Warren at SemEval-2020 Task 4: ALBERT and Multi-Task Learning for Commonsense Validation

Yuhang Wu, Hao Wu*
School of Information Science and Engineering
Yunnan University, Kunming, P.R. China
*Corresponding author, haowu@ynu.edu.cn

Abstract

This paper describes our system in subtask A of SemEval 2020 Shared Task 4. We propose a reinforcement learning model based on MTL(Multi-Task Learning) to enhance the prediction ability of commonsense validation. The experimental results demonstrate that our system outperforms the single-task text classification model. We combine MTL and ALBERT pretrain model to achieve an accuracy of 0.904 and our model is ranked 16th on the final leader board of the competition among the 45 teams.

1 Introduction

Common sense is a fundamental ability of human beings to understand the objective world. In the process of human growth, common sense is mastered through various studies. It can help people to infer other things from one fact when learning new knowledge. However, most current natural language processing (NLP) models do not have such a knowledge model, so they cannot understand and solve new problems well. It is particularly important to introduce common sense to natural language understanding (NLU) systems (Wang et al., 2019). SemEval 2020 Shared Task 4 (Wang et al., 2020) aims at testing whether a system can distinguish one statement that makes sense from another statement that makes no sense and reason out why the faked one makes no sense. Two examples of Commonsense Validation are shown is Table 1.

Statements	Classes	(Pseudo label)
The parrot went to work. The man went to work.	non-commonsense commonsense	0.222
A girl plays volleyball. A dog plays volleyball.	commonsense non-commonsense	1.125

Table 1: Dataset example.

A general paradigm to solve this task is through a binary classification model, where various neural networks are used to extract features of statements firstly, and then followed by a classification layer. However, this paradigm is often constrained by single objective optimization, which leads to poor generalization ability of the model. MTL (Zhang and Yang, 2017) can effectively cope with this problem and improve the prediction performance of multiple tasks at the same time (Ruder, 2017). To this end, we propose a reinforcement learning model built on the principle of MTL to enhance the prediction ability of commonsense validation. The features of statements are extracted with ALBERT model (Lan et al., 2019) to do commonsense validation classification. At the same time, we calculated text similarity for two statements, and make pseudo labels to construct a regression task to optimize. By this, we transfer the task of commonsense validation classification to a MTL problem. By optimizing two prediction tasks simultaneously, the generalization ability of the classification model is improved and then results into better accuracy of predictions.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 620–625
Barcelona, Spain (Online), December 12, 2020.

The remaining paper is organized as follows: we detail the model architecture in Section 2, present the experimental results and make the analysis in Section 3, and conclude our work in Section 4.

2 Methods

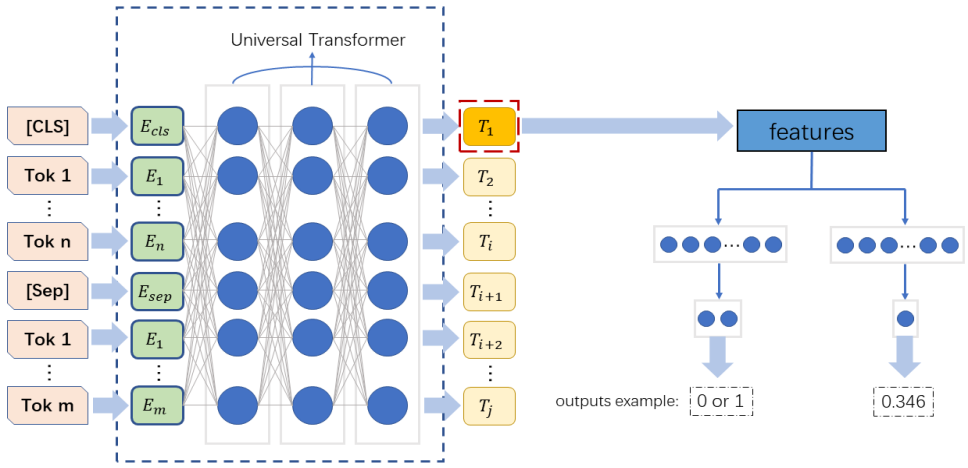

Figure 1: ALBERT and MTL model

Our model architecture can be roughly split into two modules, including an ALBERT component to learn the features of the input texts, and a multi-task component for generation of text similarity and commonsense reasoning classification. The details are presented as Figure 1.

2.1 ALBERT component

In recent years, using pre-trained models (ELMO, GPT, BERT, ALBERT, etc.) often achieve excellent results in mainstream NLP tasks (MNLI, QQP etc.(Wang et al., 2018)), so fine-tuning pretrained models has become an important method for current researchers.

ALBERT (Lan et al., 2019) is latest pretrained model developed from BERT (Devlin et al., 2018), which reduces the number of parameters through factorized embedding parameterization, and utilizes cross-layer parameters sharing technique for improving the training efficiency. ALBERT can accept one text, or a token sequence composed of multiple texts. Because the Transformer model cannot remember the time sequence information, we use [CLS] to indicate that the feature is used for classification task, insert [SEP] between the first and second statements and concatenate them, and we send the token sequence which is like $[CLS, word_1^1, word_2^1, ..., SEP, word_1^2, word_2^2, ..., word_n^2]$ to the Embedding layer of ALBERT. Then we get word embeddings $e_{cls}, e_1^1, e_2^1, ..., e_{sep}, e_1^2, e_2^2, ...e_n^2$ and fed them to ALBERT model, finally we obtained vector representations $\boldsymbol{X}$ from the last layer of the ALBERT model. $\boldsymbol{X_{cls}}$ is the set of the first element of all rows of $\boldsymbol{X}$, which we treat it as the input text feature to MTL component. At the same time, ALBERT pre-trained model is still fine-tuning.

2.2 MTL component

MTL is derived from transfer learning. We utilize MTL for decreasing the calculation scale and co-optimizing shared parameters mainly through hard parameters sharing. Dense layer is capable of learning higher-order interactions between features and thus provide additional feature selection and weighting functionality by using a nonlinear transformation. So we send feature $\boldsymbol{X}$ to two dense layers to extract implied information.

$$
\begin{aligned}
&u_1 = Relu(W_{(l)}\boldsymbol{X}_{cls} + b_{(l)}),\\
&u_2 = W_{(2)} * u_1 + b_{(2)},\\
&\alpha = Softmax(u_2)\\
&category\ label = argmax(\alpha)
\end{aligned}
\tag{1}
$$

where $\boldsymbol{X_{cls}}$ is the feature vector extracted from ALBERT model, l represents neurons number. $W_{(l)}, W_{(2)}$ and $b_{(l)}, b_{(2)}$ denote the weight matrixs and bias vectors, respectively. We use ReLU to avoid the problem of vanishing gradient. For category prediction, two neurons are used in the last layer, and we use softmax function to obtain the output with the highest probability as the category label.

At the same time, we use WMD (word mover's distance (Kusner et al., 2015)) to calculate and normalize the text similarity of two statements, and treat it as a pseudo label (Lee, 2013) for a regression task in our model. Calculation of WMD can be described as following:

$$
\begin{aligned}
&d_i = \{\boldsymbol{v_{i1}}, \boldsymbol{v_{i2}}, ..., \boldsymbol{v_{ij}}\}, i \in (1, 2)\\
&C(k, l) = ||\boldsymbol{v_{1k}} - \boldsymbol{v_{2l}}||_2,\\
&S_{wmd} = \sum_{k,l=0} T_{kl} C(k, l),\\
&S^* = \frac{S_{wmd} - S_{min}}{S_{max} - S_{min}}.
\end{aligned}
\tag{2}
$$

where d_i represents the set of word vectors in the two statements, $\boldsymbol{v}_{ij}$ represents the word vector of the j-th word of the i-th statement, then we calculate the 2-Norm of $\boldsymbol{v_{1k}} - \boldsymbol{v_{2l}}$ as transfer cost of two word vectors. After we calculate the transfer cost of all $\boldsymbol{v}_{ij}$ word vectors in the two statements, we multiply them by weights T to sum them up, we will get S_{wmd} which we treat as the similarity of two statements. We min-max normalize S_{wmd} in order to reduce the fluctuation of pseudo labels and make sure each pseudo label is between 0 and 1 for training.

$$
\begin{aligned}
&t_1 = Relu(W_{(m)}\boldsymbol{X}_{cls} + b_{(m)}),\\
&similarity = W_{(1)} * t_1 + b_{(1)}
\end{aligned}
\tag{3}
$$

where $\boldsymbol{X_{cls}}$ is the feature vector extracted from ALBERT model, $W_{(m)}, W_{(1)}$ and $b_{(m)}, b_{(1)}$ denote the weight matrixs and bias vectors. One neurons are used in last layer for similarity prediction. We consider the output as similarity and we use the S^* for back-propagation to optimize our model.

2.3 Model Training

As for model training, we construct different loss functions for two tasks. For the classification task, we adopt the classical cross-entropy loss function, which is only associated with the probability distribution of the outputs. Compared to mean square error, when the training gradient reaches the boundary, the attenuation of the training speed will be reduced, so we use cross-entropy can make the model converge faster. For the regression task, mean square error can measure the difference between the predicted value and the label value, and can amplify the value with a larger deviation.

$$
\begin{aligned}
&CE(p, Y) = \frac{1}{n}\sum_{i=1}^{n} -Y_i log(p_i) - (1 - Y_i) log(1 - p_i),\\
&MSE(y, \bar{y}) = \frac{1}{2m}\sum_{j=1}^{m}(y_j - \bar{y}_j)^2,\\
&loss = \delta CE + \xi MSE
\end{aligned}
\tag{4}
$$

where Y_i is value of commonsense label (0 or 1), p_i is probability of Y_i. y is the predicted text similarity. $\bar{y}$ represents pseudo label. In order to focus on the training of the classification task, we add different weights to the two losses, which we set δ to 1.0 and ξ to 0.1, For each layer, model parameters can be upgraded with the back propagation algorithm which iteratively computes gradients using the chain rule. Many gradient descent optimization algorithms, such as Adam, Adagrad, Adadelta and RMSprop can be utilized. Moreover, a *mini-batch* gradient descent, which processes a small subset of the training set in each iteration, is necessary to speed up training on the large-scale datasets.

3 Experiments

3.1 Dataset

The experimental dataset contains the train data, dev data and test data. Statistics of the dataset are shown in Table 2, where data is relatively balanced and does not need us to do distribution processing.

Data set	label(0)	label(1)	total
Train set	4979	5021	10000
Dev set	518	479	997
Test set	–	–	1000

Table 2: Statistics of the dataset.

3.2 Implementation details

In order to generate pseudo labels, we use **GoogleNews-vectors-negative300**[1] which were pre-trained on a large quantity of Twitter messages to embed the statements and converts the words into 300-dimensional vectors. Then we apply these vectors to calculate the WMD.

ALBERT-base[2] and BERT-base were pre-trained with BOOKCORPUS (Zhu et al., 2015) and English Wikipedia (Devlin et al., 2018). We use them to compare the performance between BERT and ALBERT for reducing calculating resource consumption and training time. We use ALBERT-xxlarge in final submission which achieves accuracy of 0.904 in Test set for achieving the best goals of the submission.

We utilize keras-2.3.1[3] to build the network framework and bert4keras[4] to build the ALBERT model. To compare the performance difference between single-task and multi-task, we used the same hyper-parameters for both single-task and multi-task models. Adam optimizer has the characteristics of efficient calculation, not affected by gradient scaling, automatic adjustment of learning rate, etc. In most cases, it is an excellent optimizer. We set the initial learning rate of Adam optimizer to 5e-6 to avoid loss explosion. And we let the batch size be 32 and the epoch be 20 to ensure that the model is fully-trained.

In order to be consistent with the subtask A ranking indicator, our system will be evaluated using accuracy, the metric is defined as follow:

$$Accuracy = \frac{number\ of\ correctly\ predicted\ labels}{number\ of\ predicted\ labels} \tag{5}$$

3.3 Result analysis

As shown in Figure 2, we record loss change in a single experiment, we can see that convergence speed of ALBERT is faster than BERT, because the dropout mechanism is removed and ALBERT's parameter sharing. ALBERT can achieve the best performance in the 3rd epoch, while BERT achieve best performance in 12th epoch. From Table 3, the ALBERT parameters are about a-tenth of BERT, the training time to achieve the best effect is a-third shorter than BERT, but the ALBERT can achieve better

[1]https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[2]https://github.com/google-research/albert
[3]https://github.com/keras-team/keras
[4]https://github.com/bojone/bert4keras

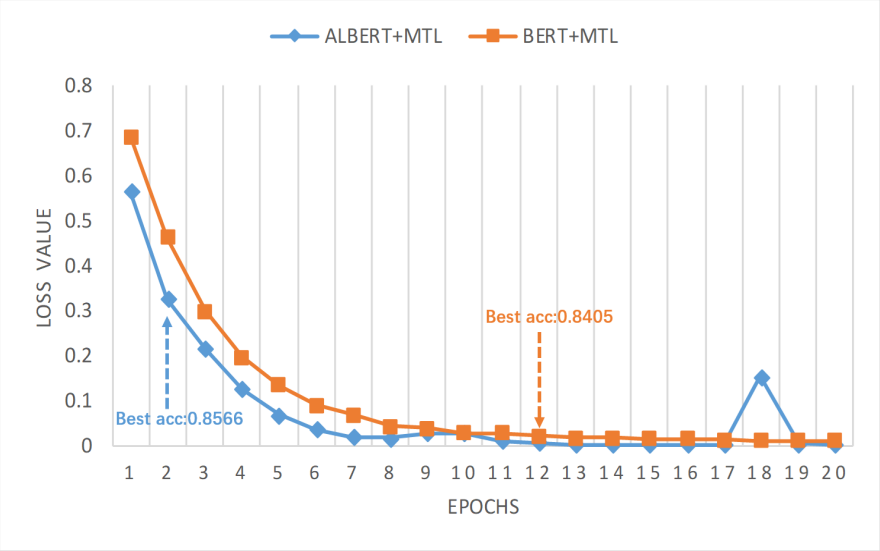

Figure 2: The training process

accuracy of 1.9% than BERT. We can conclude that it is not necessarily the more complex the better for pre-trained model, the dropout used by the super-large model is contrary to performance improvement, we cannot continuously increase the scale of parameters while discarding many learning features which are inefficient.

We further compared the multi-task model and the single-task model, and we discussed the model performance improvement of separate learning rates, as showed in Table 4. For each setting, we repeat experiments five rounds and take the averaged accuracy as the result. We can see that MTL improves the accuracy by 0.37% with the help of parameter sharing and joint optimization. And we find that the Dense layers of the multitasking component were randomly initialized, using the same learning rate with the ALBERT component would easily cause MTL component under-fitting. The results also show that separate learning rates($LR_{ALBERT} = 5e-6, LR_{MTL} = 0.1$) can improve accuracy by 0.54%. Finally, we point that our model has an accuracy improvement of 0.91% over the ALBERT single-task model.

model	**parameters**	**time per epoch**	**accuracy**
BERT-base+MTL	108M	120s	0.8494
ALBERT-base+MTL	12M	87s	0.8570

Table 3: ALBERT vs BERT

model	**accuracy**
ALBERT-base (lr=5e-6)	0.8538
[ALBERT-base+MTL](lr=5e-6)	0.8570
ALBERT-base (lr=5e-6)+MTL(lr=0.1)	0.8616

Table 4: Accuracy on dev set

4 Conclusions

We have described our ALBERT and MTL system used for SemEval 2020 Task4 subtask A. Our model has the advantages of both the ALBERT pre-trained model and MTL framework. The experimental results have upheld the merits of our proposed methods, where the accuracy rate is improved by 0.91% as compared to the ALBERT single-task model and improved by 1.44% compared to the BERT MTL model.

In the future, we will re-select the branch task to replace the similarity prediction task of semi-supervised learning with the text generation task of subtask C, strengthen the semantic information of the data, and explore the feasibility of this idea.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (61962061), partially supported by the Yunnan Provincial Foundation for Leaders of Disciplines in Science and Technology, Top Young Talents of "Ten Thousand Plan" in Yunnan Province, the Program for Excellent Young Talents of Yunnan University.

References

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*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966.

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*.

Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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. *CoRR*, abs/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.

Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

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.