Revisiting and Advancing Chinese Natural Language Understanding with Accelerated Heterogeneous Knowledge Pre-training

Taolin Zhang[1,2], **Junwei Dong**[2,3], **Jianing Wang**[1,2], **Chengyu Wang**[2]*, **Ang Wang**[2], **Yinghui Liu**[2], **Jun Huang**[2], **Yong Li**[2], **Xiaofeng He**[1]
[1] East China Normal University, Shanghai, China
[2] Alibaba Group, Hangzhou, China
[3] Chongqing University, Chongqing, China
zhangtl0519@gmail.com, chengyu.wcy@alibaba-inc.com

Abstract

Recently, knowledge-enhanced pre-trained language models (KEPLMs) improve context-aware representations via learning from structured relations in knowledge graphs, and/or linguistic knowledge from syntactic or dependency analysis. Unlike English, there is a lack of high-performing open-source Chinese KEPLMs in the natural language processing (NLP) community to support various language understanding applications. In this paper, we revisit and advance the development of Chinese natural language understanding with a series of novel Chinese KEPLMs released in various parameter sizes, namely CKBERT (Chinese knowledge-enhanced BERT). Specifically, both relational and linguistic knowledge is effectively injected into CKBERT based on two novel pre-training tasks, i.e., linguistic-aware masked language modeling and contrastive multi-hop relation modeling. Based on the above two pre-training paradigms and our in-house implemented TorchAccelerator, we have pre-trained base (110M), large (345M) and huge (1.3B) versions of CKBERT efficiently on GPU clusters. Experiments demonstrate that CKBERT outperforms strong baselines for Chinese over various benchmark NLP tasks and in terms of different model sizes. [1]

1 Introduction

Pre-trained Language Models (PLMs) such as BERT (Devlin et al., 2019) are pre-trained by self-supervised learning on large-scale text corpora to capture the rich semantic knowledge of words (Li et al., 2021; Gong et al., 2022), improving various downstream NLP tasks significantly (He et al., 2020; Xu et al., 2021; Chang et al., 2021). Although these PLMs have stored much internal knowledge (Petroni et al., 2019, 2020), they can hardly understand external background knowledge from the world such as factual and linguistic knowledge (Colon-Hernandez et al., 2021; Cui et al., 2021; Lai et al., 2021).

In the literature, most approaches of knowledge injection can be divided into two categories, including relational knowledge and linguistic knowledge. (1) Relational knowledge-based approaches inject entity and relation representations in Knowledge Graphs (KGs) trained by knowledge embedding algorithms (Zhang et al., 2019; Peters et al., 2019) or convert triples into sentences for joint pre-training (Liu et al., 2020; Sun et al., 2020). (2) Linguistic knowledge-based approaches extract semantic units from pre-training sentences such as part-of-speech tags, constituent and dependency syntactic parsing, and feed all linguistic information into various transformer-based architectures (Zhou et al., 2020; Lai et al., 2021). We observe that there can be three potential drawbacks. (1) These approaches generally utilize a single source of knowledge (i.e., inherent linguistic knowledge), which ignore important knowledge from other sources (Su et al., 2021) (i.e., relational knowledge from KGs). (2) Training large-scale KEPLMs from scratch requires high-memory computing devices and is time-consuming, which brings significant computational burdens for users (Zhang et al., 2021, 2022). (3) Most of these models are pre-trained in English only. There is a lack of powerful KEPLMs for understanding other languages (Lee et al., 2020; Pérez et al., 2021).

To overcome the above problems, we release a series of Chinese KEPLMs named CKBERT (Chinese knowledge-enhanced BERT), with heterogeneous knowledge sources injected. We particularly focus on Chinese as it is one of the most widely spoken languages other than English. The CKBERT models are pre-trained by two well-designed pre-training tasks as follows:

- **Linguistic-aware Masked Language Mod-**

* Corresponding author.

[1] All the codes and model checkpoints have been released to public in the EasyNLP framework (Wang et al., 2022). URL: https://github.com/alibaba/EasyNLP.

Proceedings of EMNLP 2022 Industry Track, pages 570–580
December 9–11, 2020.

eling (LMLM): LMLM is substantially extended from Masked Language Modeling (MLM) (Devlin et al., 2019) by introducing two key linguistics tokens derived from dependency syntactic parsing and semantic role labeling. We also insert unique markers for each linguistic component among contiguous tokens. The goal of LMLM is to predict both randomly selected tokens and linguistic tokens masked in the pre-training sentences.

- **Contrastive Multi-hop Relation Modeling (CMRM):** We sample fine-grained subgraphs from a large-scale Chinese KG by multi-hop relations to compensate for understanding the background knowledge of target entities. Specifically, we construct positive triples for matched target entities via retrieving one-hop entities in the corresponding subgraphs. Negative triples are sampled from unrelated multi-hop entities through the relation paths in the KG. The CMRM task is proposed to pull the semantics of similar entities close and push away those with irrelevant semantics.

Based on the above heterogeneous knowledge pre-training tasks, we produce various sizes of CKBERT models to meet the inference time and accuracy requirements of different real-world scenarios (Brown et al., 2020; Chowdhery et al., 2022), including base (110M), large (345M) and huge (1.3B). The models are pre-trained using our in-house implemented TorchAccelerator that effectively transforms PyTorch eager execution to graph execution on distributed GPU clusters, boosting the training speed by 40% per sample with our advanced compiler technique based on Accelerated Linear Algebra (XLA). In the experiments, we compare CKBERT against strong baseline PLMs and KEPLMs on various Chinese general and knowledge-related NLP tasks. The results demonstrate the improvement of CKBERT compared to SoTA models.

2 Related Work

We briefly summarize the related work on the following two aspects: PLMs and KEPLMs.

2.1 PLMs

Following BERT (Devlin et al., 2019), many PLMs have been proposed to improve performance in various NLP tasks. Several approaches extend BERT by employing novel token-level and sentence-level pre-training tasks. Notable PLMs include ERNIE-Baidu (Sun et al., 2019), MacBERT (Cui et al., 2020) and PERT (Cui et al., 2022) for Chinese NLU downstream tasks. Other models boost the performance by changing the internal encoder architectures. For example, XLNet (Yang et al., 2019) utilizes Transformer-XL (Dai et al., 2019) to encode long sequences by the permutation in language tokens. Sparse self-attention (Cui et al., 2019) replaces the self-attention mechanism with more interpretable attention units. Yet, other PLMs such as MT-DNN (Liu et al., 2019) combine self-supervised pre-training with the multi-task supervised learning to improve the performance of various GLUE tasks (Wang et al., 2019).

2.2 KEPLMs

These models use structured knowledge or linguistic semantics to enhance the language understanding abilities of PLMs. We summarize recent KEPLMs grouped into the following four types: (1) Knowledge-enhancement by linguistic semantics. These works use the linguistic information already available in the pre-training sentences to enhance the understanding ability of PLMs. Lattice-BERT (Lai et al., 2021) pre-trains a Chinese PLM over a word lattice (Buckman and Neubig, 2018) structure to exploit multi-granularity inputs. (2) Knowledge-enhancement by entity embeddings. For example, ERNIE-THU (Zhang et al., 2019) injects entity embeddings into contextual representations via knowledge-encoders stacked by the information fusion module. (3) Knowledge-enhancement by entity descriptions. These approaches learn entity embeddings by knowledge descriptions. For example, pre-training corpora and entity descriptions in KEPLER (Wang et al., 2021) are encoded into a unified semantic space within the same PLM. (4) Knowledge-enhancement by converted triplet's texts. K-BERT (Liu et al., 2020) and CoLAKE (Sun et al., 2020) convert relation triplets into texts and insert them into training samples without using pre-trained embeddings. In this paper, we argue that aggregating heterogeneous knowledge information can further benefit the context-aware representations of PLMs.

3 Model

In this section, we elaborate the techniques of the proposed CKBERT model. The main architecture

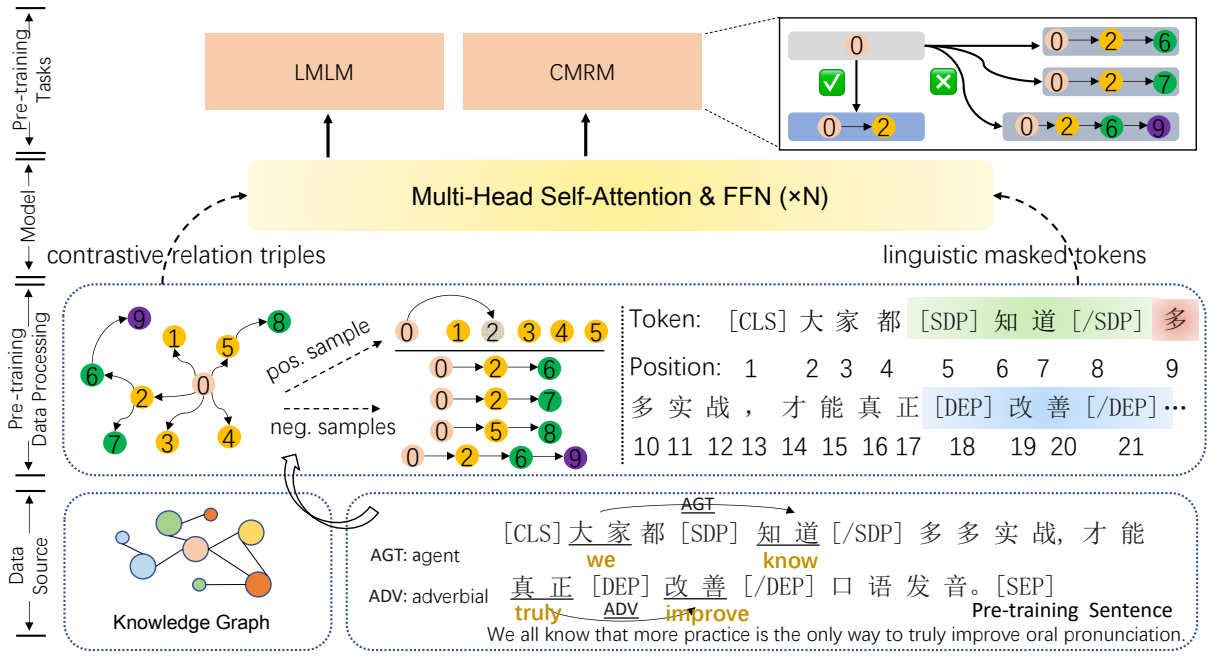

Figure 1: Model overview. The LMLM task is not only able to perform random masked token prediction (similar to BERT) but also to predict masked linguistic-aware tokens. The CMRM task injects external relation triples into PLMs through neighboring multi-hop relations. (Best viewed in color.)

of CKBERT is firstly presented in Figure 1.

3.1 Model Architecture

It accepts a sequence of M WordPiece tokens (Wu et al., 2016), $(x_1, x_2, ..., x_M)$ as input, and computes the D-dimensional contextual representations $H_i \in \mathbb{R}^{M \times D}$ by successively stacking N transformer encoder layers. We do not modify the architecture here to guarantee that CKBERT can be seamlessly integrated into any industrial applications that BERT supports with better performance.[2]

3.2 Linguistic-aware Masked Language Modeling (LMLM)

In BERT pre-training, 15% of all token positions are randomly masked for prediction. However, random masked tokens may be unimportant units such as conjunctions and prepositions (Clark et al., 2019; Hao et al., 2021). We reconstruct the input sentences and mask more tokens based on linguistic knowledge so that CKBERT can better understand the semantics of important tokens in pre-training sentences. Specifically, we use the following three steps to mask the linguistic input units:

- **Recognizing Linguistic Tokens:** We first use the off-the-shelf tool[3] to recognize important units in pre-training sentences, including dependence grammar and semantic dependency parsing. The extracted relations here serve as important sources of linguistic knowledge, including "subject-verb", "verb-object" and "adverbial" for dependence grammar and "non-agent" for semantic dependency parsing.

- **Reconstructing Input Sentences:** In addition to the original input form, based on the subjects and objects of the extracted linguistic relations, we insert special identifiers for each lexicon unit between words spans to give explicit boundary information for model pre-training. For example, we add `[DEP]` and `[/DEP]` for dependence grammar and `[SDP]` and `[/SDP]` for dependency parsing tokens.

- **Choosing Masked Tokens:** We choose 15% of token positions from the reconstructed input sentence for masking, using the special token `[MASK]`. Among these tokens, we assign 40% of the positions to randomly selected tokens and the rest to linguistic tokens. Note that these special identifiers (`[DEP]`, `[/DEP]`, `[SDP]` and `[/SDP]`) are also treated as normal tokens for masking, thus the model needs to

[2]Without loss of generality, we focus on the transformer encoder architecture only; yet our work can also be extended model architectures with slight modification.

[3]http://ltp.ai/

be aware of predicting word boundaries rather than simply filling in masks based on contexts.

After input sentences are processed, for LMLM, let $\Omega = (m_1, m_2, m_3, ..., \gamma_{K-1}, \gamma_K)$ denote the indexes of the masked tokens in the sentence X, where m_i is an index of a randomly masked token, γ_i is an index of a selected linguistic-aware masked token and K is the total number of masked tokens. Let X_Ω denote the set of masked tokens in X, and $X_{-\Omega}$ denote the set of observed (unmasked) tokens. The objective of LMLM is as follows:

$$\mathcal{L}_{mlm}(X_\Omega|X_{-\Omega}) = \frac{1}{K}\sum_{k=1}^{K}\log p(x_{m_k|\gamma_k}|X_{-\Omega};\theta) \tag{1}$$

where $x_{m_k|\gamma_k}$ denotes the randomly selected tokens or the linguistic tokens. θ represents the parameter collection of our model.

3.2.1 Contrastive Multi-hop Relation Modeling (CMRM)

In addition to LMLM, we further inject relation triples into CKBERT to make it understand the background factual knowledge of entities. For an entity in the pre-training sentence, we construct positive and negative relation triples as follows:

- **Positive Triples:** We employ entity linking to link an entity in the pre-training sentence to the target entity e_t in the KG. The relation triples w.r.t. the one-hop entities are viewed as candidate positive triples. Next, we choose a relation triple randomly from the candidates as a positive sample, denoted as t_p.
- **Negative Triples:** Because the semantic similarity between the positive triple t_p and the relation triples along the KG paths decreases, we construct L candidate negative triples $(t_n^1, t_n^2, ..., t_n^L)$ by making multiple hops starting from the target entity e_t. For example, in Fig. 1, we take the target entity e_0 as the starting node and retrieve the nodes along the edges. We obtain the ending node e_{end} with multi-hop relations $Hop(\mathcal{G}, e_0, e_{end}, r)$, where $Hop(\cdot)$ means the shortest distance between e_0 and e_{end} in KG $\mathcal{G}$. Here, we regard a triple to be negative t_n if $Hop(\cdot) > 1$ and is no larger than a small threshold δ[4]. In this paper, we set $\delta = 3$. Hence, there are four negative triples for e_0 in Figure 1. A sample three-hop path is $e_0 \rightarrow e_2 \rightarrow e_6 \rightarrow e_9$.

The CMRM task is designed for pulling similar relational triples of the target entity closely and pushing unrelated multi-hop relational triples away, in order to enhance the external background knowledge of the target entity from the KG. Concretely, after the positive sample t_p and negative samples $(t_n^1, t_n^2, ..., t_n^L)$ of the target entity e_t are retrieved, the context-aware representations of the target entity e_t can be obtained as follows:

$$h_{e_t} = \mathcal{LN}\left(\sigma\left(f_{sp}\left(h_{e_t^i}, \ldots, h_{e_t^j}\right)\mathrm{W}_1\right)\right) \tag{2}$$

where h_{e_t} is the hidden representation of the target entity e_t constructed by the entity's token representations $\left(h_{e_t^i}, \ldots, h_{e_t^j}\right)$, as an entity can have multiple tokens in the pre-training sentence. f_{sp} is the self-attentive pooling operator (Lin et al., 2017), $\sigma(\cdot)$ non-linear activation function GELU (Hendrycks and Gimpel, 2016) and $\mathcal{LN}(\cdot)$ is the LayerNorm function (Ba et al., 2016). W_1 is the learnable weight matrix.

Meanwhile, as relation triples can be viewed as natural sentences via concatenating the triple's tokens together, following Liu et al. (2020); Sun et al. (2020), we convert the triples into sentences to generate the representations obtained by the shared encoder θ (which is the transformer encoder of our CKBERT model). Hence, the representations of the positive triple h_{t_p} and the negative triples $(h_{t_n^1}, h_{t_n^2}, ..., h_{t_n^L})$ can also be derived. For the CMRM task, we employ InfoNCE (van den Oord et al., 2018) as the loss function to calculate the similarity as follows:

$$\mathcal{L}_{cl} = -\log\frac{\exp\left(cos\left(h_{e_t}, h_{t_p}\right)/\tau\right)}{\sum_{l=1}^{L}\exp\left(cos\left(h_{e_t}, h_{t_n^l}\right)/\tau\right)} \tag{3}$$

where $cos(\cdot,\cdot)$ denotes the cosine function to calculate the similarity between entity and relation representations, and τ is a pre-defined hyper-parameter.

3.3 Optimization of Model Pre-training

For model training optimization, we first give the total loss function for pre-training CKBERT based on our two novel pre-training tasks as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{mlm} + \mathcal{L}_{cl} \tag{4}$$

[4]If the threshold for the number of hops δ is too large, the model can easily distinguish the positive and negative triples due to the large semantic gaps. For effective contrastive learning, good negative triples should be "hard negatives".

Model	Text Classification						Question Answering			Total
	AFQMC	TNEWS	IFLYTEK	OCNLI	WSC	CSL	CMRC	CHID	C3	Score
BERT	72.73	55.22	59.54	66.53	72.49	81.77	73.40	79.19	57.91	69.72
MacBERT	69.90	**57.93**	60.35	67.43	74.71	82.13	73.55	79.51	**58.89**	70.28
PERT	**73.61**	54.50	57.42	66.70	76.07	**82.77**	**73.80**	80.19	58.03	70.18
ERNIE-Baidu	73.08	56.22	60.11	67.48	75.79	82.14	72.86	80.03	57.63	69.83
Lattice-BERT	72.96	56.14	58.97	67.54	76.10	81.99	73.47	80.24	57.80	70.29
K-BERT	73.15	55.91	60.19	67.83	76.21	82.24	72.74	80.29	57.48	70.35
ERNIE-THU	72.88	56.59	59.33	67.95	75.82	82.35	72.96	80.22	56.30	69.98
CKBERT-base	73.17	56.44	**60.65**	**68.53**	**76.38**	82.63	73.55	**81.69**	57.91	**71.36**
CKBERT-large	**74.75**	55.86	**60.62**	**70.57**	**78.90**	82.30	73.45	**82.34**	58.12	**72.23**
CKBERT-huge	**75.03**	**59.72**	**60.96**	**78.26**	**85.16**	**89.47**	**77.25**	**97.73**	**86.59**	**78.91**
CKBERT-huge (Ensemble)	**77.05**	**61.16**	**61.19**	**82.80**	**87.14**	**94.23**	**80.40**	**97.91**	**87.26**	**81.02**

Table 1: Performance of tasks on the CLUE 1.1 testing sets (%). The "Total Score" is the weighted averaged score of the nine tasks generated by the official website automatically. All the baseline models are base models (with the same or similar parameter size as that of BERT-base).

Here, we pre-train a series of CKBERT models on distributed GPU clusters, with codes in PyTorch. As PyTorch employs eager execution for tensor computation, it lacks graph-based intermediate representations of models, hindering deeper optimization (Paszke et al., 2019).

Inspired by LazyTensor (Suhan et al., 2021) and Pytorch/XLA on cloud TPUs[5], we develop the TorchAccelerator toolkit for Pytorch training acceleration on GPU clusters. Through XLA custom function and code parsing with an abstract syntax tree (AST), we improve the completeness and performance of the transformation from eager execution to graph execution. A computational graph is generated by TorchAccelerator. The operators on the graph will be fused. By fusing operators, the kernel launch overhead can be reduced. Moreover, fewer intermediate results are written to memory thus reducing the memory bandwidth usage. The effectiveness of computation is also improved by multi-stream optimization and asynchronous transmission of tensors. Since the implementation of TorchAccelerator is not our major focus, more details will be presented in our future work.

4 Experiments

We present comprehensive evaluation results of CKBERT. Due to space limitation, the details of data sources, baselines and hyper-parameter settings are shown in Appendices A, B, C.

[5] https://github.com/pytorch/xla

4.1 General Experimental Results

We evaluate CKBERT over a widely-used Chinese benchmark CLUE (Xu et al., 2020) and knowledge-intensive tasks to evaluate the influence of knowledge injection in CKBERT.

4.1.1 Results of CLUE Benchmark

The CLUE benchmark contains nine text classification and question answering tasks. Specifically, the text classification tasks contain various text task types, including the classification of short sentences and long sentence pairs. The results of all tasks are shown in Table 1.

From the results, we have the following observations. (1) The performance of KEPLMs has a large gap over BERT. It indicates that the injection of different knowledge sources enables the models to perform better semantic reasoning compared to pre-training on texts only. (2) The performance of CKBERT is further improved compared to previous strong baseline KEPLMs under the same parameter size in most cases. From this phenomenon, we believe that the heterogeneous knowledge sources injected into the PLMs benefit the model's results. (3) The larger the number of parameters in the model, the more effective the heterogeneous knowledge fusion is for downstream tasks. The huge model of CKBERT (1.3B parameters) outperforms base (110M) by a large margin, which is suitable for applications that require high prediction accuracy. We also build an ensemble of the huge models (denoted as CKBERT-huge (Ensemble)) from different checkpoints. The performance can be further improved by more than 2.0%.

Model	MSRA	Weibo	Onto.	Resu.
BERT	95.20	54.65	81.61	94.86
MacBERT	95.07	54.93	81.96	95.22
PERT	94.99	53.74	81.44	95.10
ERNIE-BD	**95.39**	55.14	81.17	95.13
Lat.-BERT	95.28	54.99	82.01	95.31
K-BERT	94.97	55.21	81.98	94.92
ERNIE-THU	95.25	53.85	82.03	94.89
CKBERT-base	95.35	**55.97**	**82.19**	**95.68**
CKBERT-large	**95.98**	**57.09**	**82.43**	**96.08**
CKBERT-huge	**96.79**	**58.66**	**83.87**	**97.19**

Table 2: Performance of CKBERT and baselines over four public Chinese NER datasets in term of F1 (%).

4.1.2 Results of NER

We further evaluate CKBERT over the four public NER datasets, including MSRA (Levow, 2006), Weibo (Peng and Dredze, 2015), Ontonotes 4.0[6], and Resume (Yang et al., 2017). The detailed statistics including the split sizes of training, development, and testing sets are described in Appendix A. The models are stacked by the CKBERT encoder and a softmax linear layer, whose parameters are initialized randomly. The entities recognized in the samples are labeled by the `B/I/O/S` tags. This transforms the NER task into a 4-class classification task for each token.

Table 2 shows the performance of various models on four NER datasets. It can be seen that KEPLMs outperform vanilla PLMs. In addition, our CKBERT model (base) with linguistic and external knowledge achieves a large gap performance compared to baselines. We believe that heterogeneous knowledge sources play an important role as described in the ablation study (See Section 4.2).

4.2 Ablation Study

In this part, we evaluate the effectiveness of two important model components of CKBERT on representative tasks. Specifically, We introduce several variants of CKBERT removing certain components. CKBERT-LMLM means that we remove the LMLM task and only learns the CMRM task during pre-training. CKBERT-CMRM remove the CRMR task and only perform the LMLM task. We also provide the results of continual pre-training of BERT-base to remove the influence of additional data sources of plain texts. The performance of those variants and CKBERT on the testing sets of these datasets are shown in Table 3.

Model	AFQ.	IFLY.	CMRC	Weibo
BERT-large-con.	73.96	60.35	73.42	56.12
CKBERT-large	**74.75**	**60.62**	**73.45**	**57.09**
w/o. LMLM	73.56	60.38	73.2	56.48
w/o. CMRM	72.96	59.38	74.8	56.72

Table 3: The performance of models for ablation study. "AFQ." and "IFLY." refer to AFQMC and IFLYTEX, respectively (%).

From the results, we can see that (1) Comparing CKBERT-large to BERT-large (continual pre-trained with the same pre-training data), the explicit heterogeneous knowledge is more useful than the implicit text corpus for various downstream tasks. (2) We also find that the LMLM pre-training task benefits the QA and NER tasks more, whereas the CMRM task improves the performance of plain NLU task (i.e., text classification) significantly. We conjecture that the main reason behind this phenomenon is that the external background knowledge can easily boost the performance due to the shallow semantics of these simple tasks (Yang et al., 2021).

4.3 Results of TorchAccelerator

We investigate to what extent the pre-training speed is improved when our framework is integrated with TorchAccelerator. Figure 2 shows the comparison results between TorchAccelerator and Torch Native with AMP (Automatic Mixed Precision) [7]. The metric "samples/s" means how many samples are computed by the model in each second. Note that we increase the batch size as large as possible to increase the GPUs' memory utilization and occupancy to 100%, and thus the experiments w/ and w/o. TorchAccelerator consume the same amount of computational resources. The underlying GPU is Tesla V100 32GB.

From the results, our observations are as follows. (1) When we only use TorchAccelerator without AMP, the training speed increases slightly. (2) The training speed can have a large improvement with the interaction between TorchAccelerator and AMP (+40%). This is because the kernel fusion of XLA in TorchAccelerator largely reduces the amount of memory access operations, which are the performance bottleneck when AMP is applied. Hence, our TorchAccelerator effectively reduces the consumption of resources and time during pre-training.

[6] https://catalog.ldc.upenn.edu/LDC2013T19

[7] https://github.com/NVIDIA/apex

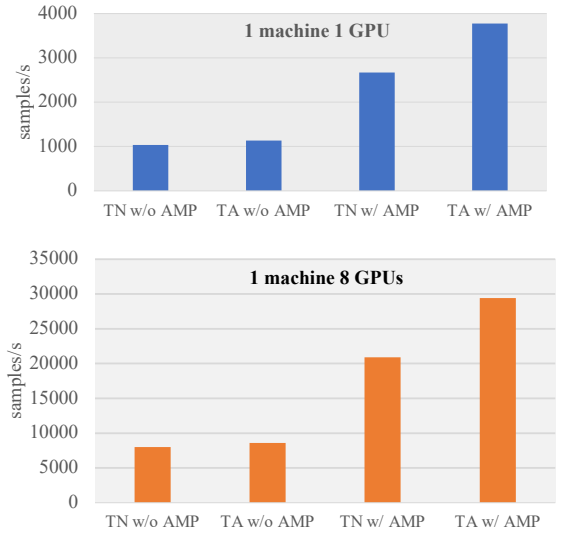

Figure 2: Training speed comparison between TorchAccelerator (TA) and Torch Native (TN).

5 Conclusion and Future Work

In this paper, we propose a novel series of Chinese KEPLMs named CKBERT to inject the heterogeneous sources including linguistic and external knowledge into the PLMs. Specifically, we design two novel pre-training tasks including linguistic-aware MLM and contrastive multi-hop relation modeling, and accelerate model pre-training by TorchAccelerator. The experiments show that our CKBERT outperforms various strong baselines including general PLMs and KEPLMs significantly over knowledge-intensive and natural language understanding tasks. Future work includes (1) integrating more knowledge sources into PLMs to further improve the performance of downstream tasks; (2) exploring heterogeneous knowledge injection to generative KEPLMs and other languages; and (3) enriching the functionalities of TorchAccelerator and releasing it to public.

Ethical Considerations

Our contribution in this work is fully methodological, namely a novel series of KEPLMs, achieving the performance improvement of downstream tasks with different parameter sizes. Hence, there is no explicit negative social influences in this work. However, transformer-based models may have some negative impacts, such as gender and social bias. Our work would unavoidably suffer from these issues. We suggest that users should carefully address potential risks when the CKBERT models are deployed online.

Acknowledgments

This work has been supported by Alibaba Group through Alibaba Innovative Research Program and Alibaba Research Intern Program.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Jacob Buckman and Graham Neubig. 2018. Neural lattice language models. *Trans. Assoc. Comput. Linguistics*, 6:529–541.

Tyler A. Chang, Yifan Xu, Weijian Xu, and Zhuowen Tu. 2021. Convolutions and self-attention: Re-interpreting relative positions in pre-trained language models. In *ACL*, pages 4322–4333.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of bert's attention. In *ACL*, pages 276–286.

Pedro Colon-Hernandez, Catherine Havasi, Jason B. Alonso, Matthew Huggins, and Cynthia Breazeal. 2021. Combining pre-trained language models and structured knowledge. *CoRR*, abs/2101.12294.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In *EMNLP*, pages 3546–3551.

Leyang Cui, Sijie Cheng, Yu Wu, and Yue Zhang. 2021. On commonsense cues in BERT for solving commonsense tasks. In *ACL*, pages 683–693.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 657–668. Association for Computational Linguistics.

Yiming Cui, Ziqing Yang, and Ting Liu. 2022. PERT: pre-training BERT with permuted language model. *CoRR*, abs/2203.06906.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

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.

Zheng Gong, Kun Zhou, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2022. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In *ACL*, pages 5923–5933.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *AAAI*, pages 12963–12971.

Yun He, Ziwei Zhu, Yin Zhang, Qin Chen, and James Caverlee. 2020. Infusing disease knowledge into BERT for health question answering, medical inference and disease name recognition. In *EMNLP*, pages 4604–4614.

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

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2021. Lattice-bert: Leveraging multi-granularity representations in chinese pre-trained language models. In *NAACL*, pages 1716–1731.

Hyunjae Lee, Jaewoong Yoon, Bonggyu Hwang, Seongho Joe, Seungjai Min, and Youngjune Gwon. 2020. Korealbert: Pretraining a lite BERT model for korean language understanding. In *ICPR*, pages 5551–5557.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *SIGHAN@COLING/ACL*, pages 108–117.

Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. 2021. How is BERT surprised? layer-wise detection of linguistic anomalies. In *ACL*, pages 4215–4228.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *ACL*, pages 4487–4496.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *EMNLP*, pages 548–554.

Juan Manuel Pérez, Damián Ariel Furman, Laura Alonso Alemany, and Franco Luque. 2021. Robertuito: a pre-trained language model for social media text in spanish. *CoRR*, abs/2111.09453.

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*, pages 43–54.

Fabio Petroni, Patrick S. H. Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *AKBC*.

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*, pages 2463–2473.

Yusheng Su, Xu Han, Zhengyan Zhang, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. 2021. Cokebert: Contextual knowledge selection and embedding towards enhanced pre-trained language models. *AI Open*, 2:127–134.

Alex Suhan, Davide Libenzi, Ailing Zhang, Parker Schuh, Brennan Saeta, Jie Young Sohn, and Denys Shabalin. 2021. Lazytensor: combining eager execution with domain-specific compilers. *CoRR*, abs/2102.13267.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. In *COLING*, pages 3660–3670.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

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

Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. Easynlp: A comprehensive and easy-to-use toolkit for natural language processing. *CoRR*, abs/2205.00258.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A chinese language understanding evaluation benchmark. In *COLING*, pages 4762–4772.

Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. 2021. Syntax-enhanced pre-trained model. In *ACL*, pages 5412–5422.

Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. 2021. A survey of knowledge enhanced pre-trained models. *CoRR*, abs/2110.00269.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. In *RANLP*, pages 784–792.

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

Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. 2022. DKPLM: decomposable knowledge-enhanced pre-trained language model for natural language understanding. In *AAAI*, pages 11703–11711.

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.

Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2021. CPM: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT : Linguistics informed multi-task BERT. In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 4450–4461.

A Data Statistics

A.1 Data Sources for Pre-training

The pre-training corpora after pre-processing contain 5 million text segments with 623,366,851 tokens (6.2 GB). We also perform simple data preprocessing on the these corpora to improve the quality of the data, including removing incorrect characters and non-Chinese characters, etc. Our KG data is downloaded from the largest authoritative Chinese KG website OpenKG [8]. The number of entities and triples of OpenKG are 16,474,936 and 140,883,574, respectively. The total number of relation types is 480,882.

A.2 Statistics of Downstream Tasks

In this paper, we choose three types downstream tasks for evaluation, including Text Classification (TC), Question Answering (QA) and Named Entity

[8] http://openkg.cn/

Dataset	# Train	# Dev	# Test	Task	Metric
AFQMC	34,334	4,316	3,861	TC	Acc@1
TNEWS	53,360	10,000	10,000	TC	Acc@1
IFLYTEK	12,133	2,599	2,600	TC	Acc@1
OCNLI	50,000	3,000	3,000	TC	Acc@1
WSC	1,244	304	2573	TC	Acc@1
CSL	20,000	3,000	3,000	TC	Acc@1
CMRC	10,142	1,002	3,219	QA	F1
CHID	84,709	3,218	3,231	QA	Acc@1
C3	11,869	3,816	3,892	QA	Acc@1
MSRA	40,000	6,675	4364	NER	F1
Resume	3,821	462	476	NER	F1
Weibo	1,350	264	262	NER	F1
Ontonotes	15,740	4300	4345	NER	F1

Table 4: The data statistics and evaluation metrics used in the experiments.

Model	$n_{param.}$	N_{layer}	N_{head}	D_{head}	D_{ff}	D_{model}
CKBERT-base	110M	12	12	64	3072	768
CKBERT-large	345M	24	16	64	4096	1024
CKBERT-huge	1.3B	24	8	256	8192	2048

Table 5: The overview of hyper-parameters settings of our model architectures. $n_{param.}$ means the total parameters of our model. N_{layer} is the number of model layers. N_{head} is the number of attention heads in each layer. D_{head} is the hidden dimension of attention heads. D_{ff} is the intermediate dimension of FFN layers. D_{model} is the output dimension of the model.

Recognition (NER). The statistics of dataset sizes are shown in Table 4. The result metrics used in our models are different among tasks. We use the Acc@1 for text classification, F1 for NER. For QA tasks, since CHID and C3 tasks are multiple choices, we use Acc@1 as the metric for the two tasks and F1 for CMRC.

B Baselines

In this work, we compare CKBERT with general PLMs and KEPLMs with knowledge triples injected, pre-trained on our text corpora:

B.1 General PLMs

We use three strong Chinese BERT-style models as baselines, namely BERT-base (Devlin et al., 2019), MacBERT (Cui et al., 2020) and PERT (Cui et al., 2022). All the model weights are initialized from Cui et al. (2020).

B.2 KEPLMs

We employ three SoTA KEPLMs continually pre-trained on our pre-training corpora as our baseline models, including ERNIE-Baidu (Sun et al., 2019), ERNIE-THU (Zhang et al., 2019) and K-BERT (Liu et al., 2020). For a fair comparison, KEPLMs using other resources rather than the KG triples are excluded in this work. All the baseline KEPLMs are injected by the same KG triples during pre-training.

C Hyper-parameters Settings

C.1 Hyper-parameters of Pre-training

For optimization, we set the learning rate as 5e-5, the max sequence length as 128, and the batch size as 20. The hidden dimension of the text encoder is

Dataset	AFQMC	TNEWS	IFLY.	OCNLI	WSC	CSL	CMRC
BS	4	12	4	32	32	16	16
Epoch	10	10	10	50	50	10	10
LR	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
MSL	256	128	256	128	128	256	512
Dataset	CHID	C3	MSRA	Resume	Weibo	Ontonotes	
BS	4	4	32	32	32	32	
Epoch	15	15	10	10	10	10	
LR	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	
MSL	192	512	128	128	128	128	

Table 6: The important fine-tuning hyper-parameters used in our CKBERT models. "BS", "LR", and "MSL" indicate the batch size, the learning rate and the max sequence length, respectively.

768. The temperature hyper-parameter τ is set to 0.5. The number of negative samples L is 3. During pre-training, all the experiments are conducted on 15 servers, each with 8 Tesla V100 GPUs (32GB).

C.2 Hyper-parameters of Model Architectures

Table 5 shows the hyper-parameters settings of our CKBERT models w.r.t. the model architectures, including base (110M), large (345M) and huge (1.3B).

C.3 Hyper-parameters of Fine-tuning

Table 6 shows the hyper-parameters settings for fine-tuning. For fair comparison, we set a unified set of important hyper-parameters for each task, including the batch size, the learning epoch, the learning rate and the max sequence length.