Self-Refine Learning For Data-Centric Text Classification

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

13-09-2021 / 16-09-2021

CITATION

Guo, Tong (2021): Self-Refine Learning For Data-Centric Text Classification. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.16610629.v1>

DOI

[10.36227/techrxiv.16610629.v1](https://doi.org/10.36227/techrxiv.16610629.v1)

Self-Refine Learning For Data-Centric Text Classification

Tong Guo

779222056@qq.com

Abstract. In industry NLP application, our manually labeled data has a certain number of noisy data. We present a simple method to find the noisy data and re-label their labels to the result of model prediction. We select the noisy data whose human label is not contained in the top-K model’s predictions. The model is trained on the origin dataset. The experiment result shows that our method works. For industry deep learning application, our method improve the text classification accuracy from 80.5% to 90.6% in dev dataset, and improve the human-evaluation accuracy from 83.2% to 90.5%.

Keywords: Deep Learning · Text Classification · NLP

1 Introduction

In recent years, deep learning [2] and BERT-based [1] model have shown significant improvement on almost all the NLP tasks. However, the most important factor for deep learning application performance is the data quantity and quality. We try to improve performance of the industry NLP application by correcting the noisy data by other most of data.

Previous works [11] first find the noisy data which human label and model prediction is not equal and re-label the noisy data manually. During the correction, the last human label and model prediction is viewed by the labeling people. However it need more human labeling. So in this work, we directly re-label the noisy data whose label is not contained in the top-K ($K=1,2,3...10$) predictions of model. We re-label the noisy data’s label to the top-1 prediction of model.

Our key contribution is:

Based on our industry dataset, we first find the noisy data which human label is not in the top-K ($K=1,2,3...10$) predictions of model. Then we re-label the label of noisy data to the top-1 prediction of the model. The experiment results shows that our idea works for our large industry dataset.

2 Relate work

BERT [1] is built by the multi-layer transformer encoder [10], which produces self-attended token representations that have been pre-trained from unlabeled

Fig. 1. Our method: re-label the noisy human labels to the model-A’s predictions

text and fine-tuned for the supervised downstream tasks. BERT achieved state-of-the-art results on many sentence-level tasks on the GLUE benchmark [3] and CLUE[12] benchmark.

Our method is different to semi-supervised learning. Semi-supervised learning solve the problem that making best use of a large amount of unlabeled data. These works include UDA [6], Mixmatch [7], Fixmatch [8], Remixmatch [9]. Our work is full supervised.

3 Our method

In this section, we describe our method in detail. Our methods is shown in Fig 1. It includes 5 steps:

Step 1, in order to solve our industry text classification problem. We manually label 2,790,000 data and split them into 2,700,000 training data and 90,000 dev data.

Step 2, we train / fine-tune the BERT model on the 2,700,000 training data. We named the result model of this step Model-A.

Step 3, we use Model-A to predict for all the 2,790,000 data. Then we find all the data whose human label are not in the top-K ($K=1,2,3...10$) predictions of model-A. We consider they are the noisy data.

Step 4, we re-label the noisy data’s human label to the top-1 prediction of model-A. Then we split the same 2,700,000:90,000 training and dev dataset.

Step 5, we train and evaluate upon the dataset of step 4 and get Model-B. As we also re-label the dev dataset by the top-1 prediction of model-A, we also manually evaluate the performance of our method.

4 The Model

We use BERT as our model. The training steps in our method belongs to the fine-tuning step in BERT. We follow the BERT convention to encode the input text.

5 Experiments

In this section we describe detail of experiment parameters and show the experiment result. The detail result is shown in Table 2. The data size in our experiment is shown in Table 1.

In fine-tuning, we use Adam [4] with learning rate of $1e-5$ and use a dropout [5] probability of 0.1 on all layers. We use BERT-Base (12 layer, 768 hidden size) as our pre-trained model.

Table 1. The data sizes for Method 1

Data Size	Description
11,000,000	All the data in our application database.
2,790,000	All the data we label in step 1 of Fig 1.
2,700,000	The the training dataset we split from the 2,790,000 data.
180,000	All the noisy data we select from the 2,790,000 data to re-label.
90,000	The dev dataset we split from the 2,790,000 data.

Table 2. The experiment result of text classification. The dev dataset is the 90,000 data. The accuracy reported by human is 5000 data that sampled from the 11,000,000 data. Model-A, Model-B are corresponding to Fig 1.

Model In Fig 1	Dev Dataset Accuracy	Human Evaluation Accuracy
Model-A	80.5%	83.2%
Model-B with K=3	84.8%	90.1%
Model-B with K=1	90.6%	90.5%

6 Analysis

As the whole dataset is large enough, so we re-label the ground truth of noisy data by other most of the data.

7 Conclusion

The experiment result shows our idea works. Our idea can apply to a broad set of deep learning industry applications. We will do the experiments like [11] that inject the prediction result of model-A to model-B.

References

1. Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
2. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25: 1097-1105.
3. Wang A, Singh A, Michael J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding[J]. arXiv preprint arXiv:1804.07461, 2018.
4. Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.
5. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.
6. Xie Q, Dai Z, Hovy E, et al. Unsupervised data augmentation for consistency training[J]. arXiv preprint arXiv:1904.12848, 2019.
7. Berthelot D, Carlini N, Goodfellow I, et al. Mixmatch: A holistic approach to semi-supervised learning[J]. arXiv preprint arXiv:1905.02249, 2019.
8. Sohn K, Berthelot D, Li C L, et al. Fixmatch: Simplifying semi-supervised learning with consistency and confidence[J]. arXiv preprint arXiv:2001.07685, 2020.
9. Berthelot D, Carlini N, Cubuk E D, et al. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring[J]. arXiv preprint arXiv:1911.09785, 2019.
10. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.
11. Guo, Tong. "Learning From How Human Correct." arXiv preprint arXiv:2102.00225 (2021).
12. Xu, Liang, et al. "Clue: A chinese language understanding evaluation benchmark." arXiv preprint arXiv:2004.05986 (2020).