KDE SenseForce at SemEval-2020 Task 4: Exploiting BERT for Commonsense Validation and Explanation

Khanddorj Mendbayar, Masaki Aono
Department of Computer Science
Toyohashi University of Technology
Toyohashi, Aichi, Japan
khanddorj@kde.cs.tut.ac.jp, aono@tut.jp

Abstract

Using a natural language understanding system for commonsense comprehension is getting increasing attention from researchers. Current multi-purpose state-of-the-art models suffer on commonsense validation and explanation tasks. We have adopted one of the state-of-the-art models and proposing a method to boost the performance of the model in commonsense related tasks.

1 Introduction

Commonsense knowledge is a knowledge that is built by everyday activities and vital for living. Because of the assumption that every person has the commonsense, we tend to discard the commonsense from social communications. Hence, most of the text data lack explicit declaration of commonsense (Liu and Singh, 2004). Nonetheless, basic knowledge of the commonsense related to space, physical interactions, and people is necessary for the development of the natural language processing, computer vision, and robotics (Davis and Marcus, 2015), and has been studied as early as 1956 (Mueller, 2014).

Currently, there are many natural language processing subfields where machine learning has started to outperform humans (Wang et al., 2017). However, even the current state-of-the-art models like BERT (Devlin et al., 2018), Elmo (Peters et al., 2018) fail in commonsense validation and explanation tasks (Wang et al., 2019).

We propose a system to cope with commonsense validation and explanation tasks using input transformation and word feature scaling. Our proposed system extracts features about every word in the sentence from its surrounding contexts except the word itself then compares extracted features with the original word features to get a score map (usually, scaling coefficient is called 'attention ', whereas it is called score map to avoid a confusion). Then the score map is used to scale the word feature in the current state-of-the-art models.

The result is noticeable, considering that the system uses only one additional fully connected layer on top of the existing system. For the Validation (A) and Explanation (B) task, the proposed system outperforms the result of the BERT by 9.5% and 27.2% , respectively. The results are submitted to the SemEval-2020 (Wang et al., 2020) and turned out to be in 33rd and 22nd places respectively. The system has room for improvement, since the tested input transformation works on only word level.

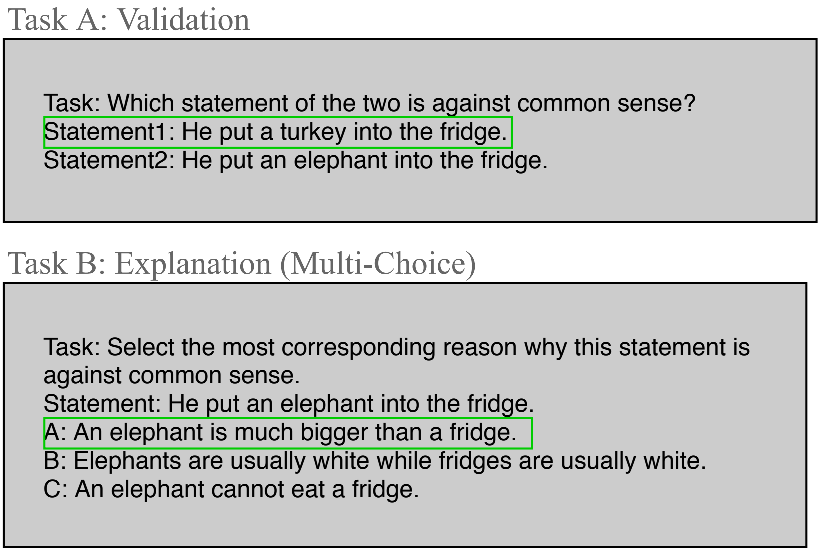
2 Background

Nowadays, Transformer's encoder and decoder model (Vaswani et al., 2017) has been used as the basis for almost all state-of-the-art technologies in natural language processing. A typical example is BERT. BERT is pretrained on a large dataset to predict masked words in a sentence from the context. This pretraining method creates a discrepancy between pretraining and finetuning (Yang et al., 2019). We think this pretraining method is used insufficiently in the commonsense validation and explanation tasks considering the pretraining dataset contains some commonsense already. To use this pretraining method better, we propose a model which uses [MASK] token in the finetuning process.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 551–555
Barcelona, Spain (Online), December 12, 2020.

SemEval-2020 Task 4 ComVE challenge consists of 3 subtasks: Validation (A), Explanation (Multi-Choice) (B) and Explanation (Generation)(C). As the organizers benchmarked, current state-of-the-art models like BERT and Elmo struggle with these subtasks, showing maximum accuracy of 74.1% and 45.6% for the first two subtasks.

For the training of the system, we only used dataset that was provided by task organizers. Examples of the first two subtasks examples are shown in Figure 1. Each subtasks dataset contains around 10,000 examples per subtask. We employed the hugging face libraries(Wolf et al., 2019) pretrained model. We submitted for the subtasks A and B of the competition.



Task A: Validation

Task: Which statement of the two is against common sense?
Statement1: He put a turkey into the fridge.
Statement2: He put an elephant into the fridge.

Task B: Explanation (Multi-Choice)

Task: Select the most corresponding reason why this statement is against common sense.
Statement: He put an elephant into the fridge.
A: An elephant is much bigger than a fridge.
B: Elephants are usually white while fridges are usually white.
C: An elephant cannot eat a fridge.

Figure 1: Sample data from the first two subtasks

3 System overview

Commonsense validation is a difficult task because it can depend on any word in the sentence. By changing a single word or a punctuation, the sentence can gain or loose its commonsense. Thus, we propose a model that consists of two stages: token level commonsense evaluation and word feature scaling.

3.1 Token level commonsense evaluation

As mentioned in the background section, [MASK] token is used in finetuning process of the BERT. It is shown that the output from the final layer of BERT corresponding to the [MASK] token contains features about a word which was masked. Instead of trying to predict the word as in BERT, we compare the last layers output tensor corresponding to the [MASK] token to the original masked word itself's features using fully connected layer. This gives a score of a single word which evaluates how much priority the system should give. In order to generate a score map of the input sentence, every word in the sentence is evaluated one by one, as illustrated in Figure 2.

For the subtask B, only the sentence without the commonsense is processed through the token level common sense evaluation, which only outputs a score map for the against-commonsense-sentence.

$$I_1 = \begin{bmatrix}1\end{bmatrix} I_2 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} I_3 = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \dots I_N = \begin{bmatrix}1 & 0 & 0 & \dots & 0\\ 0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \dots & 1\end{bmatrix} \tag{1}$$

Sigmoid activation function is applied on the score map to keep the numbers in a limit. However, after few trials, we realized that applying sigmoid function led to the degraded values of the word features. Thus, the score map is incremented by one.

Moreover, instead of a loop, the input tokens are multiplied times, where N is the length of the input, making the input N by N tensor. In the dataset we used, the maximum length of the sentence is 26. Then the input tensor is masked with identity matrix in Equation 1 pattern to ensure every word in the sentence is masked one by one.

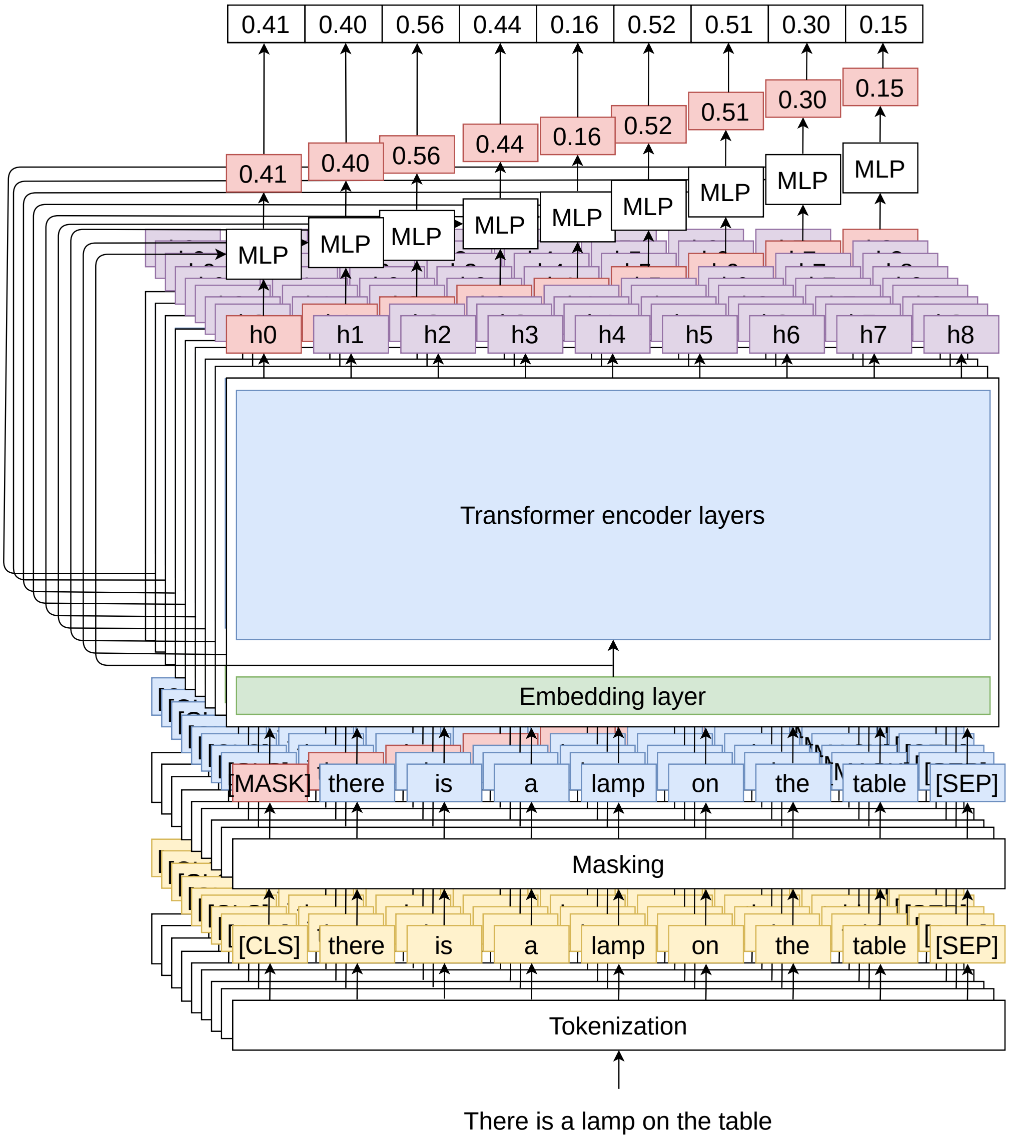


Figure 2: Input sentence is multiplied by the length of the sentence and masked by identity matrix pattern. After that BERT is used for extracting features to get the score map with a fully connected layer.

3.2 Word feature scaling

BERT is consisted of embedding layer and 24 layers of Transformer encoder layers. For the subtask A, we use BERT's embedding layer once again to get the words' features. Then, we multiply the words' features with the score maps that we obtained from the token level commonsense evaluation part. Then, scaled word features are processed through the rest of the BERT's layers to get the final classification and selection.

For the subtask B, before the embedding we concatenate the against-the-commonsense sentence and

[CLS]	he	poured	orange	juice	on	his	cereal	.	[SEP]
1.39	1.45	1.21	1.17	1.16	1.43	1.42	1.15	1.8	1.14

[CLS]	he	poured	milk	on	his	cereal	.	[SEP]
1.4	1.44	1.2	1.11	1.38	1.4	1.15	1.9	1.28

Table 1: An example of a score map for a sentence with commonsense and a sentence without commonsense

	Task A accuracy	Task B accuracy
BERT	70.1%	45.6%
KDE SenseForce	79.2%	72.8%
Best at ComVE	97%	95%

Table 2: Result of the system compared to the BERT and the best result in the ComVE.
Note: the dataset size in our experiment is about five times larger than the reference BERT model.

the options one by one, giving three inputs. Each of them is sent to the BERT embedding layer to get scaled by the score map. However, the length of the score map and the words' features are different, since we only do the token level commonsense evalutaion on the against-the-commonsense. Thus, the score map is padded by 1 on the right side to make them have an equal length.

This step can be varied in many ways because after we obtain the evaluation score map, we can use it in almost any variants of the Transformer based model, improving its accuracy as long as the tokenizer is similar. For our proposed model, we have investigated only one variant of the Transformer.

4 Experimental setup

We only train the fully connected layers as BERT fine tuning. Hence, we trained our system for 4 epochs as stated in the BERT's paper. We used batch size of 1 and learning rate of 5e-3. The reason that we used 1 as our batch size is due to the multiplication of the input sentence in the evaluation stage. The multiplied input is treated as batch in the PyTorch (Paszke et al., 2019). The code is written on Python3.6 (Van Rossum and Drake, 2009) with Pandas (McKinney and others, 2010) for reading the input files, PyTorch for machine learning and Hugging face for using BERT model. As we mentioned earlier, the dataset we used is only from task organizers, so that neither data augmentation nor external resource was used in our experiment.

5 Results

Initially, we expected the important words such as subject, verb and object to have higher scores in the token level evaluation, but the important words had lesser scores than the other words of the sentence. In Table 1, an example of the score map is shown. In addition, our model tends to fail on a sentences with a phrase of a two or more words. We deliberated that this behaviour is due to the token level evaluation.

Our proposed models accuracies on subtask A and B are 79.2% and 72.8%, respectively. Comparisons are presented in Table 2.

6 Conclusion

In this paper, we proposed a system coping with two subtasks of SemEval-2020 Task 4: Commonsense Validation and Explanation. To do that we have adopted one of current state-of-the-art models and enhanced its accuracy notably by only adding one extra fully-connected layer, which makes it memory efficient. The score map generation method, however, turns out to be computationally more expensive. The results are submitted to the SemEval-2020.

The experiment demonstrated that we could use the discrepancy between pretraining and finetuning of BERT that was caused by input corruption for score map generation for the commonsense validation and explanation subtasks.

In the future, we intend to experiment with different variants of Transformers such as GPT-2 (Radford et al., 2019), XLNet and ALBERT (Lan et al., 2019) with similar kind of input transformation and scaling.

7 Acknowledgments

We would like to thank the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan. The publication of this paper the would not be possible without their aid.

References

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Hugo Liu and Push Singh. 2004. Conceptnet - a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Erik T Mueller. 2014. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

W Wang, N Yang, F Wei, B Chang, and M Zhou. 2017. R-net: Machine reading comprehension with self-matching networks. *Natural Lang. Comput. Group, Microsoft Res. Asia, Beijing, China, Tech. Rep*, 5.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. *arXiv preprint arXiv:1906.00363*.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. SemEval-2020 task 4: Commonsense validation and explanation. In *Proceedings of The 14th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.