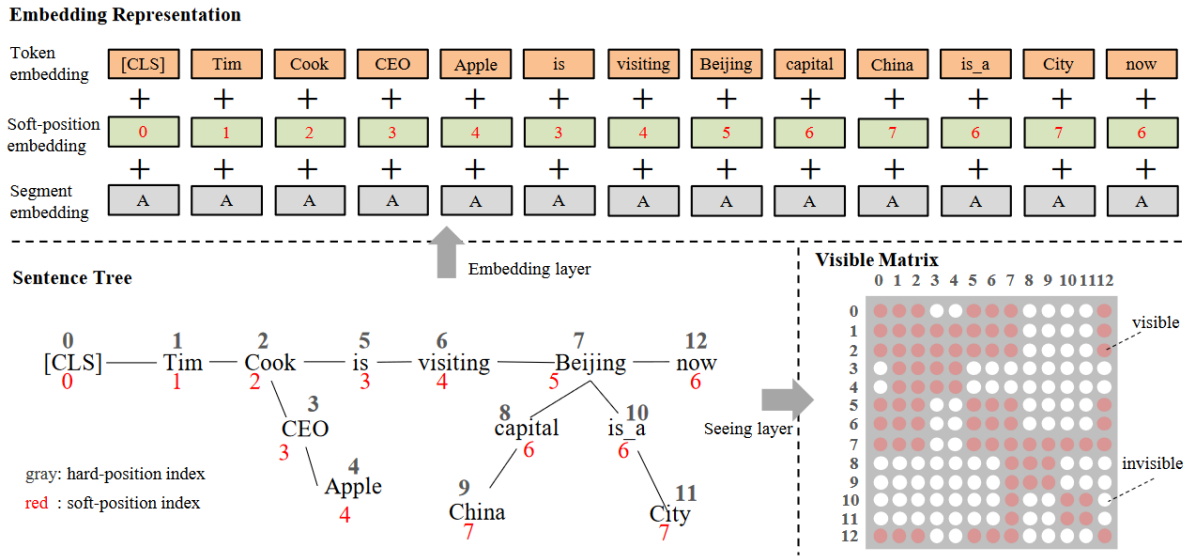

Figure 3: Structure of the Embedding layer and Seeing layer

4 Implementation details

4.1 Comparing with released source codes

In the process of replication, I referred to the source code published by the authors of the paper at the following link: https://github.com/autoliuweijie/K-BERT. I cited the model structure and the code for the named entity recognition task, and completed two tasks based on this：

- I completed the experiments for the text classification task in the thesis on my own.
- I adjusted the parameters of the model architecture to try to get better results.

4.2 Experimental parameter settings and training details

To reflect the role of KG in the RL model, we configure our K-BERT and BERT to the same parameter settings according to the basic version of Google BERT. We denote the number of self-attention layers and heads as L and A respectively, and the hidden dimension of embedding vectors as H. In detail, we have the following model configuration: L = 12, A = 12 and H = 768. The total amounts of trainable parameters of both BERT and K-BERT are the same (110M), which means that they are compatible with each other in model parameters. One thing to emphasize is that we don’t add any KG to K-BERT during the pre-training phase. Because KG binds two related entity names together, thus making the pre-trained word vectors of the two are very close or even equal and resulting in a semantic loss. Therefore, in the pre-training phase, K-BERT and BERT are equivalent, and the latter’s parameters can be assigned to the former. KG will be enabled during the fine-tuning and inferring phases.

4.3 Experimental datasets

4.3.1 Open-domain tasks

- **Book review** This dataset contains 20,000 positive and 20,000 negative reviews collected from Douban;
- **Chnsenticorp** Chnsenticorp is a hotel review dataset with a total of 12,000 reviews, including 6,000

positive reviews and 6,000 negative reviews;

• **Shopping** Shopping is a online shopping review dataset that contains 40,000 reviews, including 21,111 positive reviews and 18,889 negative reviews;

• **Weibo** Weibo is a dataset with emotional annotations from Sina Weibo, including 60,000 positive samples and 60,000 negative samples;

4.3.2 Specific-domain tasks

• **Domain QA** We crawl about 770,000 and 36,000 samples from Baidu Zhidao in financial and legal domains, including questions, netizen answers, and best answers. Based on this, we built two datasets, i.e., Finance QA and Law QA. The task is to choose the best answer for the question from the netizen answers.

• **Domain NER** Finance NER is a dataset including 3000 financial news articles manually labeled with over 65,000 name entities (people, location and organization). Medicine NER is the Clinical Named Entity Recognition (CNER) task released in CCKS 2017. The goal is to extract medical-related entity names from electronic medical records.

5 Results and analysis

5.1 Open-domain tasks

Each of the above datasets is divided into three parts: train, dev, and test. We use the train part to fine-tune the model and then evaluate its performance on the dev and test parts. The experimental results are shown in Figure 4, from which the results can be divided into three categories: (1) The KG has no significant effect on the tasks of sentiment analysis because the sentiment of a sentence can be judged based on emotional words without any knowledge; (2) The language KG (HowNet) performs better than the encyclopedic KG in terms of semantic similarity tasks; (3) For QA and NER tasks , the encyclopedic KG (CN-DBpedia) is more suitable than the language KG. Therefore, it is important to choose the right KG based on the type of task.

Models\Datasets	**Book_review**		**Chnsenticorp**		**Shopping**		**Weibo**		**XNLI**		**LCQMC**	
	Dev	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*
Pre-trainied on WikiZh by Google.												
Google BERT	88.3	**87.5**	93.3	94.3	96.7	96.3	98.2	98.3	76.0	75.4	88.4	86.2
K-BERT (HowNet)	88.6	87.2	**94.6**	**95.6**	**97.1**	**97.0**	98.3	98.3	**76.8**	**76.1**	**88.9**	86.9
K-BERT (CN-DBpedia)	**88.6**	87.3	93.9	95.3	96.6	96.5	98.3	98.3	76.5	76.0	88.6	**87.0**
Pre-trained on WikiZh and WebtextZh by us.												
Our BERT	**88.6**	87.9	94.8	95.7	96.9	**97.1**	98.2	98.2	77.0	76.3	89.0	86.7
K-BERT (HowNet)	88.5	87.4	**95.4**	95.6	96.9	96.9	98.3	**98.4**	**77.2**	**77.0**	**89.2**	**87.1**
K-BERT (CN-DBpedia)	88.8	87.9	95.0	**95.8**	**97.1**	97.0	98.3	98.3	76.2	75.9	89.0	86.9

Figure 4: Results of open-domain tasks

5.2 Specific-domain tasks

Similarly, the specific-domain datasets are split into three parts: train, dev, and test, which are used to fine-tune, select and test model, respectively. The test results of various models are illustrated in Figure 5,

where P., R. and F1 refer to Precision, Recall and F1-score, respectively. Compared with BERT, K-BERT has a significant performance improvement in terms of domain tasks. As for F1, K-BERT with CN-DBpedia can improve the performance of all tasks by 1-2%. The unique gain benefits from the domain knowledge in KG. Furthermore, it can be observed from the Medicine NER in Figure5 that the performance improvement using the MedicalKG is very obvious. From these results, we can conclude that KG, especially the domain KG, is very helpful for domain-specific tasks.

Models\Datasets	Finance_Q&A			Law_Q&A			Finance_NER			Medicine_NER		
	P.	*R.*	*F1*	*P.*	*R.*	*F1*	*P.*	*R.*	*F1*	*P.*	*R.*	*F1*
Pre-trained on WikiZh by Google.												
Google BERT	81.9	86.0	83.9	83.1	90.1	86.4	84.8	87.4	86.1	91.9	93.1	92.5
K-BERT (HowNet)	83.3	84.4	83.9	83.7	91.2	87.3	86.3	89.0	**87.6**	93.2	93.3	93.3
K-BERT (CN-DBpedia)	81.5	88.6	**84.9**	82.1	93.8	**87.5**	86.1	88.7	87.4	93.9	93.8	93.8
K-BERT (MedicalKG)	-	-	-	-	-	-	-	-	-	94.0	94.4	**94.2**
Pre-trained on WikiZh and WebtextZh by us.												
Our BERT	82.1	86.5	84.2	83.2	91.7	87.2	84.9	87.4	86.1	91.8	93.5	92.7
K-BERT (HowNet)	82.8	85.8	84.3	83.0	92.4	87.5	86.3	88.5	87.3	93.5	93.8	93.7
K-BERT (CN-DBpedia)	81.9	87.1	**84.4**	83.1	92.6	**87.6**	86.3	88.6	**87.4**	93.9	94.3	94.1
K-BERT (MedicalKG)	-	-	-	-	-	-	-	-	-	94.1	94.3	**94.2**

Figure 5: Results of Specific-domain tasks

6 Conclusion and future work

In this replication work, I learnt for the first time the combination of knowledge graphs and pre-trained models, familiarised myself with the architecture of the K-BERT model, and enhanced my ability to write code, all of which will help me in my subsequent research work. In the future, I will further familiarise myself with the architecture and principles of the model in order to improve the model structure and obtain better results.

References

[1] JOSHI M, CHEN D, LIU Y, et al. SpanBERT: Improving Pre-training by Representing and Predicting Spans[J]. Transactions of the Association for Computational Linguistics, 2019, 8: 64-77.

[2] LIU Y, OTT M, GOYAL N, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[J]. ArXiv, 2019, abs/1907.11692.

[3] ZHANG Z, HAN X, LIU Z, et al. ERNIE: Enhanced Language Representation with Informative Entities [J]., 2019.

[4] TOUTANOVA K, CHEN D, PANTEL P, et al. Representing Text for Joint Embedding of Text and Knowledge Bases[J]., 2015.

[5] HAN X, LIU Z, SUN M. Joint Representation Learning of Text and Knowledge for Knowledge Graph Completion[J]. ArXiv, 2016, abs/1611.04125.

[6] CAO Y, HOU L, LI J Z, et al. Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision[J]., 2018.