An Empirical Analysis of Fine Tuning Techniques for Question Answering Task

Yongmin Lee
20170475
lym7505@kaist.ac.kr

Donghyun Kim
20170086
github.com/chocolatefudge/nlpqa

Abstract

Recent trends in Natural Language Processing is mostly about gigantic, unsupervised pre-trained language models that could be fine-tuned for many target tasks. It dominates most of the NLP tasks including question answering. However, specific fine-tuning methodologies are not well provided in most of the question-answering task. Performing pre/post-processing or architectural change might bring improvements to our model. Therefore, in this research, we make an effort to find a strategy that is especially well suitable for question answering task with korquad-open dataset.

1 Introduction

Question answering task is defined as building automated systems that answer questions posed by humans in natural language. There could be several forms of the question-answering task regarding its contents. In this project, question and set of relevant contexts which is a search result of wiki, web, etc is given, and we have to infer the answer from given contexts. A methodology that currently dominates the question answering domain is massively pre-trained language models, such as BERT(Devlin et al., 2018) or ELECTRA(Clark et al., 2020). They all require a large amount of data in their pre-training stage. Also, even though they're pre-trained in an unsupervised way, they achieve good performance on many target tasks only when fine-tuned for a relatively small proportion. By these characteristics of language models, it's obvious that we're not able to pre-train a new model from scratch due to various limitations. Consequently, the only strategy that we could take is to solely focus on the fine-tuning step.

Our experiments are closely related to the entire pipeline of fine-tuning the language model. It could be divided into input, model, and output part. Before the data is fed into the model, we may modify this input by performing a data augmentation. For the model itself, options we have are to change the model itself, tune the hyperparameters, or modify the training structure by adopting student-teacher scheme. Finally, for the output layer, we can modify loss function and assign different weights to loss terms. By conducting various experiments for all these methods, we concluded that using a self distillation structure resulted in better performance, while other techniques brought negative effects.

2 Method

2.1 Changing Model and Hyperparameters

Two famous models in NLP is BERT and ELECTRA. They are both based on the transformer structure, which is an encoder-decoder architecture relying only on attention mechanism(Vaswani et al., 2017). BERT and ELECTRA mainly differ in their pre-train tasks. Although it's hard to say that one model is superior to the other, ELECTRA is generally known to have faster pre-training time than BERT.

In-depth, models that we used in this experiment is multilingual BERT(Huggingface) and KoELECTRA(monologg). For multilingual BERT, it's trained with 102 different kinds of languages. In case of KoELECTRA, it's trained with entirely Korean words. Therefore, we expected more accuracy and shortened training time by changing our model to KoELECTRA. Furthermore, simple hyperparameter tuning is performed on our model.

2.2 Loss Function Modification

As mentioned in 1, one existing variant in the output layer of the fine-tuning stage can be loss function modification. In the original QA model, a fully connected layer is applied to yield a final output, and it's used to predict the start/end position. Also, start and end position is set to zero in unan-

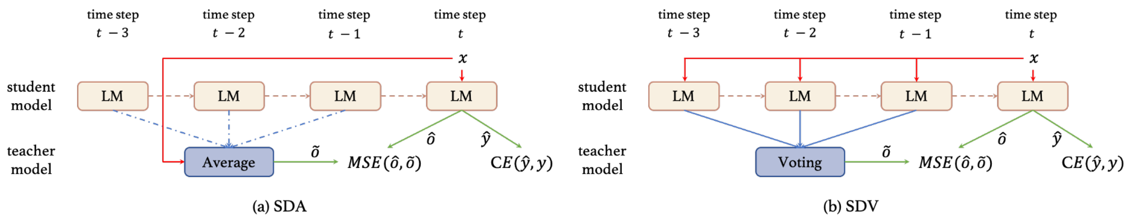

Figure 1: The structure of the Self-Distillation model. The teacher model is the ensemble of student models over recent K time steps, and K = 3 here. (a) SDA model and (b) SDV model takes different ensembling strategy.

swerable contexts. Since this loss function consists of start/end and answerable/unanswerable components,loss function can be adjusted as in Zhang and Xu (2019).

First, we weighted start loss as twice as end loss. Because knowing the correct start position is unambiguous in most cases, we thought the start position is more important in increasing the model's performance. Second, we weighted answer loss twice as unanswerable loss, as there's much less answerable data in our contexts. In this way, we tried to resolve this unbalance by paying more importance to answerable data. Overall, our objective is for the model to reach better optima by modifying weights between various losses.

2.3 EDA

Observing that data augmentation acts as a crucial component in most of the vision tasks, we incorporate augmentation also in our QA task. In general, augmenting data, can reduce overfitting and make models to learn general representation from data. Among many augmentation techniques existing for text data, we selected EDA(Easy Data Augmentation), which blindly modifies given text data. From the original EDA(Wei and Zou, 2019), synonym replacement and random insertion were removed because there's no synonym library for Korean text. Instead, our EDA consists of three operations which are Random Swap, deletion, and duplication. Also, the question text is not augmented because it is often very short and contains crucial information. Furthermore, the answer part of the context task is also not augmented since the model has to correctly find the answer if it exists. Even though this doesn't consider semantics, we expected EDA to be a powerful method to reduce overfitting and add robustness to our model.

2.4 Self-Distillation

Apart from the data or loss layer, changing the overall structure is also possible for the fine-tuning step. Models used in this experiment is SDA and SDV, which is Self Distillation Averaged, and Self distillation Voted, respectively citepxu2020improving. As you can see from Figure 1, in SDA and SDV, the model is guided from their previous versions. As a result, we could mimic the benefit of the student-teacher model without increasing memory much.

Yet, SDA and SDV differ in their teacher model construction. For SDA in Equation 1, a single teacher model($\hat{\theta}$) is generated by averaging the previous K model's parameters and produces a single output. For SDV in Equation 2, there's K different teacher model and their output is averaged instead of parameters. MSE term between teacher model and student model's output is added into total loss function as an auxiliary objective.

$$\begin{aligned}\mathcal{L}_\theta(x,y) &= CE(\mathrm{LM}(x,\theta),y)\\ &+ MSE(\mathrm{LM}(x,\theta),\mathrm{LM}(x,\hat{\theta}))\end{aligned} \tag{1}$$

$$\begin{aligned}&\mathcal{L}_\theta(x,y) = CE(\mathrm{LM}(x,\theta),y)\\ &+ MSE(\mathrm{LM}(x,\theta),\frac{1}{K}\sum_{k=1}^{K}\mathrm{LM}(x,\theta_{t-k}))\end{aligned} \tag{2}$$

Knowledge distillation is generally known to improve student models by using a fixed well-trained teacher model. However, in our method, the teacher model is created as a partial ensemble of students model at each step. Though it may not fulfill its role as a good teacher in the early phase, we expected student and teacher model to boost each other's performance gradually and hence yield a better model.

3 Experiments

All experiments proceed with KoELECTRA-v2-discriminator model. Context per question is set

Model	lr	Dropout	F1	EM
BERT	5e-5	0.1	73.71	72.32
KoELECTRA	5e-5	0.1	76.96	75.70
KoELECTRA	1e-5	0.1	**78.11**	**76.94**
KoELECTRA	5e-5	0.2	76.33	75.16
KoELECTRA	1e-5	0.2	78.09	76.77

Table 1: Comparison of two different pre-trained model and result of hyperparameter tuning on KoELECTRA.

to 3, with maximizing the number of answerable contexts. Adam optimizer is used with adam epsilon of 1e-8 and batch size of 24. The learning rate linearly decreases during the entire training step. For the dataset, only the given 60k data is used for training, and 6k of validation data is used only for validation purposes.

3.1 Changing model and Hyperparameters

As shown in Table 1, KoELECTRA outperformed multilingual BERT in both F1 and EM metrics. Since KoELECTRA is entirely pre-trained in Korean, we concluded that it has better performance in Korean text. Moreover, the ELECTRA model has approximately two-third of train time compared to BERT. Therefore, since the KoELECTRA model is both accurate and efficient, we conducted hyperparameter tuning on KoELECTRA.

Table 1 shows the result of simple hyperparameter search about learning rate and dropout. First, a smaller learning rate produced a better result in both dropout settings. Without decreasing the training speed, a smaller learning rate prevented our model from perturbing in its later phase of training and hence yield better performance. In contrast, in a bigger learning rate, both F1 and EM fluctuated even after the model is trained for sufficient epochs. In the case of dropout, overall performance didn't change much concerning dropout values. However, larger dropout took significantly longer training time to reach the same accuracy. Therefore, KoELECTRA with dropout 0.1 and LR of 1e-5 has become our new baseline for proceeding experiments.

3.2 Loss Function Modification

Unlike our hypothesis, modifying loss function had a negative result on both model's F1 and EM scores (See Table 2). More specifically, giving answer loss more weight had the most negative effect in overall performance, while weighting start loss more also didn't help the language model to improve. Failure of start loss implies that more accurate start loss doesn't always guarantee a more correct answer. One main reason is unlike human, our language model is based on the transformer structure, not in uni-directional RNN.

For the answer loss, we got lower performance on the validation set but better score on the test set. This is because the evaluation metric of the two datasets is quite different. The F1 score for the validation is measured per paragraph, which can be dominated by unanswerable data. In contrast, the F1 score for the test is measured per question and hence weighting answerable loss works better in here. However, since this discrepancy between two prediction metrics was revealed after the presentation, we didn't have enough time to prove this hypothesis.

Model	F1	EM
Baseline	**78.11**	**76.94**
Weighted start loss	77.88	76.59
Weighted answer loss	77.13	75.78
Weighted start&answer loss	77.17	75.78

Table 2: Results of modifying loss function.

Model	F1	EM
Baseline	**78.11**	**76.94**
Strong EDA (10%)	63.52	63.37
Weak EDA (5%)	72.21	70.60
Scheduled EDA (0 $\rightarrow$ 10%)	77.30	76.03

Table 3: Results of EDA with different augmentation scheduling.

3.3 EDA

From the results of Table 3, data augmentation techniques also showed a negative effect on our model. In the first experiment, each augmentation is applied for a probability of 10%. However, this leads the language model to near malfunctioning. By reducing the intensity of augmentation and incorporating with unaugmented data, and by ramping up the augmentation intensity, we could recover some performance. However, since there was no evidence that augmentation had benefit our model, we concluded that EDA augmentation was unsupportive for our language model.

The reason why EDA failed in our model can be roughly summarized as follows. First, it destroyed sentence structure and grammar. It simply disturbed our language model rather than increasing data's diversity and prevent overfitting. Second, we had not enough epochs for augmentation to work in practice. Unlike most of the CV tasks where we could train for hundreds of epochs, we're only allowed to train for no more than 10 epochs and therefore our model is not exposed to diverse augmented forms.

3.4 Self-Distillation

Changing the training structure to the student-teacher model by using the SDA or SDV model gained some performance. In-depth, the SDA model resulted in slightly worse F1 and EM values, whereas the SDV model performed better than the baseline. (See Table 4) The reason for the difference between SDA and SDV models could be found in the MSE term between student and teacher model.

Because SDA's MSE value is relatively high and unstable compared to that of SDV, SDA's correlation between teacher and student is often unsteady and disagrees. This is because, for the averaged model, a single teacher model's quality can largely be judged by each previous student model. However, in an SDV model, because we're averaging in the output stage, the teacher model's quality is ensured in most cases even though the model's performance temporally decreases. And because more qualified teacher leads to increase the robustness of a student model, SDV model is more beneficial when dealing with hard tasks such as question answering.

Therefore, we concluded that the student-teacher model in the self distilled version can benefit the fine-tuning process of the question-answering task. Still, we felt that the "self" model cannot increase the performance for a great margin since there's a strict limitation that we're using only one model. Thus, an ensemble model in which trains from different seeds in a parallel manner can yield better results.

Model	F1	EM
Baseline	78.11	76.94
SDA (K=3)	77.75	76.50
SDA (K=5)	77.76	76.50
SDV (K=3)	78.13	76.85
SDV (K=5)	**78.27**	**77.07**

Table 4: Results of the Self-Distillation model using different ensemble methods and teacher size.

4 Failure Analysis

Since most of the methods proposed in this research failed to produce good results, theoretical analysis on the reason for failure is done to summarize the reasons.

- We neglected the characteristics of our dataset itself. Some context has useful data, while others have much less amount of data, and therefore could confuse our model. Before diving into a methodology, correctly pre-processing the dataset was necessary.

- There was an error in the validation code, as mentioned in 3.3. We didn't recognize it because we thought the prediction function is valid and cannot be modified. However, this resulted in a huge discrepancy between validation and test scores and made us misinterpret the accuracy scheme. We should have discovered and fixed the training pipeline a bit to be more coherent to test metrics.

- Our methods were too independent of each other. We're just able to briefly conduct experiments in each domain, not relating them to each other. For better results, we should have integrated these methods, or dive into one specific method in depth.

5 Conclustion

From all the experimental results and analysis, we can conclude as follows.

- Smart fine-tuning cannot overcome the power of pre-training. While performance has drastically increased by simply replacing the model, most of the fine-tuning methods we devised had a really small impact on the performance.

- If fine-tuning is the only option, naively applying ideas such as loss function modification or data augmentation is often harmful. Successfully pre-processing the data and adding the teacher-student model can enhance the performance.

- Possible improvements could be made by ensemble structure or semantic augmentation. Also, since many experiments in this research are based in BERT, conducting those in BERT could produce different outcomes.

References

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

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*.

Huggingface. Multilingual bert on huggingface github repository. `https://github.com/huggingface/transformers`.

monologg. Koelectra on monologg github repository. `https://github.com/monologg/KoELECTRA`.

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.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Yuwen Zhang and Zhaozhuo Xu. 2019. Bert for question answering on squad 2.0.