Multi-task Learning with BERT in NLP

Stanford CS224N Default Project

Fan Wang
Department of Computer Science
Stanford University
wang420@stanford.edu

Abstract

In natural language processing, while deep learning techniques have achieved remarkable success in many different problems, these models are mostly task specific. When facing a situation in which multiple tasks are solved at the same time, multi-task learning is a popular paradigm to sufficiently utilize information across different tasks and to learn more efficiently or effectively. With the adoption of large language models like BERT in NLP, however, there is little research to enhance BERT to improve the performance on additional target tasks further. The downstream tasks we are interested in are Sentiment Analysis, Paraphrase Detection and Semantic Textual Similarity. In this project, we propose a multi-task learning framework based on BERT, then experiment extensions to improve the performance on the 3 tasks simultaneously. We have explored models with different architectures, different loss functions, different optimizers, adversarial learning and contrastive learning.

1 Key Information to include

- Mentor:
- External Collaborators (if you have any):
- Sharing project:

2 Introduction

Text classification as a classical problem in NLP has been studied extensively by various neural network techniques, such as convolution model (Kalchbrenner et al. (2014), Zhang et al. (2015), Conneau et al. (2016), Johnson and Zhang (2017), Zhang et al. (2017), Shen et al. (2018)), recurrent models (Liu et al. (2019), Dani et al. (2017), Seo et al. (2017)) and attention mechanisms (Yang et al. (2016), Lin et al. (2017)). On the other hand, recently, pre-trained models on large corpus are are shown that are beneficial for text classification and other NLP tasks. As a state-of-the-art language pre-trained model, BERT(Devlin et al. (2018)) has achieved amazing results in many natural language understanding (NLU) tasks, however, there is little research to enhance BERT to improve the performance on target tasks further(Sun et al. (2020)).

In this research report, we investigate how to maximize the utilization of BERT for three downstream tasks simultaneously. The tasks we have studied are Sentiment Analysis, Paraphrase Detection and Semantic Textual Similarity. We explore several extensions of fine-tuning BERT to enhance its performance. We design exhaustive experiments to make a detailed analysis of different extensions. The extensions we experiments are the following:

- Model architecture and number of layers
- Multi-task learning and training tactics
- Adversarial learning

- Contrastive learning

Our first extension is Sentence-BERT (Reimers and Gurevych (2019)), a modification of the BERT network using siamese networks which takes a pair of sentences as input to derive semantically meaningful sentence embeddings. The next step is to apply our Sentencet-BERT model in multi-task setting which presents a number of optimization challenges due to the complicated shape of multiple loss functions, making it difficult to optimize and learn efficiently compared to single task problems. To overcome, we study the round-robin strategy, training on the total loss function instead of individual loss functions, and the gradient surgery strategy (Yu et al. (2020)). One of the issues in aggressive fine-tuning is over-fitting. To alleviate the problem, we have explored adversarial learning (Goodfellow et al. (2015)) as the regularization method. The last extension we have studied is contrastive learning, a simple framework which uses entailment pairs as positives and contraction pairs (Gao et al. (2021), Jiang et al. (2022)) as hard negatives.

3 Related Work

We first introduce BERT, then, we briefly review the information related to extensions that we have applied in fine-tuning the BERT model on our multi-task problem.

BERT (Devlin et al. (2018)) is a pre-trained large language model based on the transformer (Vaswani et al. (2017)) structure. It sets for various NLP tasks the new state-of-the-art results, including question answering, sentence classification, and sentence-pair regression. To apply BERT in sentence-pair regression, the input consists of the two sentences, separated by a special [SEP] token. Multi-head attention algorithm is applied and the output is passed to another activation function or a simple regression function is derive the final prediction for backpropagation. A drawback of this input structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT.

3.1 Sentence-BERT

The Sentence-BERT (Reimers and Gurevych (2019)) method adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding. Then use a siamese network, the sentence embedding generated by BERT are semantically meaningful and can be compared with a similarity measure, such like cosine-similarity or Euclidean distance. In the siamese network, the two BERT models have tied weights. The structure is illustrated in figure1.

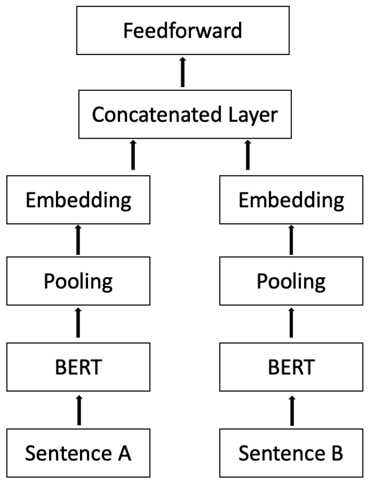


Figure 1: SBERT architecture with siamese network structure

3.2 Gradient surgery

While the BERT model and deep learning techniques in general have shown remarkable promise in NLP. The tasks are usually learned individually. Sometimes data availability / efficiency is a challenge. Multi-task learning has emerged as the approach to learn more efficiently and shared knowledge across different tasks. However, multi-Task learning can be very challenging when gradients of different tasks are of severely different magnitudes or point into conflicting directions. Gradient surgery method (Yu et al. (2020)) is a simple yet general approach for avoiding the interference between gradients of different loss functions. As a model-agnostic approach, the method involves projecting conflicting gradients (PCGrad) and will retain optimality guarantees.

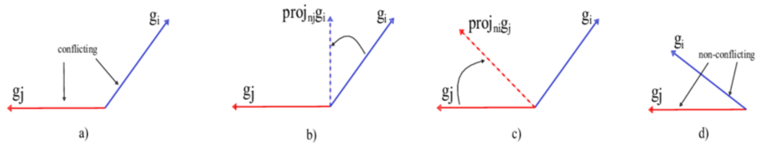


Figure 2: Conflicting gradient and PCGrad

3.3 Adversarial learning

Aggressive fine-tuning in multi-task setting can cause over-fitting. Adversarial training provides a meaningful way of regularizing supervised learning algorithms to combat the problem. Among the many available adversarial techniques, FGSM (Fast Gradient Sign Method), FGM (Fast Gradient Method), and PGD (Projected Gradient Descent) are the three related to this project. FGSM (Goodfellow et al. (2015)) linearizes the cost function around the current value of parameters, obtaining an optimal max-norm constrained pertubation, which can be computed efficiently by backpropagation. FGM (Miyato et al. (2016)) takes one step further by removing the sign function used in FGSM and normarlizes the gradient of loss function. One can intepret that FGM and FGSM as a simple one-step scheme to solve the inner maximization problem. A more powerful adversary is the multi-step variant, which is essentially PGD (Madry et al. (2017)) on the loss function.

3.4 Contrastive learning

The idea of contrastive learning is originally from computer vision. In the paper(Chen et al. (2020)), the authors show that by combining the right data augmentations and the normalized temperature-scaled cross entropy loss function, the model achieves a new state-of-the-art performance. The key idea is to to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors. The application of contrastive learning in NLP is summerized in SimCSE, Simple Contrastive Learning of Sentence Embeddings (Gao et al. (2021)) and PromCSE, Prompt-based Contrstive Learning for Sentence Embedding (Jiang et al. (2022)). Supervised SimCSE uses labels from the training data into contrastive learning and PromCSE connects contrastive learning with energy-based learning. Both approaches have shown improvements over the previous best results.

4 Approach

We start from the BERT model as the bottom and build three towers on top for the multiple tasks. The BERT layers across different tasks share the same weights. We have experimented a structure in which the weights of pooling layers between taks 2 and task 3 are shared, it yields suboptimal performance. Thus we have decided not to share any additional layers beside the BERT layer. The overall network strcture is depicted in figure3. As for the baseline, we use the vanilla BERT fine-tuned on task 1 only and evaluate the model on all 3 tasks.

4.1 Concatenated layer and feedforward network

There is no concatenated layer in task 1 since only one sentence is used as the input, we use a dropout layer and a linear(BERT_Hidden_Size, Number_of_Labels) layer as the feedforward network. In task 2 and task 3, we have explored the following three options shown in figure 44.

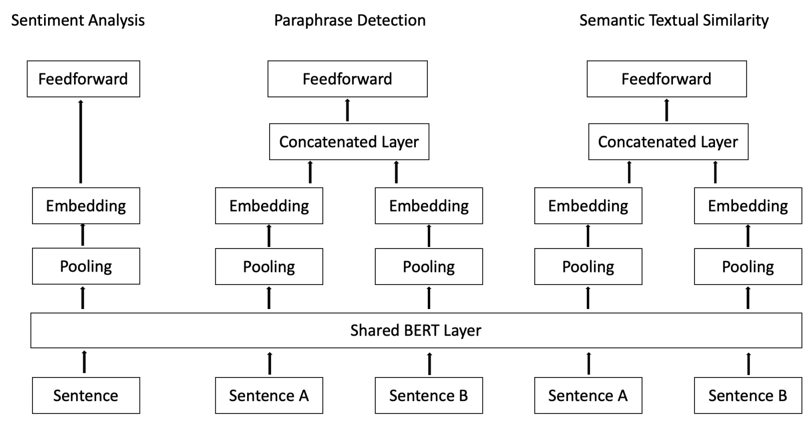
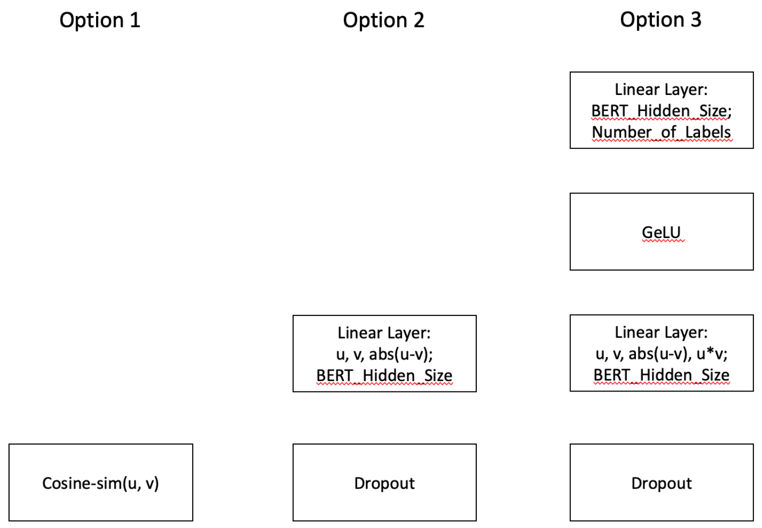


Figure 3: Multi-task network structure



Option 1
Option 2
Option 3
Linear Layer: BERT_Hidden_Size; Number_of_Labels
GeLU
Linear Layer: u, v, abs(u-v); BERT_Hidden_Size
Linear Layer: u, v, abs(u-v), u*v; BERT_Hidden_Size
Cosine-sim(u, v)
Dropout
Dropout

Figure 4: Concatenated layer and feedforward network

4.2 Gradient surgery, round-robin and total loss

To train the multi-task model efficiently, we use three methods. The gradient surgery method is introduced in paper (Yu et al. (2020)); we use an off-the-shelf implementation of the algorithm from pytorch-optimizer package in github [1]. The full update procedure is described in table1 below:

Gradient Surgery Algorithm
1. Compute number of batches for each tasks
2. While All(number of batches) > 0 do:
3. calculate loss of task 1, update batch counts
4. calculate loss of task 2, update batch counts
5. calculate loss of task 3, update batch counts
6. apply PCGrad and backpropagate

Table 1: Gradient surgery algorithm

[1]https://github.com/jettify/pytorch-optimizer

The round-robin and the total loss methods are relatively straightforward compared to the gradient surgery approach, with the algorithms listed in table2 below. We implement the logic ourselves.

	Round-robin	Total Loss
1	Compute number of batches for each tasks	Compute number of batches for each tasks
2	While sum(number of batches) > 0 do:	While All(number of batches) > 0 do:
3	calculate loss of task 1, if loss, backpropagate	calculate loss of task 1, update batch counts
4	calculate loss of task 2, if loss, backpropagate	calculate loss of task 2, update batch counts
5	calculate loss of task 3, if loss, backpropagate	calculate loss of task 3, update batch counts
6	update batch counts	calculate the sum of loss, backpropagate

Table 2: Round-robin and total loss algorithms

4.3 Adversarial learning

The code to implement the 3 adversarial methods, FGSM (Goodfellow et al. (2015)), FGM (Miyato et al. (2016)) and PGD (Madry et al. (2017)) is from github[2]. We list the key functions of each method below:

- FGSM: $x_{adv} = x + \epsilon * sign(\nabla_x Loss)$
- FGM: $x_{adv} = x + \epsilon * \nabla_x Loss / L_{2norm}(\nabla_x Loss)$
- PGD: $x_{adv} = \prod_{x+S} x + \alpha * sign(\nabla_x Loss)$

4.4 Contrastive learning

We view contrastive learning as a data augmentation method. To set up the contrastive learning with our labeled dataset, we implement the logic below ourselves. The current logic is based on a double for-loop, and thus is very slow in running time. We note the vectorization of the logic as a TODO for improvement. We have applied this logic in calculating the loss function in task 2.

```
Contrastive Learning
1. While number of batches > 0 do:
2.     calculate logits and extract label from the batch input data
3.     do i from 1 to batch size:
4.         do j from 1 to batch size:
5.             if i != j:
6.                 calculate logits based on input index i and input index j
7.                 set label to 0
8.                 concatenate logits from step 2 and logits from step 6
9.                 concatenate label from step 2 and label from step 7
10.    calculate the updated loss function
```

Table 3: Contrastive learning

5 Experiments

5.1 Data

We use Stanford Sentiment Treebank Dataset for Task 1: Sentiment Analysis; Quora Dataset for Task 2: Paraphrase Detection; and SemEval STS Benchmark Dataset for Task 3: Semantic Textual Similarity.

Stanford Sentiment Treebank Dataset The Stanford Sentiment Treebank consists of 11,855 single sentences extracted from movie reviews. Each phrase has a label if negative, somewhat negative, neutral, somewhat positive, or positives. We the following splits:

[2]https://github.com/xiaopp123/adversarial_train

- train (8,544 samples)
- dev (1,101 samples)
- test (2,210 samples)

Quora Dataset The quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. A subset of this dataset is provided with the following splits:

- train (141,506 samples)
- dev (20,215 samples)
- test (40,431 samples)

SemEval STS Benchmark Dataset The STSB dataset consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). The splits are:

- train (6,041 samples)
- dev (864 samples)
- test (1,716 samples)

5.2 Evaluation method

Accuracy is used as the metric in the tasks of sentiment analysis and paraphrase detection. Pearson correlation coefficient is used in the task of semantic textual similarity.

5.3 Experimental details

Our BERT model backbone is the provided default one: "bert-base-uncased", together with the default learning rate (1e-5). We have explored different learning rate (2e-5 and 3e-5), it doesn't seem to have a big impact on the performance. We use the Adam optimizer in all experiments. The batch size are modified to fully use the available GPU and to ensure the number of batches in each task is roughly the same due to the imbalanced training data.

The table below summarizes the details of experiments we have performed. The option 2 and option 3 in model structure column are defined in figure 44 of section 4.1. We have also explored models with option 1 as the structure, but their performance are usually worse than the other two options. We feel that the two terms, abs(u-v) and u*v, are sufficient if not have more information than a simple cosine-similarity term. Thus we haven't combined option 1 with other extensions in our experiments. In addition, we have observed a 10% improvement in performance after switching from option 2 to option 3. Therefore, option 3 is selected.

Number	Batch Size	Model Structure	Multi-task Loss	Adversarial	Contrastive
1	8/64/8	option 2	round-robin		
2	8/64/8	option 3	round-robin		
3	4/64/2	option 3	total loss		
4	4/64/2	option 3	gradient surgery		
5	4/64/2	option 3	round-robin	FGM	
6	4/64/2	option 3	round-robin	FGSM	
7	4/64/2	option 3	round-robin	PGD	
8	4/64/2	option 3	round-robin		Contrastive

Table 4: Experimental details

5.4 Results

Our baseline is the model fine-tuned only on SST dataset. We note the baseline model as experiment 0 and report its performance together with other experiments described in section 5.3 in the table5.

Description	sst-dev	quora-dev	sts-dev	overall average
experiment 0	0.529	0.453	0.018	0.333
experiment 1	0.496	0.844	0.623	0.654
experiment 2	0.502	0.869	0.765	0.712
experiment 3	0.500	0.856	0.786	0.714
experiment 4	0.495	0.864	0.789	0.718
experiment 5	0.500	0.856	0.806	0.721
experiment 6	0.469	0.870	0.791	0.710
experiment 7	0.509	0.845	0.792	0.715
experiment 8	0.393	0.843	0.575	0.603

Table 5: Experimental results

Due to the limited numbers of submissions allowed to test leaderboard (i.e., 3 times in total), we only report the model performance on development data.

The model submitted to leaderboard is from experiment 5, with overall performance metric 0.717 in the test leaderboard and performance in each of the three tasks:

- SST test Accuracy: 0.498
- Paraphrase test Accuracy: 0.853
- STS test Correlation: 0.799

In experiment 8, the model is only trained with 1 epoch and takes 6 hours to finish due to the double for-loop logic. Other than that, all our experiments have a better performance than the baseline model. Using more layers results into a 10% increase starting from experiment 3. However, the overall performance hits a plateau in the (0.71, 0.72) range. The multiple extensions we have tried don't improve the performance much.

Our model performs best in the Paraphrase Detection task since the we have the most samples in the Quora dataset, around 144K. The performance in the Semantic Textual Similarity task is the second best one since the input data to Paraphrase Detection and Semantic Textual Similarity are both sentence pairs. The model performs worst in the Sentiment Analysis task, only around 0.5. This might not be a surprise since the input data is single sentence.

6 Analysis

Without adding additional data into our training sample or ensembling with another model, we think the peak performance is 0.5 for Sentiment Analysis task, 0.87 for Paraphrase Detection task and 0.80 for Semantic Textual Similarity task. Due to the high imbalance between training samples (task 1: 8K, task 2: 144K and task 3: 6K), the knowledge extracted from task 2 dominates our multi-task models. Various extensions do not have a huge impact on the performance. Though it would be interested to know the result if we are able to run experiment 8 for full 10 epochs.

7 Conclusion

In this research project, we have fine-tuned the BERT model in a multi-tasks setting. We think that adding another Sentiment Analysis alike data will help to improve the model performance on task 1 and adding another Semantic Textual Similarity alike data will help to improve the model performance on task 3. In addition, we would like to fix the contrastive learning logic to observe its impact and include another model with a different pre-trained BERT model as the backbone for the next step.

References

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *arxiv*.

Alexis Conneau, Holger Schwenk, Loıc Barrault, and Yann Lecun. 2016. Very deep convolutional net- works for natural language processing. In *arXiv*.

Yogatama Dani, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text clas- sification with recurrent neural networks. In *arxiv*.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *arxiv*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *arxiv*.

Yuxin Jiang, Linhan Zhang, and Wei Wang. 2022. Improved universal sentence embeddings with prompt-based contrastive learning and energy-based learning. In *arxiv*.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 562–570.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural net- work for modelling sentences. In *arxiv*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *arxiv*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *arxiv*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attack. In *arxiv*.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. In *arxiv*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *arxiv*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Neural speed reading via skim-rnn. In *arxiv*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2018. Deconvolutional latent-variable model for text sequence matching. In *In Thirty-Second AAAI Conference on Artificial Intelligence*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification? In *arxiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *arxiv*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *arxiv*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *In Advances in neural information pro- cessing systems*, page 649–657.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *In Advances in Neural Information Processing Systems*, pages 4169–4179.