# DEEPYANG at SemEval-2020 Task 4: Using the Hidden Layer State of BERT model for Differentiating Common Sense

**Yang Bai, Xiaobing Zhou***
School of Information Science and Engineering
Yunnan University, Yunnan, P.R. China

## Abstract

Introducing common sense to natural language understanding systems has received increasing research attention. To facilitate the researches on common sense reasoning, the SemEval-2020 Task 4 Commonsense Validation and Explanation(ComVE) is proposed. We participate in sub-task A and try various methods including traditional machine learning methods, deep learning methods, and also recent pre-trained language models. Finally, we concatenate the original output of BERT and the output vector of BERT hidden layer state to obtain more abundant semantic information features, and obtain competitive results. Our model achieves an accuracy of 0.8510 in the final test data and ranks 25th among all the teams.

## 1 Introduction

Humans can utilize a variety of knowledge and reasoning to help understand the meanings of language. For example, it is easy for a human to know that someone can put a turkey into a refrigerator. Still, he can never put an elephant into it with basic commonsense reasoning, but it can be non-trivial for a system to tell the difference(Wang et al., 2019). What enables us to arrive at this conclusion is the knowledge we have about the world and the underlying reasoning process, often called commonsense thought (Minsky, 2000) or commonsense reasoning(Davis and Marcus, 2015), that allows us to connect pieces of knowledge to reach the new conclusion. We know that the refrigerator is household appliances and turkeys are the food we usually eat, but elephants cannot be eaten; the elephant is larger than the refrigerator and cannot be put in it. While this kind of knowledge and reasoning becomes so naturally to humans, it is extremely difficult for machines. Despite significant advances in natural language processing in the last several decades, machines are still far away from having this type of natural language inference (NLI) ability(Storks et al., 2019). Therefore, we need an effective automatic method to differentiate natural language statements that make sense from those that do not make sense.

In this paper, the used methods and the results obtained by our team, DEEPYANG, on the ComVE shared task organized at SemEval 2020 are presented(Wang et al., 2020). The ComVE shared task includes three sub-tasks Commonsense Validation, Commonsense Explanation (Multi-Choice), and Commonsense Explanation (Generation). We only participate in subtask A. The main goal in sub-task A is to choose which statement of the two is against common sense. By definition, the sub-task A is to select from two natural language statements with similar wordings which one makes sense and which one does not make sense. For this subtask, we try traditional machine learning methods, deep learning methods, and pre-trained models. After comparison, we choose the BERT-base model and concatenate the original output of BERT and the output vector of BERT hidden layer to obtain more abundant semantic information features. Finally, our model achieves an accuracy of 0.8510 on the test set, and ranks 25th.

## 2 Related Work

Recent years, the NLP community have witnessed a variety of works that enhances machines ability to perform deep language understanding which goes between the lines, rather relying on reasoning

*Corresponding author:zhouxb@ynu.edu.com




*Proceedings of the 14th International Workshop on Semantic Evaluation*, pages 516–520
Barcelona, Spain (Online), December 12, 2020.

and knowledge of the world. Many benchmark tasks and datasets have been presented to support the development and evaluation of such natural language inference ability. For example, using traditional machine learning methods mainly includes Support Vector Machines (SVMs)(Hearst et al., 1998), etc. Nowadays, deep neural networks have become mainstream for NLP, such as Convolutional Neural Networks(Kim, 2014) and Long Short Term Memory networks (LSTMs)(Wang et al., 2016)(Wang et al., 2018).

However, in traditional word embedding models such as word2vec(Mikolov et al., 2013), GloVe(Pennington et al., 2014) or FastText(Joulin et al., 2016), the embedding vectors are context-independent. Despite the target word appears in different contexts, after training, the embedding vector remains the same. Consequently, these embeddings lack the capability of modeling different word senses in different contexts, although this phenomenon is prevalent in language.

To address this problem, recent works have developed contextual word representation models, e.g., Embeddings from Language Models (ELMO)(Peters et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT)(Devlin et al., 2018). Based on the context they appeared, these models give words different embedding vectors . These pre-trained word representations can be used as features or fine-tuned for downstream tasks. For example, the Generative Pre-trained Transformer (GPT)(Radford et al., 2018) and BERT introduce minimal task-specific parameters and can be easily fine-tuned on the downstream tasks with modified final layers and loss functions.

## 3 Methodology

In this section, the methods operated by our team for sub-task A are explained. We use several methods including traditional machine learning methods, deep learning methods, and pretrained Language Model, such as SVM, LSTM, Attention, BERT, and so on. As expected, as the out-of-the-box pre-trained and fine-tuned BERT model has shown great performance in many kinds of NLP problems, BERT also has achieved a good result by comparison in this work. Finally, We choose the BERT model as our base model. Next, we introduce the methods we use in this work.

### 3.1 Traditional Machine Learning Methods

Traditional machine learning methods, in particular, SVM is one of the most robust and accurate methods among all well-known data mining algorithms. Therefore, in our experiments, we apply the SVM classifiers. But it doesn't perform well on the validation set.

### 3.2 Word Embedding Model

Representing a word by using a low-dimensional vector is the usual way in natural language processing. The fastText tool is used in our system to get the word representation of the sentences. A low dimensional vector in fastText is associated with each word, and hidden representations can be shared between different classes of classifiers so that textual information can be used together in different classes. We use the pre-trained fastText embedding and try a combination of different models, such as LSTM, CNN, Attention, and other different combinations, but can't get satisfying results.

### 3.3 BERT with Hidden State

Because the training of model in this task must make full use of the whole sentence content to extract useful linguistic, syntactic, and semantic features which may help to make a deeper understanding of the sentences, while at the same time, less subjected by the noisiness of speech. So, we use BERT in subtask-A. Unlike most of the other methods, BERT uses the bidirectional representation to make use of the left and right context to gain a deeper understanding of a sentence by capturing long-range dependencies between each part of the sentence. In the classification task, the original output of BERT(pooler output) is obtained by its last layer hidden state of the first token of the sequence (CLS token) further processed by a linear layer and a *tanh* activation function. But the pooler output is usually not a good summary of the semantic content of the input.[1]

[1] https://huggingface.co/transformers/model_doc/bert.html#bertmodel

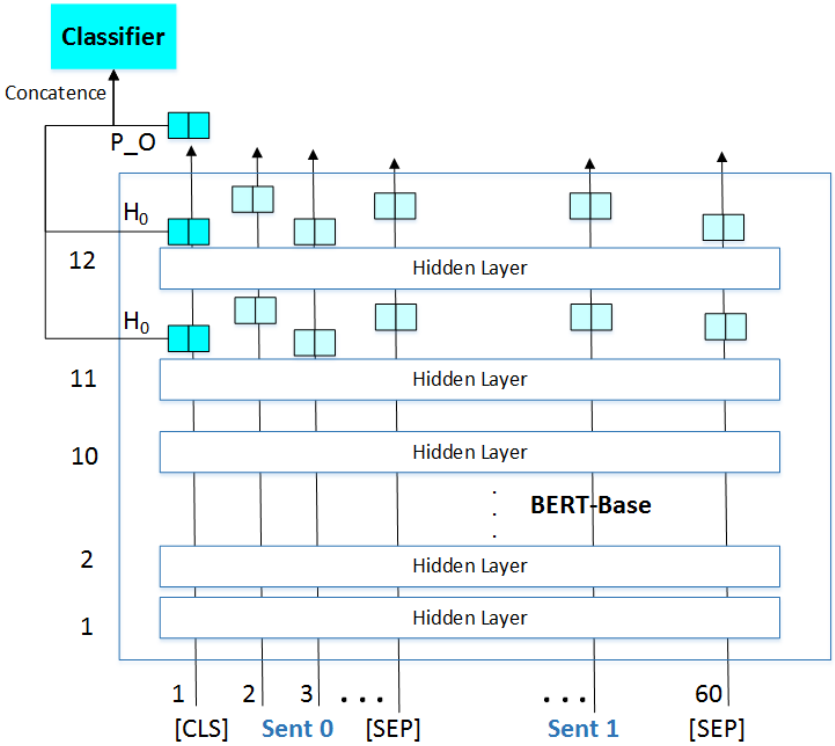


Figure 1: Our model($P\_O$ is the pooler output, $H_0$ is hidden-state of the first token of the sequence(CLS token) at the output of the hidden layer of the model.)

It is very necessary to explore the deep representation learning of Bert, which can help us to understand the limitations of Bert more clearly, so as to improve it. Recent studies have shown that the top hidden layer of BERT can learn more abundant semantic information features.(Jawahar et al., 2019) Therefore, in order to make the model get more abundant semantic information features, we propose the following model which is shown in Figure 1. Firstly we get the pooler output ($P\_O$). Secondly, we take the $H_0$ of the last two hidden layers. Finally, we concatenate $P\_O$ and $H_0$ into the classifier.

## 4 Experiments and Results

### 4.1 Pre-processing

For traditional machine learning methods, in word embedding model, we combine $sent0$ and $sent1$, and then use $< eops >$ as the separator of two sentences in the input data, formed as follows: $sent0$+$< eops >$+$sent1$. BERT uses the wordpiece tool for word segmentation and inserts special separators ($[CLS]$, which are used to separate each sample) and separator ($[SEP]$, which are used to separate different sentences in the sample), formed as follows: $[CLS]$+$sent0$+$[SEP]$+$sent1$+$[SEP]$.

### 4.2 Experimental settings

For the word embedding model, the crawl-300d-2M embedding[2] is 2 million word vectors trained with subword information on Common Crawl (600B tokens). We use crawl-300d-2M as our word embedding and input them to the LSTM and Attention module.

For the BERT, we use BERT-Base[3] pre-trained model, which contains 12 layers. We use stratified 5-fold cross-validation[4] with 42 random seeds for training set, and stratified sampling ensures that the proportion of samples in each category of each fold data set remains unchanged. We use Adam optimizer with a learning rate as 1e-5 and CrossEntropy Loss. In order to save GPU memory, the batch size is set to

[2]https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M.vec.zip
[3]https://huggingface.co/bert-base-cased#
[4]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

4 and the $gradient\_accumulation\_steps$ is set to 4, so that each time a sample is an input, the gradient is accumulated 4 times, and then the back-propagation update parameters are performed. We extract the hidden layer state of BERT by setting the $output\_hidden\_States$ is true.

### 4.3 Results and analysis

As shown in Table 1, the data set is balanced. In this task, we use accuracy to measure the performance of the proposed method. Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

| | sent size | label 0 | label 1 |
|---|---|---|---|
| Train | 10000 | 4979 | 5021 |

Table 1: Distribution of labels in train datasets.

Table 2 shows the system performance of the various methods on the test set. BERT has a significant advantage over other methods on the test set. It is likely because the original BERT models trained on BookCorpus(Zhu et al., 2015) and English Wikipedia contain sufficient common knowledge. We think this can be attributed to BERTs training on Next Sentence Prediction, which can assist in handling the logic relationship between two sentences. (Jawahar et al., 2019) points out that the high hidden layer of BERT can learn rich semantic information features. Therefore, in order to get more abundant semantic information features, we also take advantage of the top hidden layer state of BERT. Finally, we achieve competitive results with an accuracy of 85.1%, ranking 25th in subtasks.

| **Our Method** | **Accuracy** |
|---|---|
| SVM | 52.3% |
| BiLSTM-CNN | 64.4% |
| BiLSTM-BiGRU-HAN | 67.9% |
| BiLSTM-BiGRU-Attention | 69.3% |
| BERT-BASE | 78.1% |
| BERT-BASE(5-fold) | 79.8.% |
| **BERT with Hidden State(our model with 5-fold)** | **85.1%** |

Table 2: Our result using different methods on the test set.

## 5 Conclusion

In this paper, we address the challenge of automatically differentiate natural language statements that make sense from those that do not make sense. We conduct experiments with SVM, word embedding model, and BERT. After comparison, we obtain more abundant semantic information features by concatenating the original output of BERT and the output vector of BERT hidden layer state. The result shows that it is helpful to improve the performance of BERT to obtain more abundant semantic information features by extracting the hidden state of BERT. The code is available online.[5]

## Acknowledgments

This work was supported by the Natural Science Foundations of China under Grants 61463050, the NSF of Yunnan Province under Grant 2015FB113.

## References

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

[5]https://github.com/byew/counter

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*.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Ganesh Jawahar, Benot Sagot, and Djam Seddah. 2019. What does bert learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Marvin Minsky. 2000. Commonsense-based interfaces. *Communications of the ACM*, 43(8):66–73.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Shane Storks, Qiaozi Gao, and Joyce Yue Chai. 2019. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *ArXiv*, abs/1904.01172.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1957–1968.

Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual lstm model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy, July. Association for Computational Linguistics.

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.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.