BLCU-NLP at COIN - Shared Task: Stagewise Fine-tuning BERT for Commonsense Inference in Everyday Narrations

Chunhua Liu [*] **Shike Wang** [*] **Bohan Li** [*] **Dong Yu** [†*✉]

[*] Beijing Language and Culture University
[†] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences

{chunhualiu596, shikewang98, bohanli.lavida}@gmail.com
yudong_blcu@126.com

Abstract

This paper describes our system for COIN Shared Task 1: Commonsense Inference in Everyday Narrations. To inject more external knowledge to better reason over the narrative passage, question and answer, the system adopts a stagewise fine-tuning method based on pre-trained BERT model. More specifically, the first stage is to fine-tune on additional machine reading comprehension dataset to learn more commonsense knowledge. The second stage is to fine-tune on target-task (MCScript2.0) with MCScript (2018) dataset assisted. Experimental results show that our system achieves significant improvements over the baseline systems with 84.2% accuracy on the official test dataset.

1 Introduction

The COIN Shared Task1 aims to evaluate a system's commonsense inference ability in everyday narrations by selecting an appropriate answer from two candidates for each question, which also can be seen as a multiple-choice reading comprehension (MCRC) task. The most difficult part of this task lies in about 50% of the questions cannot be answered directly from the passage, because commonsense knowledge required to answer questions is not missing. Commonsense knowledge is essential but challenging to acquire and represent because of its invisible and implicit proprieties. Accordingly, the key solution to this problem is how to introduce world knowledge contained in additional databases or datasets into the system.

Neural networks have gained amazing results in various machine reading comprehension tasks. Typical strategy adapts neural encoder such as LSTM (Long Short-Term Memory) (Hochreiter and Schmidhuber, 1997) or CNN (Convolutional Neural Network)(LeCun and Bengio, 1998) to encode a passage, a question and a candidate answer separately and then employs attention mechanism to model interactions among them. This kind of method performs well on questions that can be answered from given passage texts but shows limited performance on questions demanding external knowledge to answer. Recently, the approach of the pre-training language model on large-scale free-texts to acquire external knowledge and then transferring learned background knowledge to a downstream task has displayed promising improvements in a variety of natural language processing tasks.

Compared with existing pre-trained language models, like ELMo (Peters et al., 2018) and GPT (Alec Radford, 2018), BERT stands out in language representation and understanding by introducing masked language model and next sentence prediction task. Hence, we choose BERT as the basic model to explore how much a pre-trained language model can help to solve the commonsense inference problem. In this process, two primary questions are guiding this work:

- How much gains can a pre-trained language model bring for the commonsense inference task?
- How to add more commonsense knowledge to a pre-trained language model to assist commonsense inference?

For the first question, we designed several groups of experiments to compare the performance of BERT$_{base}$ and BERT$_{large}$ on three types of questions provided in the target task, including text-based, script-based (also called commonsense-based), and text-or-script.

For another question, we present a two-staged fine-tuning approach to add more commonsense knowledge to the model. This first stage is to fine-tune on pre-trained encoder with additional corpus beyond English Wikipedia and BooksCorpus Some other genre corpus, like news-wire texts, English examine texts, and everyday narrations are considered in this phase. This way empowers the encoder to learn and store more knowledge about the world and thus improve its commonsense inference ability. The second stage is to fine-tune the updated encoder with a top classification layer on target-task with the support of additional commonsense datasets.

2 BERT for COIN-Everyday Narrations

2.1 How to Fine-tune BERT?

BERT is a bidirectional transformer encoder trained on the task of masked language model and next sentence prediction, equipping with powerful language encoding capacity. Devlin et al. (2018) provides two pre-trained model sizes: BERT$_{base}$ and BERT$_{large}$ with the different parameters, such as layers {12, 24}, self-attention heads {12, 16}, and hidden size {768, 1024}. BERT can encode any sequence less than 512 tokens, like a sentence or a paragraph. Generally, the final hidden outputs of the first token [CLS] is considered as the overall representation of the whole input sequence.

When applying BERT to MCRC task, the input token sequence is the concatenation of each candidate answer with the corresponding question and passage in the following format:
[CLS] Passage [SEP] Question Candidate [SEP]. So, the final hidden state of [CLS] represents the comprehensive understanding of the passage, question and candidate answer. Additionally, a classification layer is required to stack on the top of the BERT model to score for each candidate answer. The candidate who has the highest score would be regarded as the correct answer. When fine-tuning, the weight of both the BERT and classification layer are modified to adapt to the target task with the goal of minimizing the cross-entropy loss.

2.2 How powerful is BERT?

In this part, BERT$_{base}$ and BERT$_{large}$ model are fine-tuned as above described. Figure 1 show the gap between baseline model Attentive Reader (Hermann et al., 2015) re-implemented by Ostermann et al. (2019) and two pre-trained BERT models. The pre-trained models brings about 12.2% to 15.6% improvements, which obviously outperform the baseline system.

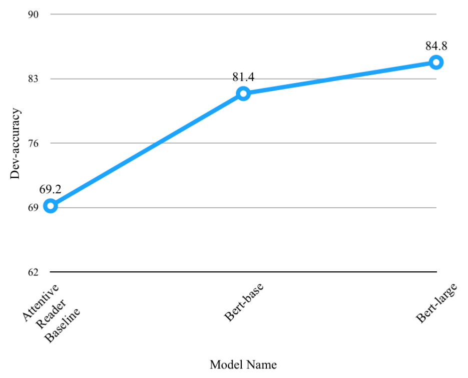

Figure 1: Comparing pre-trained models with baseline model.

Moreover, in order to compare the difference between BERT$_{base}$ and BERT$_{large}$, we give the statistics of their performance on three question labels. Figure 2 illustrates that 1) Both models are good at answering questions whose answer are given in the corresponding passages. 2) Even text-script questions can be answered based on either given passages or external commonsense, it's still hard for BERT$_{b}ase$ model to answer. 3) Compared with BERT$_{base}$, BERT$_{large}$ shows significant improvements in both script-based questions and text-script questions. These observations reveal that training more texts with a larger model is more likely to learn more commonsense knowledge. So, we use BERT$_{large}$ model in the following experiments.

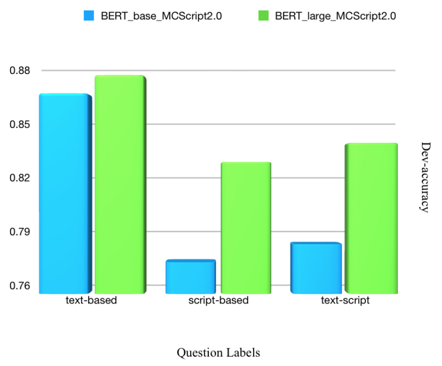

Figure 2: Comparing two BERT models on three question labels.

3 Stagewise Fine-tuning BERT

The overview of stage-wise fine-tuning BERT presented in this work is shown in Figure 3. It consists of two phases: encoder fine-tuning stage, classifier and encoder fine-tuning stage.

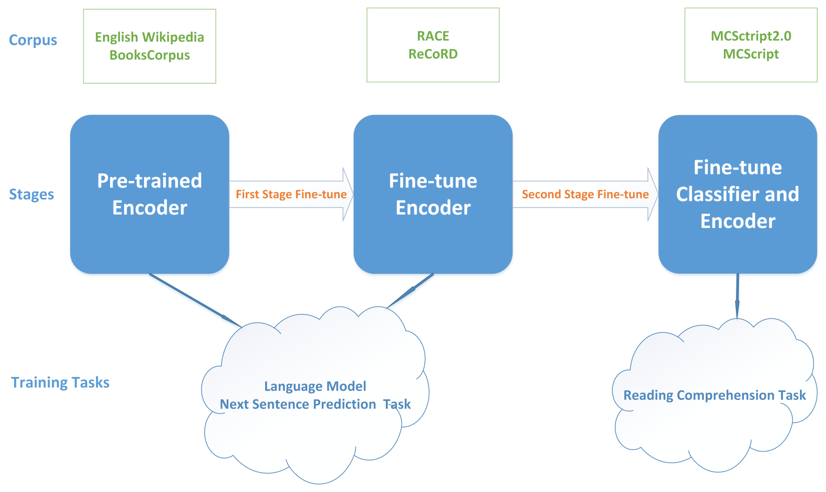

Figure 3: A view of two fine-tuning stages.

Dataset Name	Content	#Passages
RACE (Lai et al., 2017)	Mid/High Exam	25K
ReCoRD (Zhang et al., 2018)	News articles	80K
ROCStories (Mostafazadeh et al., 2016)	Narrations	3K (dev + test)
MCScript (Ostermann et al., 2018)	Narrations	2.1K
MCScript2.0 (Ostermann et al., 2019)	Narrations	3.5K
Inscript (Modi et al., 2017)	Narrations	1K
DES (Wanzare et al., 2019)	Narrations	0.5K

Table 1: Datasets used in this paper for fine-tuning.

3.1 Encoder Fine-tuning

In this work, we denote the pre-trained BERT$_{large}$ model including the embedding part as well as the 12 layers of transformer blocks as a whole and name them as the encoder.

The encoder is responsible for sequence representation by transforming raw input tokens into a fixed representation. Weights in the encoder decide how one token is represented and how one token in a sequence interacts with another. Therefore, allowing the encoder to witness more and train longer can enhance its representation ability and thus able to encode new input with a wide range of structures, writing styles and expressions.

Hence, in this phase, the key lies in how to find more applicable data to train the encoder further, also called fine-tune the encoder based on the pre-trained BERT$_{large}$. When choosing new data, we take two aspects, the task form pertaining to the MCRC and the content relating to everyday narrations into consideration. In addition, based on the number of datasets used for training, the encoder fine-tuning can be classified into two categories: single-dataset fine-tuning and multi-datasets fine-tuning. The former means to fine-tune the encoder using only a single dataset. The latter fine-tunes the encoder by taking data from multiple datasets as input. Table 1 lists datasets selected in this paper.

For most datasets, only paragraphs or passages are used as training data, whose questions and answers are ignored. But to add more question-answering information, we add some questions and their answers to the passages, which is motivated by the task of the next sentence prediction. For dataset like RACE, whose questions and candidate answer is free-text, we randomly pick a question for the passage as well as its answer and then append it to the end of the passage. In this way, the question is treated as the next sentence for the final sentence of the passage, and similarly, the answer can be seen as the next sentence for the question. This technique makes it possible for the encoder to learn the potential questioning and its answer.

When fine-tuning, the model is still jointly trained on the task of masked language model and next sentence prediction. All weights are modified

Datasets for first stage	Datasets for second stage	Dev-acc(%)	Test-acc(%)
-	MCScript2.0	84.8	-
MCScript2.0		84.8	
MCScript		83.6	
RACE		84.9	
ReCoRD		<u>85.9</u>	
ROCStories		83.5	
Inscript		83.6	
DES		83.9	
RACE	MCScript2.0 + MCScript	85.1	-
RACE	MCScript2.0 + MCScript-w/o-who-how	85.7	-
RACE+ReCoRD	MCScript2.0	85.0	-
RACE+ReCoRD	MCScript2.0 + MCScript	<u>86.0</u>	-
RACE+ReCoRD	MCScript2.0 + MCScript-w/o-who-how	**86.6**	**84.2**
RACE+ReCoRD	MCScript2.0 + SWAG	85.9	-
All Datasets	MCScript2.0 + MCScript-w/o-who-how	84.7	-

Table 2: Main results.

when fine-tuned.

3.2 Classifier and Encoder Fine-tuning

The fist stage fine-tuning endows the encoder with more new knowledge, while the second stage fine-tuning focuses on adjusting weights in the encoder to adapt to the target task. This phase is carried on the fine-tuned encoder with the support of additional commonsense datasets, like MCScript (Ostermann et al., 2018), SWAG (Zellers et al., 2018). The most benefits come from the MCScript, which is the dataset used for evaluation of SemEval 2018 Task 11. When doing experiments, an interesting discovery is found that using the entire MCScript is not the best choice. Filtering some types of questions leads to better results on the development set of MCScript2.0. During fine-tuning, a classification layer is also added on the top of BERT and the training is guided by minimizing the cross-entropy loss.

4 Experiments and Results

4.1 Data

COIN Shared Task 1 uses MCScript2.0 corpus, which consists of three kinds of question labels, including text-based, script-based, and text-script. For text-based question, the answer can be deduced from the information provided in passages, while script-based question can only be answered with the support of external commonsense knowledge. The text-script question can be answered either depends on passages or external script knowledge.

4.2 Experiment Setup

We use the Pytorch version of pre-trained BERT implemented by huggingface[1]. Adam Optimizer (Kingma and Ba, 2014) is used to optimize the model, which is trained on TITAN RTX with 2 GPUs. Important hyper-parameters for training are listed in Table 3.

Description	**first stage**	**second stage**
t: tokens max length	350	300
e: fine-tune epoch	{3,4}	{3,4}
α: learning rate	3e-5	1e-5
b: batch size	32	64
g:gradient accumulation step	4	8

Table 3: Hyper-parameters settings used during training.

4.3 Results and Analysis

Table 2 demonstrates results of various trained models, consisting of three groups. This first group experiment is designed for first stage fin-tune, so only the target dataset, which refers to the MCScript2.0, is used in the second stage. The second group is conducted to help find the most suitable dataset to assist second-stage fine-tuning. The last group is a combination of the previous two groups.

[1]https://github.com/huggingface/pytorch-transformers

By observing the results in the first group, we can see that using the dataset from target task to first fine-tune the encoder didn't bring any improvements, indicating that using the same data to train model twice is unnecessary. Also, instead of raising the accuracy, training with some datasets even damage the model and diminish the accuracy. This can be attributed to many reasons, for example, the passage in ROCStories is too short compared with the target task, the data size of Inscript and DES is too small. However, with the prop of RACE and ReCoRD, the model has secured some advances. So, in the next phase fine-tuning, the two datasets are mixed to fine-tune the encoder in the first stage.

The second group of results points out that removing the question types with who and how in the MCScript can surprisingly increase the accuracy compared with using the whole set of MCScript. This is possibly caused by the different data distribution in the two datasets.

Our final submitted model is first fine-tuned on both RACE and ReCoRD and then fine-tuned with data in MCScript without the question type of who and how in MCScript2.0, which achieves the accuracy of 86.6% and 84.2% on the development set and test set separately and ranks fourth on the final test leaderboard.

5 Conclusion

This paper depicts our system that fine-tunes the pre-trained BERT model with two stages, which outperforms far further than the baseline model and achieves the accuracy of 84.2% in the official test dataset. Experimental results indicate that both stages fine-tuning bring benefits to the model. Besides, experiments reveal that BERT$_{large}$ excels at commonsense inference task.

Acknowledgement

This work is funded by the open project of Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (IIP2019-4).

References

Tim Salimans Ilya Sutskever Alec Radford, Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*.

Yann LeCun and Yoshua Bengio. 1998. Convolutional networks for images, speech, and time series.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2017. Inscript: Narrative texts annotated with script information. *CoRR*, abs/1703.05260.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *ArXiv*, abs/1604.01696.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mcscript: A novel dataset for assessing machine comprehension using script knowledge.

Simon Ostermann, Michael Roth, and Manfred Pinkal. 2019. MCScript2.0: A machine comprehension corpus focused on script events and participants. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 103–117, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Lilian D. A. Wanzare, Michael Roth, and Manfred Pinkal. 2019. Detecting everyday scenarios in narrative texts. *ArXiv*, abs/1906.04102.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

Shenmin Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *ArXiv*, abs/1810.12885.