KGE-BERT: Inspire BERT with Knowledge Graph Embedding for Question Answering

YINGZHUO YU, University of Michigan, Ann Arbor, US
JIAMIN YANG, University of Michigan, Ann Arbor, US

1 INTRODUCTION

The ability to understand the text and answer questions based on that is a fundamental but challenging task for machines in Natural Language Processing (NLP) field. Instead of selecting the answers from a list of given choices, the system should locate the answer in the paragraph, which requires a deep understanding towards the given text. It has many applications such as Web search and dialogue systems. Solving this problem is expected to help lay the foundation for comprehension ability of machines.

The question-answering task is formulated as follows: Given a context and several questions based on it, the model is asked to predicted whether a question has an answer or not, and the specific location of an existing answer in context. In this project, we aim to design and implement a new NLP model called the Knowledge Graph Embedding BERT (KGE-BERT) that incorporates the knowledge graph learnt from contexts into the BERT model to approach this problem. The challenge of the task itself lies in the complexity of language where the way of formulating sentences can be diverse. For our implementation particularly, it met with the difficulty of finding an effective way of introducing the graph structure into BERT model. We both work on the design of the model, and in terms of implementation, Yingzhuo Yu focused on realizing KGE-BERT and fine-tuning the model, whereas Jiamin Yang worked on setting baseline with different pre-trained BERT models.

2 RELATED WORK

With the development of large pretrained language models, great progress has been achieved in machine reading comprehension. BERT [1] that jointly conditions context from right and left sides to pretrain deep bidirectional representations shows that it can be finetuned to create state-of-the-art models for question answering. However, in the paper of SQuAD2.0[9], it demonstrates relatively low performance of existing models compared with human accuracy and thus, encourages model improvement on this challenging comprehension task.

In recent years, embedding knowledge graphs into pre-trained models becomes a popular research direction and has proven to have the power of boosting model performance. ERNIE [13] was developed that incorporates lexical, syntactic, and knowledge information simultaneously into language model using token encoder and knowledge encoder. The former learns lexical and syntactic features through multi-head attention and the latter fuses tokens and their corresponding entities to extract knowledge features. BERT-MK [4] is built upon ERINE and further looks into the way of learning representations of a graph. For one entity, its near in and out entities are utilized to form triples and converted into a graph representation. The graph is then represented by nodes of entities and relationships to feed into the transformer and perform the task of reconstructing those triples in the aim of learning embeddings. Knowledge-Infused BERT [3] outperforms BERT-large in language and domain understanding by infusing knowledge context from ConceptNet and WordNet to BERT. In a book classification task, knowledge graph containing author

Authors' addresses: Yingzhuo Yu, University of Michigan, Ann Arbor, US, yyzjason@umich.com; Jiamin Yang, University of Michigan, Ann Arbor, US, jiaminy@umich.edu.

relationships are utilized to get author embeddings. For one document, those author embeddings as well as some metadata that include extra information on document are concatenated to the raw output of BERT model [7].
Inspired by the previous work and the observation that a lot of questions are asking about the relationship between entities, we refer to the idea of introducing the knowledge graph to improve the performance of BERT on question answering task. However, instead of using the knowledge context from the existing large knowledge graph, we will build the knowledge graph from the given context to get the corresponding knowledge embedding as the additional features and apply them for question answering task. The detailed model architecture design will be discussed in the later Methodology section.

3 DATA PREPROCESSING

3.1 Dataset

We use SQuAD2.0 for this question-answering task. In this dataset, context are given followed by several questions. For each of the question, it may or may not contain an answer. For a question that has an answer, several possible valid ones are given along with their start and end position in the context. The following Table 1 presents a summary of this dataset[9].

Train	
Total examples	130,319
Proportion of questions that has answer	0.67
Average number of tokens for each answer	4.23
Average number of tokens for each question	12.13
Average number of tokens for each context	151.62
Development	
Total examples	11,873
Proportion of questions that has answer	0.50
Average number of tokens for each answer	4.08
Average number of tokens for each question	12.31
Average number of tokens for each context	163.54

Table 1. SQuAD2.0 statistics.

3.2 BERT Input

We use the BERT tokenizer to process the context and the questions to get input_ids, attention_mask and token_type_ids as the input of BERT Layer. The tokenizer transforms a pair of question and paragraph in the following way.

$$[CLS], q_1, q_2, ..., q_n, [SEP], c_1, c_2, ..., c_m$$

And then, convert those tokens into the index as input_ids. The token_type_ids is used to let BERT distinguish context and question. Hence, in token_type_ids, all the tokens of question including [CLS] and [SEP] are represented as 0 while all the tokens of paragraph context are represented as 1. Since the sequence of our input may exceed the maximum sequence input length of BERT, we set $stride = 128$ and apply sliding window to split the context into several parts. Correspondingly, we rearrange the start position of the answer in dataset. For those input whose length is less than maximum length, we apply padding and set corresponding ids in attention_mask as 0.

4 METHODOLOGY

4.1 Model Architecture

The final design for our model architecture is shown in Figure 2. For the BERT[1] Embeddding Layer, we feed into one paragraph context together with one question as normal. The preprocessing part of the input context and question has been mentioned in the data preprocessing part above.

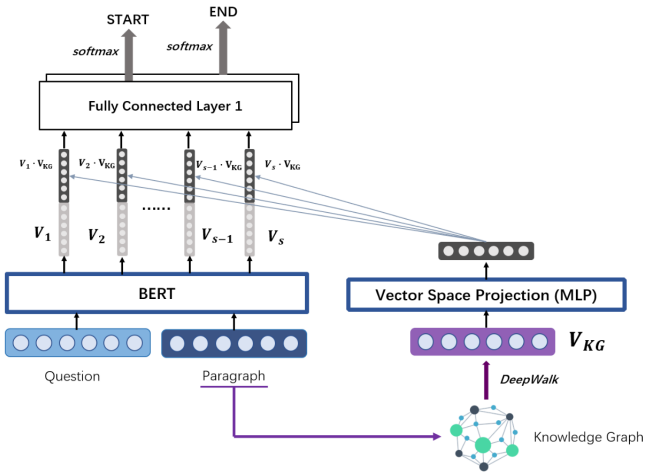

Fig. 1. Knowledge Graph Embedded BERT, Designed Model for Question Answering.

For the knowledge graph embedding part, we firstly extract the entity pairs from the paragraph context through dependency parsing using spaCy[2] to construct the knowledge graph. To get the knowledge graph embedding which shows the relationship between entity nodes, we apply DeepWalk[8]. It uses local information obtained from random walks to learn latent representations for nodes in the graph by treating walks as the equivalent of sentences. We set the dimension of the knowledge graph embedding as 300. For each paragraph, we only select five entities with their knowledge graph embeddings. The selection of those five entities are based on their relevance with the question. We use gensim[11]:'glove-wiki-gigaword-300' to get the word embedding for each entity. As for the question, we average the embedding vector of each word in the question to get the sentence embedding. Then, the entities are ranked by their cosine similarity with that sentence embedding of the question. Getting the five entities, we concatenate their knowledge graph embeddings into one vector of size 5×300. However, we can not just directly inject this vector since it's actually the space representation of entities and thus, its vector space is heterogeneous with the vector space of the BERT model output. Instead, we feed the knowledge graph embedding into vector space projection module, the fully connected layer before injecting it to the BERT output.

To help the system predict the answer from the paragraph, we want to inject the information of entities relationship which is relevant with the question in the paragraph context. Thus, we make dot product between the knowledge graph embedding vector and each output vector from BERT Layer. Then we concatenated the result with each original output vector.

$$\left.\begin{array}{r} BERT\ output:\ v_1, v_2, ..., v_n \\ Knowledge\ Graph\ Embedding:\ V_{KG} \end{array}\right\} \longrightarrow [v_1, v_1 \cdot V_{KG}], [v_2, v_2 \cdot V_{KG}], ..., [v_n, v_n \cdot V_{KG}]$$

Finally, we feed the output vectors above into a fully connected layer to get two scores for each output. One for start position and another represents the end position. Softmax function will be applied respectively to determine the answer in the paragraph. After choosing the position with largest probability as the start of the answer, we choose the position after that with largest probability as the end of the answer. If both start and end position are zero, it indicates that there is no answer to the question.

5 EXPERIMENTS

5.1 Model Implementation

We implemented the KGE-BERT model described above in PyTorch. Our model was trained on Training Data and evaluated on Validation Data of SQuAD2.0[9]. The relevant hyperparameters are listed in the following Table 2.

Parameter	Value
loss function	CrossEntropyLoss
optimizer	AdamW
batch size	10
learning rate	2e-5
weight decay	0.02
max seq length	384
doc stride	128
number of epochs	10
BERT Model	bert-base-uncased

Table 2. Hyperparameters table.

The BERT model bert-base-uncased was accessed by the Python package transformers. All training process was run on a NVIDIA RTX A4000 GPU and it totally took 576 minutes for the whole training process.

5.2 Evaluation Metric

Three metrics are applied to evaluate our model performance including Exact Match (EM), Macro-averaged F1-score and Answer vs No Answer (AvNA)[10]. (1) EM calculates the proportion of predicted answers that is exactly the same as the ground truth answer. (2) F1-score measures the maximum overlap of the predicted answer and each of the ground truth for one question and take the average across all questions. (3) AvNA measures the accuracy of predicting hasAnswer or hasNoAnswer.

5.3 Results

Our baseline model for comparison is BERT based model (uncased) directly followed by two fully connected layers with softmax to getting start and end index of the answer. In the histogram below, we compare the performance of our model KGE-BERT with the baseline model. According to the histogram, our model greatly outperforms the baseline

result in all three evaluation metrics. We can conclude that infusing Knowledge Graph Embedding of entities for BERT helps improve the question answering performance for language models.

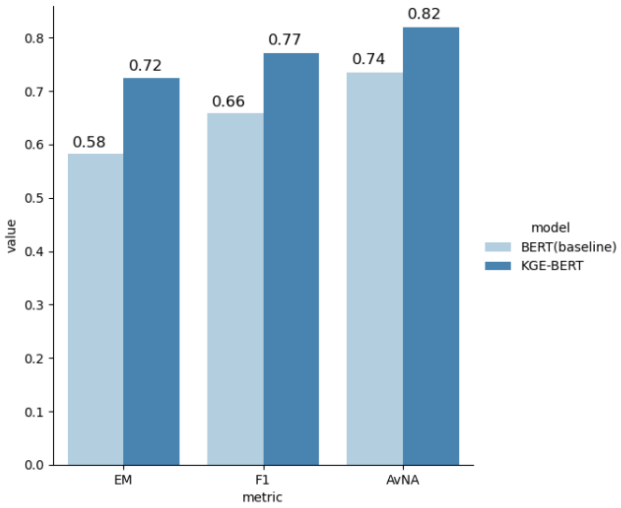

Fig. 2. Model performance comparison result.

6 CONCLUSIONS AND IMPROVEMENTS

6.1 Conclusion

In this project, we designed and implemented a NLP model that combined the knowledge graph embedding with the BERT output to enhance its performance on SQuAD2.0[10]. According to the comparison result, we can conclude that infusing Knowledge Graph Embedding of entities for BERT helps improve the question answering performance for language models. That improvement can caused by the fact that the questions in dataset are mainly regarding the relationships between entities in the corresponding paragraph and therefore, explicitly injecting the knowledge graph embedding to BERT helps further utilize the information of entities relationship.

6.2 Future Improvements

6.3 Experiments on Large Pretrained Language Model

Due to the limit of time and computational resources, we only fine-tuned our model based on Bert-base-uncased. However, we can try other large pretrained language models including Bert-large-uncased, DistilBERT[12], RoBERTa[5] and compare the performance results.

6.4 Knowledge Graph Embedding

In our project, we used DeepWalk[8] to get the embeddings of each entity from knowledge graph. However, since it only learns latent representations for nodes in the graph by random walks, the information of relationships words between each two entities fails to be caught in the knowledge graph embeddings. Thus, in the future work, we can apply other graph embedding techniques that can also take relationships words into considerations such as ConvKB[6]

which uses a convolutional neural network to capture the global relationships and transitional characteristics between entities and relations in knowledge graph.

Besides, when we checked the knowledge graph we built in this project, we found that in many cases, the entities we extract from the context are "it", which makes it hard to build relationships with what is referred to. Thus, in future work, we can apply coreference resolution model to replace those ambiguous entities in knowledge graph.

7 GITHUB CODE LINK

All relevant codes have been uploaded on GitHub with link:
KGE-BERT: https://github.com/Jason-csc/KGE-BERT.
BERT baseline: https://github.com/jessicayjm/kge-bert-base.git.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[2] Explosion. 2022. *spaCy · Industrial-strength NLP.* Retrieved April 1, 2022 from http://spacy.io

[3] Keyur Faldu, Amit P. Sheth, Prashant Kikani, and Hemang Akabari. 2021. KI-BERT: Infusing Knowledge Context for Better Language and Domain Understanding. *CoRR* abs/2104.08145 (2021). arXiv:2104.08145 https://arxiv.org/abs/2104.08145

[4] 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. *EMNLP* (2019). https://arxiv.org/abs/1912.00147

[5] 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. https://doi.org/10.48550/ARXIV.1907.11692

[6] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers).* Association for Computational Linguistics. https://doi.org/10.18653/v1/n18-2053

[7] Malte Ostendorff, Peter Bourgonje, Maria Berger, Julián Moreno Schneider, Georg Rehm, and Bela Gipp. 2019. Enriching BERT with Knowledge Graph Embeddings for Document Classification. *CoRR* abs/1909.08402 (2019). arXiv:1909.08402 http://arxiv.org/abs/1909.08402

[8] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14).* ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[9] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *CoRR* abs/1806.03822 (2018). arXiv:1806.03822 http://arxiv.org/abs/1806.03822

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuad: 100,000+ questions for machine comprehension of text. *EMNLP* (2016). arXiv:1606.05250 [cs.CL] https://arxiv.org/abs/1606.05250

[11] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.* ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[13] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. *ACL* (2019). https://arxiv.org/abs/1905.07129