Enhancing multi-task fine-tuning on BERT-based model

Stanford CS224N {Default} Project

Xiaochen Xiong
Department of Biology
Stanford University
xxc15@stanford.edu
TA mentor: Josh Singh

Abstract

Nature language processing tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity, are important for using and understanding the languages. A recent pre-trained language model called Bidirectional Encoder Representations from Transformers (BERT) can perform multiple nature language processing tasks after fine-tuning with simple additional task-specific layers. However, it is a challenge to train multi-task model with a shared BERT model. In this report, we aim to improve the BERT-based performance for multi-task learning. We first completed an implementation of the base BERT model and showed that it can perform single tasks, namely sentiment analysis. We then applied the BERT model to multiple tasks with a shared BERT model. We tested different sampling approaches, concatenating sentence pair as input, non-linear neural network as task-specific layer, and further pre-training to enhance the multi-task performance. We are able to achieve SST acc. of 0.528, QQP acc. of 0.886, and STS corr. of 0.875 on the unseen test datasets. Besides focusing on the performance, we assessed the learning characteristics of sequential learning and pre-training; and analyze the model predictions in detail to understand the potential causes leading to errors. In summary, we report here methods that can improve the multi-task fine-tuning with BERT and potential future directions to further improve the performance.

1 Introduction

Large language model fine-tuning with pre-trained model is an effective way of tackling many natural language processing tasks. Recent transformer models using attention to learn universal language representations (Vaswani et al., 2017). BERT integrating context from both left and right enables it to have a more comprehensive understanding of languages (Devlin et al., 2018). BERT also demonstrates that a model pre-trained on a large corpus can be relatively readily fine-tuned to create state-of-the-art models for a wide range of tasks. Here, we apply the BERT model for multi-task learning.

Multi-Task Learning (MTL) is the process of learning different tasks using a shared representation (Caruana, 1997) (Liu et al., 2019). MTL can be challenging because gradients from different tasks may conflict between each other and harm the making progress (Yu et al., 2020). In this project, we aim to fine-tune the BERT model to perform three tasks, namely sentiment analysis, paraphrase detection, and semantic textual similarity. This will serve as an example of how to enhance the pre-trained model to be more generalized to different natural language processing tasks.

2 Related Work

2.1 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) has been pre-trained for masked-language modeling (MLM) and next sentence prediction (NSP)(Devlin et al., 2018). MLM allows BERT to understand the relationship between words, while NSP allows BERT to understand the relationship between adjacent sentences. We are inspired by these approach. We tested whether further pre-train the model with domain-specific corpus with MLM can help the model better understand the sentences and enhance the performance. Meanwhile, we reorganized the two sentences of each data point from QQP and STS to mimic the ones from NSP and asked whether the model can re-purpose the inputs (especially token type embeddings) of NSP for QQP and STS tasks.

2.2 Square root sampling and anneal sampling

One of the challenge for MTL is that datasets for different tasks can have vastly different sizes. If training data is sampled proportionally by the dataset size, the model may be trained to bias towards tasks with a lot more data, causing tasks with less data to be under-trained. A generally used method to sample the datasets with a probabilities proportional to the dataset size powered by a factor smaller than 1. A commonly used factor is 0.5 and sampling method can be called "square root sampling". Stickland has also "annealed sampling" method, where the factor decreases linearly with the increase of epoch. (Stickland and Murray, 2019) In this case, tasks are trained more equally towards the end of training, which has shown noticed to be beneficial. We are inspired by their methods and further combined these two methods to formulate a "min anneal sampling" method.

2.3 Further pre-training

Previous studies suggest that further pre-training can improve the task performance(Sun et al., 2019). There are three different ways to perform further pre-training on different datasets: Within-task pre-training (using training data of a target task); In-domain pre-training (using combined training data from the same domain of a target task); Cross-domain pre-training (using training data from multiple domains)(Sun et al., 2019). Here, we adapted empirical hyperparameters from previous studies to do a within-task pre-training (Sun et al., 2019) (Devlin et al., 2018) with some permutations and asked whether further pre-training can improve our task performance and what is the best way of further pre-training.

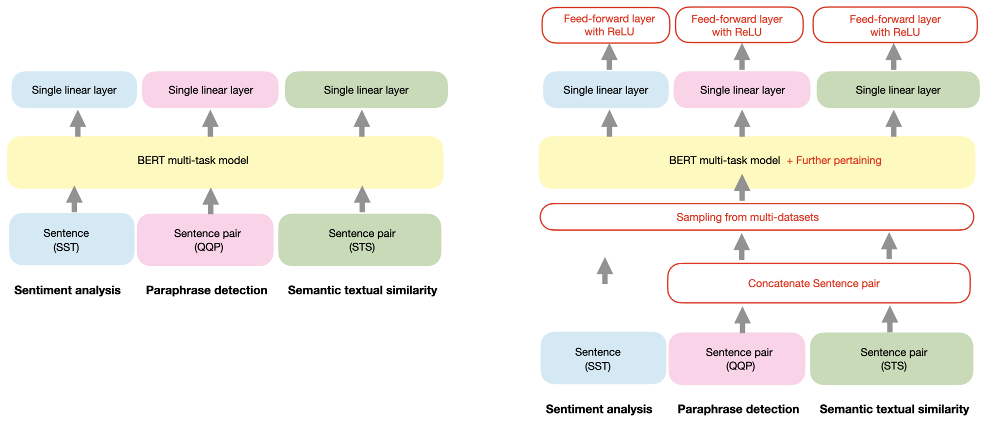


Figure 1: Architecture of baseline (left) and enhanced (right) multi-task BERT model (different extension methods shown in red box)

3 Approach

3.1 BERT multitask baseline

To apply the BERT model for multitasks, we first use the base BERT pre-trained parameters and add one additional task specific layer for different tasks (Figure 1). This serves as our baseline. Here, for the sentiment analysis, we add linear layer with Softmax to the sentence embedding and train with cross entropy loss for the 5-grain sentiment output. For paraphrase detection, we concatenate the sentence pairs and feed through a linear layer, and use the binary cross entropy loss for the binary output. For semantic textual similarity, we also concatenate the sentence pairs and feed through a linear layer and use mean squared error loss.

3.2 Sequential learning and Min anneal sampling

We have two approaches to compare against each other, sequential learning and mixed batch learning. For sequential learning, we train the model with the full dataset for each task sequentially, in the order of SST, QQP, and STS. For mixed batch learning, we mixed mini-batches from 3 tasks. We mixed the datasets by a method combining Square root sampling and anneal sampling. Here, we call "min annealing sampling". At the beginning of the training, the sampling is scaled with square root sampling; while at the end of the training, the datasets are sampled more equally. In this way, we have a overall balanced training across different tasks and even more balanced training at the end of the training phase.

$$p_i \propto N_i^{\alpha} \tag{1}$$

$$\alpha = min(0.5, 1 - 0.9\frac{epoch}{MAXEpoch}) \tag{2}$$

3.3 Concatenated sentences as input

For the paraphrase detection and semantic textual similarity tasks, it involve the comparison of the sentence pair. One way of learning is to input the sentences separately and compare their embeddings in the task-specific layer. Another way of learning is to input the sentences in concatenated manner and output a single embedding for two sentences for downstream tasks. In principle, this approach can unleash the ability of BERT to understand the relationship between sentences, instead of solely taking BERT as a sentence meaning extractor. Specifically, we concatenate the pair of sentences and delete the [CLS] token of the second sentence to form a input as: [CLS] Sentence A [SEP] Sentence B [SEP]. In another variation, we hypothesize the model may fail to distinguish two sentences due to shared token as separation token and end of sentence token. Therefore, we replaced the [CLS] token of the second sentence as a unused token [unused36] to form a input as: [CLS] Sentence A [SEP] [unused36] Sentence B [SEP]. Meanwhile, we add a token type id embedding with first sentence type id as 0 and second sentence type id as 1.

3.4 Non-linear feed-forward layers as task-specific layers

Considering that a single linear layer might not be able to capture more complicated information, we has tested adding another linear layer and a ReLU activation layer between the BERT last embedding layer and task-specific output layer which is written as $y = \text{ReLU}(Wx + b)$.

3.5 Further pre-training

Comparing to the general corpus which BERT has been pre-trained on, the datasets we used for multitask fine-tuning might have different distributions and properties comparing to the general corpus. Therefore, we tried in-task pre-training, where we further pre-train the model using the three datasets for the tasks. The pre-training is done with MLM task with the same masking strategy as orginal BERT pre-training. Each sentence from 3 datasets are randomly mixed with sentence pairs are separately treated as individual sentences. To find out the best pre-trained model, we save the further pre-trained model after 3, 18, and 40 Epochs respectively and assess their performance.

4 Experiments

4.1 Data

This project uses Stanford Sentiment Treebankv(SST) (Socher et al., 2013), CFIMDB, Quora dataset (QQP) (Chen et al., 2018) and SemEval STS Benchmark (Agirre et al., 2013). The information of them are listed in 1.

4.2 Evaluation method

For the BERT single-tasking, we majorly measure the accuracy. As for the BERT multi-tasking, we measure the accuracy for the sentiment analysis and paraphrase detection; and the Pearson correlation for the semantic textual similarity. To evaluate the multi-tasking ability of the model, we also sum

Task	Dataset	Training data	Dev data	Test data
Sentiment analysis	SST	8,544	1,101	2,210
Sentiment analysis	CFIMDB	1,701	245	488
Paraphrase detection	QQP	283,010	40,429	80,859
Semantic textual similarity	STS	6,040	863	1,725

Table 1: Dataset size information for different tasks

up the three metrics: $Overall = Average[Acc_{(SST)} + Acc_{(QQP)} + \frac{1}{2}(1 + Corr_{(STS)})]$. We are focusing on improving the model performance based on these evaluation metrics.

4.3 Experimental details

We used 'bert-base-uncased' from hugging face as our pre-trained foundation model and tokenizer. This model has a hidden size of 768, 12 Transformer blocks and 12 self-attention heads. We further pre-train the model on a NVIDIA T4 with a total of 40 epochs, a batch size of 32, a drop-out rate of 0.1, AdamW optimizerwith 1 = 0.9 and 2 = 0.999. We used a slanted triangular learning rates (Howard and Ruder, 2018) with a base learning rate of 5e-5, and first 70000 steps (10%) as warm-up steps. We fine-tuned this model for our downstream tasks on a NVIDIA T4 with a batch size of 8, learning rate of 1e-5 for full model fine-tuning and 1e-3 for last layer fine-tuning. The dropout probability is always kept at 0.3. We use AdamW with 1 = 0.9 and 2 = 0.999. We have set the max number of the epoch to 10. The final evaluation and accuracy shown are based on the best model on dev set across epochs.

4.4 Results

4.4.1 Single-task learning with the BERT for Sentiment Analysis

To test the performance of using the base BERT model for sentiment analysis, we implement the pre-trained BERT model and then apply one additional linear layer for fine-tuning for either the last linear layer of the BERT model or the full BERT model. The accuracies for SST dataset and the CFIMDB dataset using different fine-tuning modes are shown in the Table 2. Here, we got SST accuracy being 0.390 with only last layer and 0.516 with full model; CFIMDB accuracy being 0.743 with only last layer and 0.971 with full model. These huge leaps suggest that fine-tuning on the full model achieve much better performance than only on the last layer. Full model fine-tuning allows all layers of the BERT model to adapt to the specific task, enabling greater flexibility and capability.

Fine-tuning mode:	last linear layer	full model
SST	0.390	0.516
CFIMDB	0.743	0.971

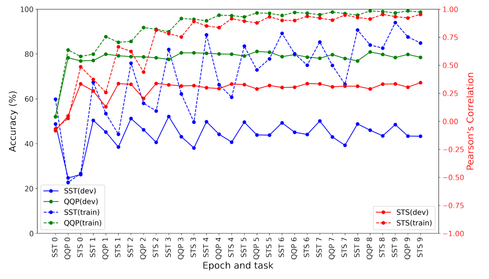
Table 2: Performance of sentiment classification tasks with different fine-tuning modes.

4.4.2 Baseline of multi-task learning with BERT

To set up the baseline for multi-task learning of the model, we add single additional layer and separately fine-tune on last linear layer of the BERT model for each task. After fine-tuning, we get an accuracy for SST and QQP at around 39% and 65.5% respectively. And we get person's correlation for STS at around 0.26 (Table 3). The results suggest that the BERT model is capable of performing multitasks with the same model set and achieve a performance higher than random. While there is still lots of room for further improvement.

4.4.3 Mixed mini-batch learning with min anneal sampling outperforms sequential learning

As indicated with the single-task fine-tuning, we hypothesize that fine-tuning on the full model will give a better performance. To enable full model fine-tuning, we came up with two different approaches, sequential learning or mixed mini-batch learning. With sequential learning, we observed fluctuation of accuracy during the training process, indicating an interference among different tasks (Figure 2a). However, the performance tends to converge, suggesting that sequential learning can efficiently train a multi-task model. While, as expected, the mixed mini-batch learning has a smooth path towards convergence (Figure 2b). In both cases, we observe a strong over-fitting, where the training datasets have very high accuracies/correlation and much higher than the dev sets. Comparing two methods, mixed mini-batch learning has slightly better performance and slightly less over-fitting. Therefore, we carried mixed mini-batch learning forward. This better performance can be from the mixed mini-batch approach, which avoids continuously over-writing; but can also be from the min anneal sampling, which trains different task in a more balanced manner.


(a) BERT full-model + sequential learning

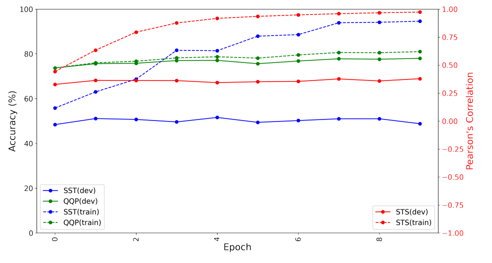

(b) BERT full-model + Mixed mini-batch learning

Figure 2: Performance metrics by Epoch for different tasks with two full model fine-tuning methods

4.4.4 Two sentences concatenated as input strongly boosts performance

Since the above model has worst performance on the STS task, we sought to improve on this task. We hypothesize that the above model we have implemented has treated pre-trained BERT model largely as a sentence meaning extractor. Therefore, we would like to test whether feeding the pair of two sentences in concatenated manner and doing downstream task with the last hidden layer embedding can increase the performance of sentence pair task. In practice, we have implemented two approaches, one simply concatenating two sentences, and another one further adding a unused token to separate two sentences and token type embedding. First of all, this is the most effective approach we implemented to improve the performance. We observe that QQP accuracy increases from 0.778 to 0.879 and STS correlation from 0.376 to 0.867, along with a slight decrease of SST from 0.51 to 0.507 (Table 3). This result indeed supports our hypothesis. Comparing the two approaches (both with further pre-trained model), we found that QQP acc. and STS corr. have slightly increase, 0.002 and 0.006 respectively, along with again a decrease of the SST task (Table 3). This result also supports our idea that more information indicating individual sentence can improve the model performance on sentence pair tasks. However, in both cases, we found that asking the model to look at or focus on two sentences may interfere the performance of one sentence task.

4.4.5 Non-linear feed-forward layers as last task-specific layers improves the performance

To further improve the performance, we hypothesize that using only one linear layer as the last task-specific layers may lead to lack of complexity and capacity. Therefore, we tested our idea by inducing non-linear neural networks, ReLU between two linear layers here. We implemented this method on top of our model with simple sentence concatenation. We found that it increase QQP acc. by 0.012 and STS corr. by 0.016 with a slight decrease of SST acc. by 0.002 (Table 3). This indicates that the model fine-tuning can benefit from more sophisticated task-specific layers.

4.4.6 Further pre-training improves the performance

Further pre-training with in-domain or in-task data has been shown to be beneficial to task-specific fine-tuning. Therefore, we further pre-trained the model on the datasets of the three tasks with MLM task, in order to allow the model better understand the meaning the task-specific content. To assess how well the model learns during the further pre-training, we track the training loss and dev loss over 40 Epochs. Results suggest that both the train and dev loss linearly decrease throughout the whole pre-training phase (Figure 3), indicating a continous learning without significant over-fitting. Furthermore, we would like to test whether pre-training with over epochs can provide more performance boost. We applied the models after 3, 18, and 40 Epochs to fine-tuning. We found that the model with 3 pre-training epochs provides a significant increase to the performance across all

Model	SST Acc	QQP Acc	STS Corr	Overall
BERT baseline (final layer)	0.391	0.655	0.262	0.559
BERT full-model + sequential learning	0.512	0.788	0.330	0.655
BERT full-model + Min anneal sampling	0.510	0.778	0.376	0.659
BERT full-model + Min anneal sampling + concatenate	0.507	0.879	0.867	0.773
BERT full-model + Min anneal sampling + concatenate + further pre-train (Epoch = 3)	0.519	0.886	0.876	0.781
BERT full-model + Min anneal sampling + concatenate + further pre-train (Epoch = 18)	0.493	0.875	0.844	0.763
BERT full-model + Min anneal sampling + concatenate + further pre-train (Epoch = 40)	0.495	0.876	0.852	0.765
BERT full-model + Min anneal sampling + concatenate + type emb + further pre-train (Epoch = 3)	0.510	0.888	0.882	0.780
BERT full-model + Min anneal sampling + concatenate + ReLU	0.505	**0.891**	**0.883**	0.779
BERT full-model + Min anneal sampling + concatenate + ReLU + type emb + further pre-train (Epoch = 3)	**0.522**	0.886	**0.883**	**0.783**
BERT full-model + Min anneal sampling + concatenate + ReLU + further pre-train (Epoch = 40)	0.492	0.883	0.855	0.768
Test set BERT full-model + Min anneal sampling + concatenate + ReLU + type emb + further pre-train (Epoch = 3)	0.528	0.886	0.875	0.783

Table 3: Performance matrics comparison of different models

three tasks, SST acc. by 0.012, QQP acc. by 0.007, and STS corr. by 0.009 (Table 3). However, unexpectedly, we found that models with 18 and 50 pre-training epochs significantly decrease the fine-tuning performance.This suggests that further pre-training can help fine-tuning when properly done.

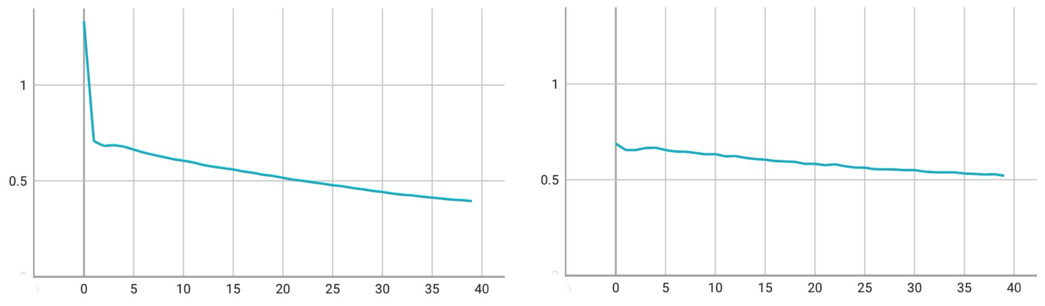


Figure 3: Further pre-training show decrease of train (left) and dev (right) loss over 40 Epochs

5 Analysis

5.1 Analysis for sentiment classification

When doing single-task fine-tuning for sentiment classification, the model performance on the CFIMDB dataset is better than SST dataset. The different characteristics and distributions of the dataset might lead to this result. This is probably because SST is 5-grain classification task while CFIMDB is a binary task. It is challenging for the model to distinguish the ambiguous classifications.

To evaluate the performance of our model on sentiment classification and check whether the error is due to the ambiguity, we generate the confusion matrix (Figure 4). The majority of the mistakes made by the model are one class away from the true value, suggesting the decent performance of the model. When comparing the error rate for different classes, neutral sentences seem to have highest error rate, potentially due to the ambiguity. Besides neutral sentences, "somewhat positive" and "somewhat negative" have more examples and higher accuracy compared with "positive" and "negative" sentences. This indicates that the imbalance of dataset may be a cause of misclassifying "positive" and "negative" sentences.

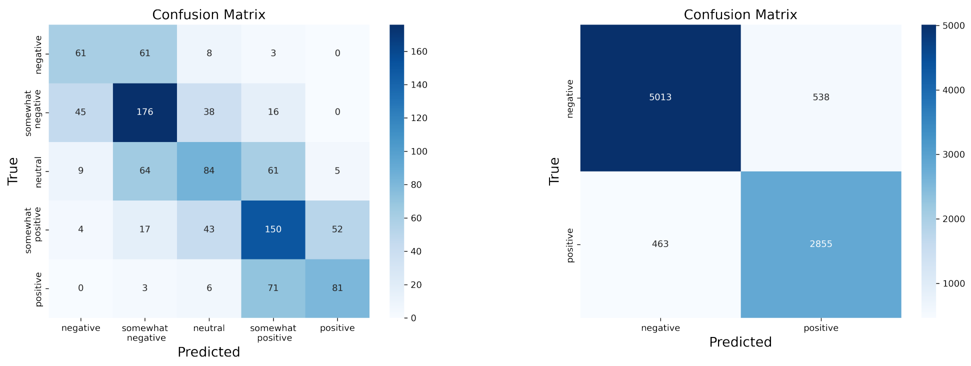


Figure 4: Confusion matrix for SST (left) and QQP (right)

5.2 Analysis for paraphrase detection

For the paraphrase detection task, the number of false-negative rate (˜ 20%, predicted negatives by all positives) is higher than false-positive rate (˜ 10%, predicted positives by all negatives) (Figure 4). Here, we observe an imbalance distribution in the dataset as there are twice more negatives (non-paraphrase sentence pairs) than positives (paraphrase sentence pairs). To better understand why the model make wrong predictions, we take a closer look at the false-positive and false-negative examples (Table 4). We observed different patterns of mistakes. Firstly, the model makes a clear misunderstanding of the sentences. When the sentence pair share similar key words, the model predict them as paraphrase even though their meaning is different (for example sentence 3). Secondly, some sentence pairs are paraphrase to some extent. When we asked different people to label the example, human opinions cannot agree with each other (data not shown). Therefore, these examples can also be a grey area for the model to predict (for example sentence 1 and 4). Thirdly, in some cases, the true label is incorrectly label (for example sentence 2). Out of these cases, only the scenario can be potentially improved with the model.

	Sentence 1	Sentence 2	True	Predict
1	What is Shaoxing vinegar?	What is Shaoxing vinegar used for?	1	0
2	Does homeopathic medicine work?	What is homeopathy? How does it work?	1	0
3	How long will Quora exist?	How long has Quora existed?	0	1
4	Who is your favorite model?	Who are your favorite models?	0	1

Table 4: Example of sentence pairs with wrong model detection

5.3 Analysis for semantic textual similarity

Here, we plotted predicted values against true values. This plot suggests that the predicted values are very close to the true values, mostly within +1 and -1 range. We further explored whether the differences between the true and predicted similarity is related to the length of the sentence pairs (Figure 5) or the sentence length differences in the pairs (data not shown). However, we don't find any obvious correlations, suggesting the model is robust to sentences of different length.

We further checked the cases with the biggest discrepancy between the true and predicted similarity (Table 5). Firstly, when one sentence has a lot more details. The model can underestimate the similarity (for example sentence 1). Secondly, when the sentence has abbreviations, the model cannot associate them with the full name (for example sentence 2). Thirdly, we found that when two

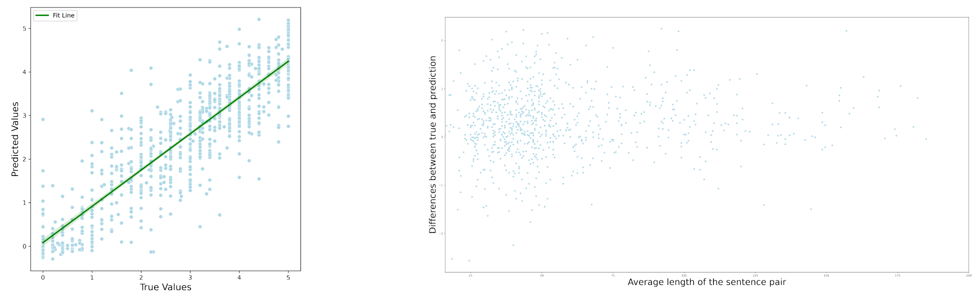


Figure 5: Analysis for the performance of SST on dev set. Left: scatter plot for true value v.s. predicted value. Right: scatter plot for (true value - predicted value) v.s. sentence length

short sentences shared a lot of key words, the model may overestimate the similarity even when the meaning is actually opposite (for example sentence 3 and 4).

	Sentence 1	Sentence 2	True	Predict
1	In these days of googling, it's sloppy to not find the source of a quotation.	I agree with Kate Sherwood, you should be able to attribute most quotes these days by simple fact checking.	3.2	1.0
2	Carney sets high bar to change at BoE	Carney sets high bar to changes at Bank of England	5	2.8
3	Work into it slowly.	It seems to work.	0	2.5
4	Indian stocks open lower	Indian stocks close lower	1.8	4.4

Table 5: Example of sentence pairs with largest difference between model prediction and true similarity

6 Conclusion

Large language model fine-tuning with pre-trained model is an effective way of tackling many natural language processing tasks. In this report, we improve the BERT-based performance for multi-task learning with various approaches.

We have come up with a "min anneal sampling" method to mix mini-batches for the multi-task fine-tuning. We found concatenating sentence pair as input is the most effective way to improve the model performance on sentence pair tasks. We reason that this approach unleash the capability of BERT model to understand the relationship between the two, instead of taking the BERT model solely as a sentence meaning extractor. In addition, we found using non-linear neural network as task-specific layer can further improve the model performance. We reason that this approach increases the complexity and capacity of the task-specific layers. We also tested whether and how further in-task pre-training can improve the performance. We show that further pre-trained model with low number of epochs can improve the downstream task performance. Combining these effective methods, we are able to achieve SST acc. of 0.528, QQP acc. of 0.886, and STS corr. of 0.875 on the unseen test dataset. We have also tried sequential learning and found the model would seriously over-fit. And we found that training loss and dev loss can decrease linearly during the pre-training throughout 40 epochs.

In the end, we analyze the model predictions in depth to understand the potential causes leading to errors. Overall, we show that the incorrect predicted values are mostly close the true value. Part of the inaccuracy can be attributed to the imbalance of the dataset. Part can be due to ambiguity of some data where different human may have different opinions. Surprisingly, we also found some incorrectly labeled data in the QQP dataset, highlighting the importance of data quality in model evaluation. In summary, we report here methods that can improve the multi- task fine-tuning with BERT and potential future directions to further improve the performance.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs.

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*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pages 194–206. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.