K-BERT: Enabling Language Representation with Knowledge Graph

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

1 Peking University, Beijing, China
2 Tencent Research, Beijing, China
3 Beijing Normal University, Beijing, China
{dataliu, pwang}@pku.edu.cn, SherronWang@gmail.com, {rickzhou, nlpzhe, zhiruowang, damonju, haotangdeng}@tencent.com

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), in which triples are injected into the sentences as domain knowledge. However, too much knowledge incorporation may divert the sentence from its correct meaning, which is called knowledge noise (KN) issue. To overcome KN, K-BERT introduces soft-position and visible matrix to limit the impact of knowledge. K-BERT can easily inject domain knowledge into the models by being equipped with a KG without pre-training by itself because it is capable of loading model parameters from the pre-trained BERT. Our investigation reveals 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.

Introduction
Unsupervised pre-trained Language Representation (LR) models like BERT (Devlin et al. 2018) 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. For example, the Google BERT pre-trained on Wikipedia can not give full play to its value when dealing with electronic medical record (EMR) analysis task in the medical field.

When reading a text from a specific-domain, ordinary people can only comprehend words according to its context, while experts are able to make inferences with relevant domain knowledge. Publicly-provided models, like BERT, GPT (Radford et al. 2018), and XLNet (Yang et al. 2019), who were pre-trained over open-domain corpora, act just like an ordinary people. Even though they can refresh the state-of-the-art of GLUE (Wang et al. 2018) benchmark by learning from open-domain corpora, they may fail in some domain-specific tasks, due to little knowledge connection between specific and open domain. One way to solve this problem is to pre-train a model emphasizing domain-specific by ourselves, instead of using the publicly-provided ones. However, pre-training is time-consuming and computationally expensive, making it unacceptable to most users.

Moreover, although injecting domain-specific knowledge during pre-training is possible for LR models, this process of knowledge acquisition can be inefficient and expensive. For example, if we want the model to acquire the knowledge of “Paracetamol can treat cold”, a large number of co-occurrences of ”Paracetamol” and ”cold” are required in the pre-training corpus. Instead of this strategy, what else can we do to make the model a domain expert? The knowledge graph (KG), which was called ontology in early research, serves as a good solution. As knowledge refined into a structured form, KGs over many fields have been constructed, e.g., SNOMED-CT (Bodenreider 2008) in medical field, HowNet (Dong, Dong, and Hao 2006) in Chinese conception. If KG can be integrated into the LR model, it will equip the model with domain knowledge, enhancing the model’s performance over domain-specific tasks, while reducing the cost of pre-training on a large scale. Besides, the resulting models posses greater interpretability, for the injected knowledge is manually editable.

However, there are two challenges lies in the road of this knowledge integration: (1) Heterogeneous Embedding Space (HES): In general, the embedding vectors of words in text and entities in KG are obtained in separate ways, making their vector-space inconsistent; (2) Knowledge Noise (KN): Too much knowledge incorporation may divert the sentence from its correct meaning. To overcome these challenges, this paper propose a Knowledge-enabled Bidirectional Encoder Representation from Transformers (K-BERT). K-BERT is capable of loading any pre-trained BERT models due to they are identical in parameters. In addition, K-BERT can easily inject domain knowledge into
the models by equipped with a KG without pre-training. This characteristic of K-BERT is very convenient for users with limited computing resources. Experimental results on twelve Chinese NLP tasks demonstrate that the K-BERT gains superior performances on domain-specific tasks. The main contributions of this paper can be summarized as follows:

- This paper proposes a knowledge-enabled LR model, namely K-BERT, which is compatible with BERT and can incorporate domain knowledge without HES and KN issues;
- With the delicate injection of KG, K-BERT significantly outperforms BERT not only on domain-specific tasks, but also plenty of those in the open-domain;
- The codes of K-BERT and our self-developed knowledge graphs are publicly available at https://github.com/autoliuweijie/K-BERT.

**Related Work**

Since Google Inc. launched BERT in 2018, many endeavors have been made for further optimization, basically focusing on the pre-training process and the encoder.

In optimizing pre-training process, Baidu-ERNIE (Sun et al. 2019) and BERT-WWM (Cui et al. 2019) adopt whole-word masking rather than single character masking for pre-training BERT in Chinese corpora. SpanBERT (Joshi et al. 2019) extended BERT by masking contiguous random spans and proposed a span boundary objective. RoBERTa (Liu et al. 2019) 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. In optimizing the encoder of BERT, XLNet (Yang et al. 2019) replaced the Transformer in BERT with Transformer-XL (Dai et al. 2019) to improve its ability to process long sentences. THU-ERNIE (Zhang et al. 2019) modified the encoder of BERT to an aggregator for the mutual integration of word and entities.

While the pre-trained LR model is an emerging direction, there is little work on its fusion with KG. THU-ERNIE (Zhang et al. 2019) is a pioneer in this direction by fusing entity information, but the relations between entities are ignored by it. COMET (Bosselut et al. 2019) employed the triples in KG as corpus to train GPT (Radford et al. 2018) for common sense learning, which is very inefficient.

Before the emergence of pre-trained LR models, there were several studies that combined KG with word vectors. Wang et al. (2014) proposed a novel method of jointly embedding entities and words into the same continuous vector space basing on the idea of word2vec (Mikolov et al. 2013). 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. Han, Liu, and Sun (2016) applied a convolutional neural network and a KG completion task to learn the representation of text and knowledge jointly. Cao et al. (2018) carried out cross-lingual representation learning for words and entities via attentive distant supervision.

**Methodology**

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

**Notation**

We denote a sentence $s = [w_0, w_1, w_2, ..., w_n]$ as a sequence of tokens, where $n$ is the length of this sentence. In this paper, English tokens are taken at the word-level, while Chinese tokens are at character-level. Each token $w_i$ is included in the vocabulary $\mathcal{V}$, $w_i \in \mathcal{V}$. KG, denoted as $\mathcal{K}$, is a collection of triples $\epsilon = (w_i, r_j, w_k)$, where $w_i$ and $w_k$ are the name of entities, and $r_j \in \mathcal{V}$ is the relation between them. All the triples are in KG.

**Model architecture**

As shown in Figure 1, the model architecture of K-BERT consists of four modules, i.e., knowledge layer, embedding layer, seeing layer and mask-transformer. For an input sentence, the knowledge layer first injects relevant triples into
it from a KG, transforming the original sentence into a knowledge-rich sentence tree. The sentence tree is then simultaneously fed into the embedding layer and the seeing layer and then converted to a token-level embedding representation and a visible matrix. The visible matrix is used to control the visible area of each token, preventing changing the meaning of the original sentence due to too much knowledge injected.

**Knowledge layer**

The knowledge layer (KL) is used for sentence knowledge injection and sentence tree conversion. Specifically, given an input sentence \( s = [w_0, w_1, w_2, ..., w_n] \) and a KG \( K \), KL outputs a sentence tree \( t = [w_0, w_1, ..., w_i(r_i0, r_i1), ..., (r_{ik}, w_{ik})], ..., w_n] \). This process can be divided into two steps: knowledge query (K-Query) and knowledge injection (K-Inject).

In K-Query, all the entity names involved in the sentence \( s \) are selected to query their corresponding triples from \( K \). K-Query can be formulated as (1),

\[
E = K_{Query}(s, K),
\]

where \( E = [(w_i, r_i0, w_i0), ..., (w_i, r_{ik}, w_{ik})] \) is a collection of the corresponding triples.

Next, K-Inject injects the queried \( E \) into the sentence \( s \) by stitching the triples in \( E \) to their corresponding position, and generates a sentence tree \( t \). The structure of \( t \) is illustrated in Figure 3.

In this paper, a sentence tree can have multiple branches, but its depth is fixed to 1, which means that the entity names in triples will not derive branches iteratively. K-Inject can be formulated as (2).

\[
t = K_{Inject}(s, E). \tag{2}
\]

**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.

**Token embedding** In this work, the token embedding is consistent with BERT, and the vocabulary provided by Google BERT is adopted in this paper. Each token in the sentence tree is converted to an embedding vector of dimension \( H \) via a trainable lookup table. In addition, K-BERT
also uses [CLS] as a classification tag and uses [MASK] to mask tokens. The difference between the token embeddings of K-BERT and BERT is that the tokens in the sentence tree require rearrangement before the embedding operation. In our re-arrange strategy, tokens in the branch are inserted after the corresponding node, while subsequent tokens are moved backwards. As the example illustrated in Figure 2, the sentence tree is rearranged as “Tim Cook CEO Apple is visiting Beijing capital China is a City now”. Although this procedure is simple, it makes the sentence unreadable and lost correct structural information. Fortunately, it can be solved by the soft-position and visible matrix.

**Soft-position embedding** For BERT, if there is no position embedding, it will be equivalent to a bag-of-word model, resulting in a lack of structural information (i.e., the order of tokens). All the structural information of the BERT’s input sentence is contained in the position embedding, which allows us to add the missing structural information back to the unreadable rearranged sentence. Taking the sentence tree in Figure 2 as an example, after rearranging, [CEO] and [Apple] are inserted between [Cook] and [is], but the subject of [is] should be [Cook] instead of [Apple]. To solve this problem, we only need to set the position number of [is] to 3 instead of 5. So when calculating the self-attention score in the transformer encoder, [is] is at the next position of [Cook] by the equivalent. However, another problem arises: the position numbers of [is] and [CEO] are both 3, which makes them close in position when calculating self-attention, but in fact, there is no connection between them. The solution to this problem is Mask-Self-Attention, which will be covered in the next subsection.

**Segment embedding** Like BERT, K-BERT also uses segmentation embedding to identify different sentences when multiple sentences are included. For example, when two sentences \([w_0, w_1, \ldots, w_m]\) and \([w_{10}, w_{11}, \ldots, w_{1m}]\) are inputted, they are combined into one sentence \([CLS], w_0, w_1, \ldots, w_m, [SEP], w_{10}, w_{11}, \ldots, w_{1m}\) with [SEP]. For the combined sentence, it is marked with a sequence of segment tags, \([A, A, A, \ldots, A, B, B, \ldots, B]\).

**Seeing layer**

Seeing layer is the biggest difference between K-BERT and BERT, and also what makes 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 visible matrix \(M\) is defined as (3),

\[
M_{ij} = \begin{cases} 0 & w_i \not\supset w_j \\ -\infty & w_i \supset w_j \end{cases}
\]

(3) where, \(w_i \not\supset w_j\) indicates that \(w_i\) and \(w_j\) are in the same branch, while \(w_i \supset w_j\) are not. \(i\) and \(j\) are the hard-position index.

**Mask-Transformer**

To some degree, the visible matrix \(M\) contains the structural information of the sentence tree. The Transformer (Vaswani et al. 2017) encoder in BERT cannot receive \(M\) as an input, so we need to modify it to Mask-Transformer, which can limit the self-attention region according to \(M\). Mask-Transformer is a stack of multiple mask-self-attention blocks. As BERT, we denote the number of layers (i.e., mask-self-attention blocks) as \(L\), the hidden size as \(H\), and the number of mask-self-attention heads as \(A\).

**Mask-Self-Attention:** To prevent false semantic changes by taking advantage of the sentence structure information in \(M\), we propose a mask-self-attention, which is an extension of self-attention. Formally, the mask-self-attention is (4).

\[
Q^{i+1}, K^{i+1}, V^{i+1} = h^iW_q, h^iW_k, h^iW_v,
\]

(4)

\[
S^{i+1} = \text{softmax}(Q^{i+1}K^{i+1T} + M)\sqrt{d_k},
\]

(5)

\[
h^{i+1} = S^{i+1}V^{i+1},
\]

(6)

where \(W_q, W_k \text{ and } W_v\) are trainable model parameters. \(h^i\) is the hidden state of the \(i\)-th mask-self-attention blocks. \(d_k\) is the scaling factor, \(M\) is the visible matrix calculated by the seeing layer. Intuitively, if \(w_k\) is invisible to \(w_j\), the \(M_{jk}\) will mask the attention score \(S_{jk}^{i+1}\) to 0, which means \(w_k\) make no contribution to the hidden state of \(w_j\).

As the example illustrated in Figure 4, \(h_{11}^{i+1}[Apple]\) has no effect on \(h_{11}^{i+1}[CLS]\) because [Apple] is invisible to [CLS]. However, \(h_{11}^{i+1}[CLS]\) can obtain the information of \(h_{11}^{i+1}[Apple]\) indirectly.
through $h^{t+1}_\text{[Cook]}$ because $\text{Apple}$ is visible to $\text{Cook}$ and $\text{Cook}$ is visible to $\text{CLS}$. The advantage of this process is that $\text{Apple}$ enriches the representation of $\text{Cook}$, but does not directly affect the meaning of the original sentence.

**Experiments**

In this section, we present the details of the K-BERT fine-tuning results on twelve Chinese NLP tasks, among which eight are open-domain, and four are specific-domain.

**Pre-training corpora**

In this paper, we adopt two Chinese corpora for pre-training, i.e., WikiZh\(^2\) and WebtextZh\(^3\).

- **WikiZh** WikiZh refers to the Chinese Wikipedia corpus, which is used to train Chinese BERT in (Devlin et al. 2018). WikiZh contains a total of 1 million well-formed Chinese entries with 12 million sentences and size of 1.2G.

- **WebtextZh** WebtextZh is a large-scale, high-quality Chinese question and answer (Q&A) corpus with 4.1 million entries and a size of 3.7G. Each entry in WebtextZh belongs to a topic, with a total of 28,000 topics.

**Knowledge graph**

We employ three Chinese KGs, CN-DBpedia\(^4\), HowNet\(^5\), and MedicalKG.

- **CN-DBpedia** CN-DBpedia (Xu et al. 2017) is a large-scale open-domain encyclopedic KG developed by the Knowledge Work Laboratory of Fudan University, covering tens of millions of entities and hundreds of millions of relations. In this paper, we refine the official CN-DBpedia by eliminating those triples whose entity names are less than 2 in length or contain special characters. The refined CN-DBpedia contains a total of 5.17 million triples.

- **HowNet** HowNet is a large-scale language knowledge base for Chinese vocabulary and concepts (Dong, Dong, and Hao 2006), in which each Chinese word is annotated with semantic units called sememes. If we take \{word, contain, sememes\} as a triple, HowNet is a language KG. Similarly, we refine the official HowNet by eliminating those triples whose entity names are less than 2 in length or contain special characters. The refined HowNet contains a total of 52,576 triples.

- **MedicalKG** MedicalKG is our self-developed Chinese medical concept KG, which contains four types of hypernym (symptoms, diseases, parts, and treatments). MedicalKG contains a total of 13,864 triples and is open source as part of K-BERT.

**Baselines**

In this paper, we compare K-BERT to two baselines:

- **Google BERT** The model was pre-trained on WikiZh and released by Google (Devlin et al. 2018).

- **Our BERT** Our reimplementation of BERT with pre-training on WikiZh and WebtextZh, released in UER\(^7\).

**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 (Devlin et al. 2018). We denote the number of (mask-)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.

For K-BERT pre-training, all settings are consistent with (Devlin et al. 2018). 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.

**Open-domain tasks**

We first compare the performance of K-BERT with BERT on eight Chinese open-domain NLP tasks. Among these eight tasks, Book\_review\(^8\), Chnsenticorp\(^9\), Shopping\(^10\), and Weibo\(^11\) are single-sentence classification tasks:

- **Book\_review** This dataset contains 20,000 positive and 20,000 negative reviews collected from Douban\(^12\);

- **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 an 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.

XNLI (Conneau et al. 2018), LCQMC (Liu et al. 2018) are two-sentence classification tasks, NLPCC-DBQA\(^13\) is a Q&A matching task, and MSRA-NER (Levow 2006) is a Named Entity Recognition (NER) task:

\[^6\]https://github.com/google-research/bert
\[^7\]https://github.com/dhibi/UR-py
\[^8\]https://embedding.github.io/evaluation/
\[^9\]https://github.com/pengming617/bert\_classification
\[^10\]https://share.weiyun.com/5xxYig
\[^11\]https://share.weiyun.com/S1EsV0W
\[^12\]https://book.douban.com/
Table 1: Results of various models on sentence classification tasks on open-domain tasks (Acc. %)

<table>
<thead>
<tr>
<th>Models\Datasets</th>
<th>Book_review</th>
<th>Chnsenticorp</th>
<th>Shopping</th>
<th>Weibo</th>
<th>XNLI</th>
<th>LCQMC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
</tr>
<tr>
<td>Google BERT</td>
<td>88.3 87.5</td>
<td>93.3 94.3</td>
<td>96.7 96.3</td>
<td>98.2 98.3</td>
<td>76.0 75.4</td>
<td>88.4 86.2</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>88.6 87.2</td>
<td><strong>94.6 95.6</strong></td>
<td><strong>97.1 97.0</strong></td>
<td>98.3 98.3</td>
<td><strong>76.8 76.1</strong></td>
<td>88.9 86.9</td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td><strong>88.6 87.3</strong></td>
<td>93.9 95.3</td>
<td>96.6 96.5</td>
<td>98.3 98.3</td>
<td>76.5 76.0</td>
<td><strong>88.6 87.0</strong></td>
</tr>
</tbody>
</table>

Pre-trained on WikiZh by Google.

<table>
<thead>
<tr>
<th>Models\Datasets</th>
<th>Book_review</th>
<th>Chnsenticorp</th>
<th>Shopping</th>
<th>Weibo</th>
<th>XNLI</th>
<th>LCQMC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
<td>Dev Test</td>
</tr>
<tr>
<td>Our BERT</td>
<td><strong>88.6 87.9</strong></td>
<td>94.8 95.7</td>
<td>96.9 97.1</td>
<td>98.2 98.2</td>
<td>77.0 76.3</td>
<td>89.0 86.7</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>88.5 87.4</td>
<td><strong>95.4 95.6</strong></td>
<td>96.9 96.9</td>
<td>98.3 <strong>98.4 77.2</strong></td>
<td>77.0 89.2</td>
<td></td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td>88.8 87.9</td>
<td>95.0 <strong>95.8</strong></td>
<td><strong>97.1 97.0</strong></td>
<td>98.3 98.3</td>
<td>76.2 75.9</td>
<td>89.0 86.9</td>
</tr>
</tbody>
</table>

Pre-trained on WikiZh and WebtextZh by us.

Table 2: Results of various models on NLPCC-DBQA (MRR %) and MSRA-NER (F1 %)

<table>
<thead>
<tr>
<th>Models\Datasets</th>
<th>NLPCC-DBQA</th>
<th>MSRA-NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev Test</td>
<td>Dev Test</td>
</tr>
<tr>
<td>Pre-trained on WikiZh by Google.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google BERT</td>
<td>93.4 93.3</td>
<td>94.5 93.6</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>93.2 93.1</td>
<td>95.8 94.5</td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td><strong>94.5 94.3</strong></td>
<td><strong>96.6 95.7</strong></td>
</tr>
</tbody>
</table>

Pre-trained on WikiZh and WebtextZh by us.

<table>
<thead>
<tr>
<th>Models\Datasets</th>
<th>NLPCC-DBQA</th>
<th>MSRA-NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev Test</td>
<td>Dev Test</td>
</tr>
<tr>
<td>Our BERT</td>
<td>93.3 93.6</td>
<td>95.7 94.6</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>93.2 93.1</td>
<td>96.3 95.6</td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td><strong>93.6 94.2</strong></td>
<td><strong>96.4 95.6</strong></td>
</tr>
</tbody>
</table>

- XNLI is a cross-language language understanding dataset in which each entry contains two sentences and the task is to determine their relation ("Entailment", "Contradict" or "Neutral");
- LCQMC is a large-scale Chinese question matching corpus. The goal of this task is to determine if the two questions have a similar intent;
- NLPCC-DBQA is a task to predict answers to each question from the given document;
- MSRA-NER is a NER dataset published by Microsoft. This task is to recognize the entity names in the text, including person names, place names, organization names, etc.

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 Table 1 and 2, from which the results can be divided into three categories: (1) The KG has no significant effect on the tasks of sentiment analysis (i.e., Book_review, Chnsenticorp, Shopping and Weibo) 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 (i.e., XNLI and LCQMC); (3) For Q&A and NER tasks (i.e., NLPCC-DBQA and MSRA-NER), 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.

In addition, it can be observed that the use of an additional corpus (WebtextZh) can also bring performance boost, but not as significant as KG. As MSRA-NER shown in Table 2, the CN-DBpedia improves F1 from 93.6% to 95.7%, while the WebtextZh only increases it to 94.6%.

Specific-domain tasks

In fact, the task where K-BERT really shines is in specific-domain. Because KG is good at giving LR model with domain knowledge.

Domain Q&A We crawl about 770,000 and 36,000 Q&A samples from Baidu Zhidao\(^{14}\) in financial and legal domains, including questions, netizen answers, and best answers. Based on this, we built two datasets, i.e., Finance\_Q&A and Law\_Q&A. The task is to choose the best answer for the question from the netizen’s answers.

Domain NER Finance\_NER\(^{15}\) 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\(^{16}\). The goal is to extract medical-related entity names from electronic medical records.

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 Table 3, 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,

\(^{14}\)https://zhidao.baidu.com
\(^{15}\)https://embedding.github.io/evaluation/#extrinsic
\(^{16}\)https://biendata.com/competition/CCKS2017/2/
Table 3: Results of various models on specific-domain tasks (%).

<table>
<thead>
<tr>
<th>Models \ Datasets</th>
<th>Finance Q&amp;A</th>
<th>Law Q&amp;A</th>
<th>Finance NER</th>
<th>Medicine NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.</td>
<td>R.</td>
<td>F1</td>
<td>P.</td>
</tr>
<tr>
<td>Google BERT</td>
<td>81.9</td>
<td>86.0</td>
<td>83.9</td>
<td>83.1</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>83.3</td>
<td>84.4</td>
<td>83.9</td>
<td>83.7</td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td>81.5</td>
<td>88.6</td>
<td><strong>84.9</strong></td>
<td>82.1</td>
</tr>
<tr>
<td>K-BERT (MedicalKG)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pre-trained on WikiZh by Google.

<table>
<thead>
<tr>
<th>Models \ Datasets</th>
<th>Finance Q&amp;A</th>
<th>Law Q&amp;A</th>
<th>Finance NER</th>
<th>Medicine NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.</td>
<td>R.</td>
<td>F1</td>
<td>P.</td>
</tr>
<tr>
<td>Our BERT</td>
<td>82.1</td>
<td>86.5</td>
<td>84.2</td>
<td>83.2</td>
</tr>
<tr>
<td>K-BERT (HowNet)</td>
<td>82.8</td>
<td>85.8</td>
<td>84.3</td>
<td>83.0</td>
</tr>
<tr>
<td>K-BERT (CN-DBpedia)</td>
<td>81.9</td>
<td>87.1</td>
<td><strong>84.4</strong></td>
<td>83.1</td>
</tr>
<tr>
<td>K-BERT (MedicalKG)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Pre-trained on WikiZh and WebtextZh by us.

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 Table 3 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.

Ablation studies

![Figure 5: Ablation studies: (a) Law Q&A with CN-DBpedia; (b) Medicine NER with MedicalKG;](image)

In this subsection, we explore the effects of the soft-position and visible matrix for K-BERT using two domain-specific tasks (Law Q&A and Medicine NER). “w/o soft-position” refers to fine-tuning K-BERT with hard-position instead of soft-position. “w/o visible matrix” means that the all tokens are visible to each other. BERT is equivalent to the K-BERT without KG. As shown in Figure 5, we have the following observations: (1) In both tasks, without soft-position or visible matrix, the performance of K-BERT has declined; (2) In Law Q&A (Figure 5(a)), K-BERT without visible matrix is worse than BERT, which confirms the existence of KN, i.e., improperly adding knowledge can lead to performance degradation; (3) In Figure 5(a), K-BERT reaches its peak at epoch 2, while BERT is at epoch 4, which proves that K-BERT converges faster than BERT. In general, we can conclude that the soft-position and the visible matrix can make K-BERT more robust to KN interference and thus make more efficient use of knowledge.

Conclusions

In this paper, we propose K-BERT to enable language representation with knowledge graphs, achieving the capability of commonsense or domain knowledge. Specifically, K-BERT first injects knowledge from KG into a sentence, making it a knowledge-rich sentence tree. Next, soft-position and visible matrix are adapted to control the scope of knowledge, preventing it from deviating from its original meaning.

Despite the challenges of HES and KN, our investigation reveals promising results on twelve open-/specific-domain NLP tasks. Empirical results demonstrate that KG is especially helpful for knowledge-driven specific-domain tasks and can be used to solve problems that require domain experts. Besides, K-BERT is compatible with the model parameters of BERT, which means that users can directly adopt the available pre-trained BERT parameters (e.g., Google BERT, Baidu-ERNIE, etc.) on K-BERT without pre-training by themselves.

These positive results point to future work in (1) improving K-Query to enable filtering of unimportant triples based
on context; (2) extending this approach to other LR models such as ELMo (Peters et al. 2018), XLNet (Yang et al. 2019), etc;

Acknowledgement

This work is funded by the National Key R&D Program of China (2017YFB1200700), and 2019 Tencent Rhino-Bird Elite Training Program.

References


Dong, Z.; Dong, Q.; and Hao, C. 2006. Hownet and the computation of meaning. Citeseer.

