minBERT and Downstream Tasks

Stanford CS224N Default Project

Jingxi (Harvey) Cai
Department of Materials Science and Engineering
Stanford University
harvec@stanford.edu

Abstract

Multi-task models are more efficient than single-task models as they demand less storage, time and energy cost. Motivated by this advantage, the goal of this project is to first reproduce the minBERT model. The model will be later pre-trained, fine-tuned, and utilized to perform **Sentiment Analysis Tasks**. Further extensions were then be implemented to make the model adapted to other tasks, including **Paraphrase Detection** and **Semantic Textual Similarity**. Extensions including Round-Robin, Cosine Similarity and Smoothness-Inducing Adversarial Regularization (SIAR) were incorporated in the model to improve the performance. Optimizers besides AdamW have also been experimented to find the most promising optimization for the multi-task BERT model.

1 Key Information to include

- Mentor: Xiaoyuan Ni
- External Collaborators: N/A
- Sharing project: N/A

2 Introduction

Multi-task models are interesting because they can simultaneously learn to perform multiple related tasks. In other words, instead of training a separate model for each task, a multi-task model can learn to perform multiple tasks by sharing lower-level representations across them. This can result in more efficient and effective learning, especially when the tasks have related or overlapping features.

Multi-task learning has been shown to be effective in a variety of applications, such as natural language processing, computer vision, and speech recognition. For example, in natural language processing, a multi-task model can learn to perform tasks such as sentiment analysis, text classification, and named entity recognition simultaneously. In computer vision, a multi-task model can learn to perform tasks such as object detection, segmentation, and classification together. Furthermore, multi-task models can also help with data efficiency, as they can learn from related tasks even when there is limited data available for a particular task. Additionally, they can also lead to more generalizable models, as the shared representations learned across tasks can capture more broadly applicable features.

In this project, we will extend the minBERT model to perform multiple tasks. BERT is a highly effective model for a variety of natural language processing (NLP) tasks, such as sentiment analysis and text classification. Therefore, we hypothesize that BERT can be adapted to handle multiple downstream tasks simultaneously. After loading the pre-trained BERT model. We will fine-tune the model on three different tasks as shown in Figure 1: Sentiment Analysis Tasks, Paraphrase Detection, and Semantic Textual Similarity. As illustrated by the illustration, fine-tuning happens separately for each task, however, they share the same model parameters. Therefore, finding a proper balance amongst the three tasks will be the major challenge for this project. This process will involve finding

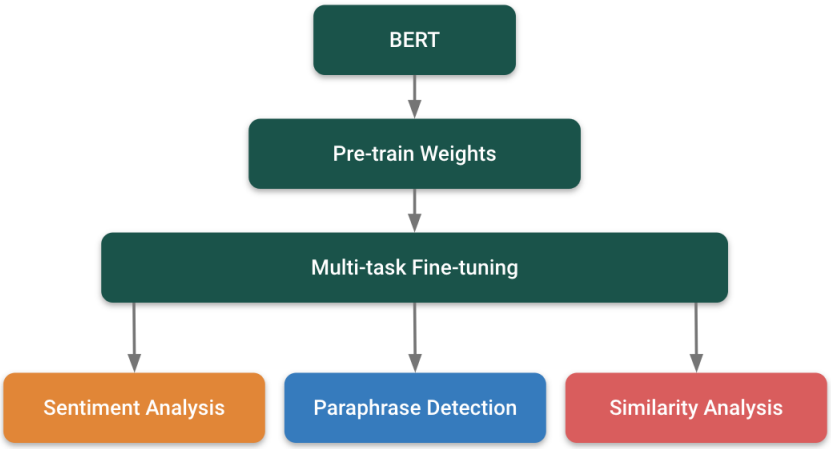

Figure 1: Multi-task BERT Model.

the proper loss functions and distributing the loss properly. Finding a proper way to process the datasets to accommodate for training will also be another focus.

3 Related Work

BERT is a relatively new NLP model that is developed by Google AI researchers Jacob Devlin and Toutanova (2018). The Bidirectional Encoder Representations from Transformers (BERT) is pre-trained on a large corpus of text using a novel technique called Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). With a transformer architecture, BERT can be later fine-tuned for various NLP tasks. After comparing BERT with several state-of-the-art models on a range of benchmark datasets. It was found that BERT achieves new state-of-the-art results on a wide range of tasks, including the GLUE benchmark, which evaluates models on various natural language understanding tasks.

Work has indeed been done on BERT to perform the tasks intended for this project, such as text classification. (Sun et al., 2019) Multi-task model is also not a new subject, as it has been around for several decades in NLP. It is no surprise that BERT has also been a candidate for multi-tasking in many research. For example, Stickland and Murray (2019) explored the potential of the projected attention layer in BERT to produce more efficient adaptation to various downstream tasks. However, little work was found on BERT model that combines the three exact tasks to be performed in this project. Hence, we can fortunately refer to the guidelines provided by the past work single-task BERT and multi-task fine-tuning to implement a more robust multi-task BERT model.

4 Approach

4.1 minBert and AdamW

The first step is to build a working minBERT model. Apart from the code already provided by the project handout, the attention and forward functions were implemented following the Multi-Head Self-Attention and Scaled Dot-Product described in ***Attention is All Your Need*** as illustrated in Figure 2. (Vaswani et al., 2017) The Adam Optimizer was implemented according to the pseudo-code in the handout. (Loshchilov and Hutter, 2017) (Kingma and Ba, 2014)

Sanity check was conducted. After that, A test training was performed on both the `Stanford Sentiment Treebank (SST) Dataset` and the `CFIMDB Dataset` described in Table 1. The results were later compared to the baseline to ensure the model was functioning correctly.

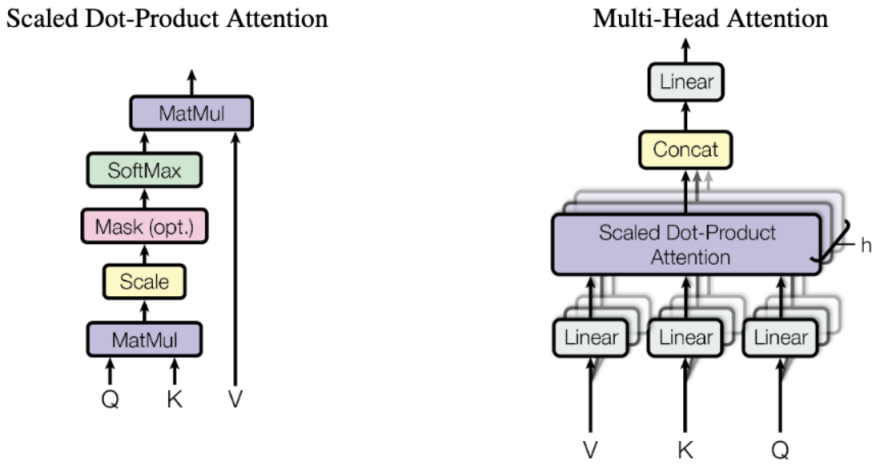

Figure 2: Scaled Dot-Product and Multi-Head Self-Attention

4.2 Round-Robin

To enable the model to perform multiple tasks, we need to train the model on multiple datasets with different inputs/outputs, specifically on three tasks, **Sentiment Analysis**, **Paraphrase Detection**, and **Semantic Textual Similarity Analysis**. Instead of using separate training for each task completely, an interweaving round-robin method was adopted. (Figure 3) For each round, we will train the model on only one batch of data for each task and repeat the process until completion. To address the difference in the number of data points across the datasets, the batch size of the largest dataset (`Quora Dataset`) was set to 64, and the batch sizes of the two other datasets were calculated accordingly so that they have the closest number of batches.

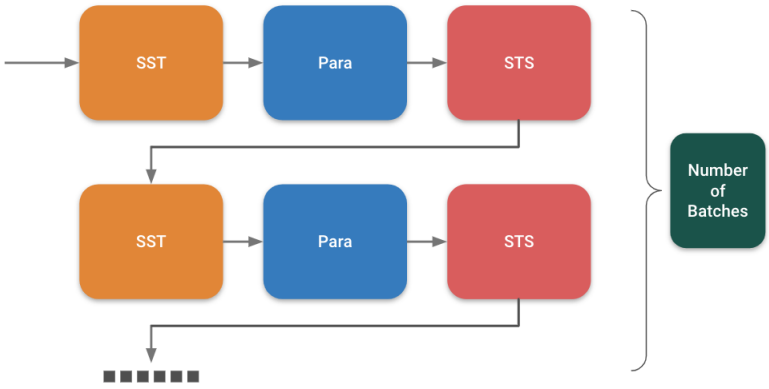

Figure 3: Round-Robin

4.3 Cosine similarity

As the PDT and STS tasks are both tasks that essentially detect the similarity between 2 sentences, during fine-tuning on the PDT and STS tasks, cosine similarity was calculated before the logits were passed to the loss functions as a experiment on the model performance. (Reimers and Gurevych, 2019)

Stanford Sentiment Treebank (SST) Dataset	
Description	consists of 11,855 single sentences from movie reviews extracted from movie reviews. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive.
Splits	Train / Dev / Test : 8,544 / 1,101 / 2,210 examples
Input	a sentence representing a movie review
Output	one of [negative, somewhat negative, neutral, somewhat positive, or positive]
CFIMDB Dataset	
Description	consists of 2,434 highly polar movie reviews. Each movie review has a binary label of negative or positive.
Splits	Train / Dev / Test : 1,701 / 245 / 488 examples
Input	a sentence representing a movie review
Output	negative, somewhat negative, neutral, somewhat positive, or positive
Quora Dataset	
Description	consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another.
Splits	Train / Dev / Test : 141,506 / 20,215 / 40,431 examples
Input	a pair of questions
Output	a boolean label representing whether they are paraphrases
SemEval STS Benchmark Dataset	
Description	consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning)
Splits	Train / Dev / Test : 6,041 / 864 / 1,726 examples
Input	a pair of sentences
Output	a scale from 0 to 5 representing whether they are similar

Table 1: Dataset Information

4.4 Regularized optimization

To aggressive updates can happen during fine-tuning, regularized optimization Jiang et al. (2019) will be applied to mitigate such overfitting. Two types of optimizations were intended to be explored, but only the SIAR was implemented due to time limitations.

1. **Smoothness-Inducing Adversarial Regularization (SIAR)**. In SIAR, an adversarial loss is added to the standard loss function of the model. This adversarial loss is designed to encourage the model to produce smooth predictions by penalizing sharp changes or discontinuities in the output. This will encourage the model to produce smooth predictions and prevent it from memorizing examples from the training datasets.
2. **Bergman Proximal Point Optimziation (BPPO)**. The purpose of BPPO is to solve a sequence of smooth convex subproblems, where each subproblem is obtained by adding a "proximal term" to the objective function. The proximal term is a function that measures the distance between the current iterate and some fixed point, which is typically the solution of a simpler problem.

5 Experiments

5.1 Data

Four datasets are being used for this project. The SST and CFIMDB datasets are associated with Sentiment Analysis. the CFIMDB will not be used in the training of the multi-task classifier. The

Quora dataset is associated with Paraphrase Detection. The SemEval STS Benchmark dataset is associated with Semantic Textual Similarity Analysis. Dataset details can be found in Table 1.

5.2 Evaluation method

Since we are training a multitask model, we have different evaluation metrics for separate tasks. For the **Sentiment Analysis** and **Paraphrase Detection**, we will use accuracy as the metrics as the sentiment classes and the paraphrase classes/labels are very well defined. For the **Semantic Textual Similarity Analysis**, we will calculate the Pearson correlation of the true similarity values against the predicted similarity values across the test dataset.

5.3 Experimental details

Three datasets were divided into batches of different sizes to follow the Round-Robin approach. The batch size of the largest dataset (`Quora Dataset`) was set to 64, which is the largest to fit the memory storage. The `STT Dataset`, the `Quora Dataset` and the `STS Dataset` were read with 2136, 2211, and 2014 batches. The hidden size was kept at 768. Learning rate of the pre-train model was set to 10^{-3}, and the learning rate of the fine-tune model was set to 10^{-5}. All the training was done in AWS EC2.

5.4 Results

5.4.1 Model Setup Experiment

The optimizer used across this experiment is the default AdamW. As shown by Table 2, all the pre-train and fine-tune models outperform the baseline. Moreover, the fine-tune models all have better performance than the pre-train model regardless of its model setup. It is also evident that both the Cosine Similarity and the SIAR improve the overall performance of BERT.

Model Setup, Details and Extensions Used	**SST**	**Para**	**STS**	**Overall**
Baseline	0.515	0.597	-0.057	1.055
Pretrain (`lr = 1e-3`)	0.502	0.525	0.378	1.405
Fintune (`lr = 1e-5`)				
`Basic`	0.497	0.527	0.667	1.609
`Cosine Similarity`	0.506	0.768	0.690	1.964
`Cosine Similarity + SIAR(lambda_tv = 0.01)`	0.524	0.787	0.682	**1.993**

Table 2: Performances of BERT Experimented

5.4.2 Optimizer Experiment

Due to time limit, only the training result of the first epoch was considered in the comparison of different optimizers. Among the optimizers experimented, the self-implemented **AdamW** has the best performance. **LAMB** is very close to **AdamW** with an arbitrarily chosen learning rate.

Optimizer and Learning Rate	**SST**	**Para**	**STS**	**Overall**
`AdamW(lr = 1e-5)`	0.455	0.716	0.617	**1.788**
`Adamax(lr = 1e-5)`	0.346	0.685	0.242	1.273
`QHAdam(lr = 1e-5)`	0.448	0.709	0.541	1.698
`LAMB(lr = 1e-4)`	0.436	0.713	0.558	1.707
`LAMB(lr = 1e-5)`				1.213

Table 3: First Epoch Performance of Various Optimizers

6 Analysis

We can see an obvious boost in the model performance as we adjusted the model setup in the Paraphrase Detection and Semantic Textual Similarity Analysis. However, the improvement of Sentiment Analysis is rather limited and usually capped at around 0.5, which makes sense as there are 5 discrete labels and we are evaluating its performance simply by accuracy. It is possible that the model's prediction gets closer to the intended sentiment as the training goes on, but it is not reflected in our metrics.

By Inspecting the typical training progress, we can see that the gap between the training performance and dev performance grows as training progresses (Figure 4). It is highly likely that the model is overfitting on the training data, especially for SST and STS. By applying the SIAR loss, which was intended to prevent overfitting, there has indeed been a boost for SST, but the correlation of STS seemed to decrease. As the training process only saves the highest overall-performing model, it is possible that the STS performance is relatively low in this epoch, as in some other epochs, a correlation over 0.71 in STS has been observed. Such difference demonstrated that SIAR is effective in preventing overfitting.

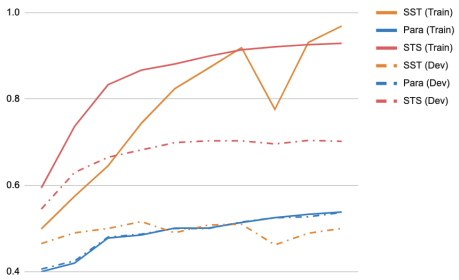

Figure 4: Example Training Progress

There is a drastic enhancement in the performance of Para after the Cosine Similarity was applied, which is odd as the performance of STS only increased by about 0.2. This mistake was likely made due to the wrong logits and loss function conversion at the start of training (Basic).

The reason why the default AdamW has the best performance amongst all the optimizers can be attributed to the learning rate, which is suggested by the project handout. The optimal learning rates for the other optimizers on the other hands, need further inspection to be determined. For example, a small tweak in the learning rate of LAMB optimizer from `1e-4` to `1e-5` improves the overall performance by a large amount. Therefore, it is possible that with a proper learning rate, the other optimizers can outperform the AdamW optimizer.

7 Conclusion

At this point, we can conclude that we have a working multi-task BERT model with acceptable performance and BERT is indeed a promising candidate for handling multiple NLP tasks at the same time. Cosine Similarity and Smoothness-Inducing Adversarial Regularization are both effective extensions that visibly improve the model performance with the self-implemented AdamW optimizer. However, there is still room for improvements. As there are still plenty of unimplemented ideas due to time limits, potential future work can be done:

- **Contrastive Learning**. Used to similarities and differences between pairs of data samples. Training the model to maximize the similarity between positive pairs (pairs of samples that are similar) and minimize the similarity between negative pairs (pairs of samples that are dissimilar).
- **Bergman Proximal Point Optimziation**. Similar to Smoothness-Inducing Adversarial Regularization, used to smooth the model and prevent overfitting.
- **Optimizers and Parameters**. There is a wide range of optimizers/loss functions/learning rates and possibly other parameters that can be tested to optimize the performance.

References

Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

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*, volume 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. In *Advances in Neural Information Processing Systems*, pages 5998–6008.