minBERT and Downstream Tasks Final Report

Stanford CS224N Default Project

Bingqing Zu
Stanford University
`zubq@stanford.edu`

Yixuan Lin
Stanford University
`yixuanli@stanford.edu`

Abstract

This project aims to implement some key aspects of the original minBERT model and extend the model to perform three downstream tasks spontaneously, sentiment analysis, paraphrase detection and semantic textual similarity. The main goal is to explore methods to improve the model performance on the three tasks. We use mixing dataset and train the three tasks spontaneously. We designed a multi head prediction layer for each task on top of the base BERT model and experimented with taking different hidden states as downstream inputs. We also used cosine similarity loss for text similarity task which shows great improvement. To balance the optimization process among three different tasks, we implemented gradient surgery approach which also shows better results. To enable the generalization ability and prevent from overfitting we also added SMART regularization method. Additionally, we experimented with different training techniques such as dropout, weight decay, learning rate decay etc. Our model evaluated on dev set can get overall accuracy 0.746 and 0.520 on sentiment classification, 0.782 on paraphrase detection and 0.873 on textual similarity.

1 Key Information to include

- Mentor: David Lim
- External Collaborators, Sharing project: NA
- Team contributions: Bingqing and Yixuan both contribute on research new methods, code implementation, training model and writing report.

2 Introduction

The Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019) obtains new state-of-the-art results on many natural language processing tasks. It is powerful and simple to fine-tune the pre-trained BERT model with just one additional output layer to create models reaches state-of-the-art performance on a wide range of tasks. However, using the BERT model to perform multiple downstream tasks spontaneously is still under exploration. This project aims to study the BERT model, as well as exploring methods to improve the model performance on three downstream tasks spontaneously, specifically sentiment analysis, paraphrase detection and semantic textual similarity.

In the first part of the project, we complete the implementation of the BERT model, in order to study the core architecture of BERT. Adam optimizer with weight decay (Ilya and Frank, 2019) is implemented as the model optimizer. The model is trained, fine-tune and evaluate for sentiment analysis on the Stanford Sentiment Treeback dataset (Socher et al., 2013) and CFIMDB dataset.

In the second part of the project, we explore multiple methods to improve the model performance on multi-task learning. Even though the fine-tuned BERT model performs well on sentiment analysis task, it is time and resources consuming to fine-tune multiple BERT models for each domain-specific downstream task. Therefore, enabling the model to perform multiple tasks spontaneously becomes essential. Inspired by Xiaodong et al. (2019), we utilize the architecture of shared BERT model wights across all tasks and one task-specific layer on the top for each task. The order of training on three datasets for three tasks is experimented at first. The result indicates that training on shuffled mixing dataset performs much better than training three tasks sequentially. Due to gradient interference between tasks, we adapt gradient surgery method presented in paper Tianhe et al. (2020) to solve conflicting gradient. To further improve the model performance on semantic related tasks, we also explore the MEAN pooling strategy, which uses the mean of all output vectors instead of output of the CLS-token, as well as cosine-similarity for computing similarity (**?**). Additionally, we adjust some parameters such as dropout rate, weight decay and SMART regulation Jiang et al. (2020) for further enhancement.

3 Related Work

The BERT model from Devlin et al. (2019) achieves state-of-the-art results on many natural language processing (NLP) tasks. It is pretrained on large textual content using Masked Language Modeling and Next Sentence Prediction method, and provides a powerful base model for fine-tune on wide range of NLP tasks.

To optimize the model performance for multi-task learning, Xiaodong et al. (2019) proposes a baseline architecture using shared BERT model layer and task-specific predicition heads, which enables the multi-task capability of the BERT model. In addition, Qiwei et al. (2022) uses a auxiliary-main task architecture *MTRec* that utilizes multi-field relating information to help train on a main task. In order to solve the gradient conflicting issue between multiple tasks, Tianhe et al. (2020) provides a gradient surgery technique to solve gradient conflicting and boost multi-task model performance.

For exploring further improvement, Reimers and Gurevych (2019) proposes MEAN pooling strategy, which comprehend semantic context better then a single CLS-token, as well as using cosine-similarity score for semantic similarity task, which enhances our model performance on similarity task. In terms of regulation to prevent over-fitting, Ilya and Frank (2019) proposes a weight decay approach based on Adam optimization, and Jiang et al. (2020) introduces the smoothness-inducing adversarial regulation approach.

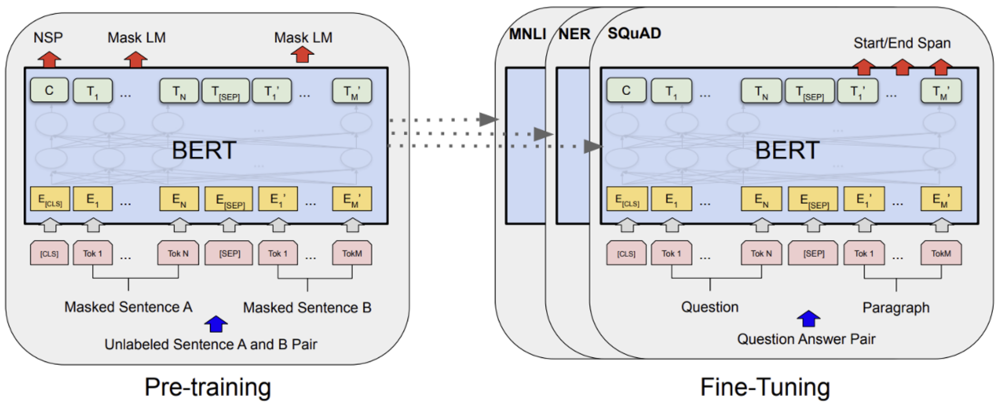


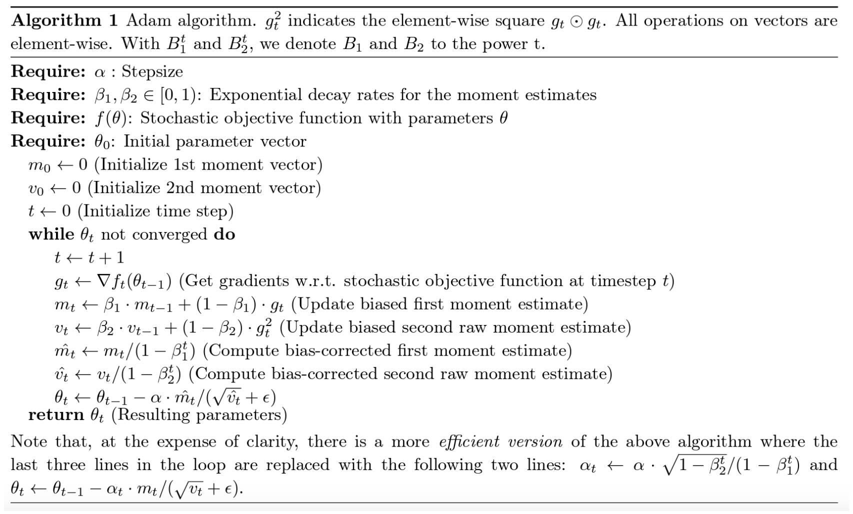
Figure 1: Pretrain and Fine Tune Process (Devlin et al., 2019)

4 Approach

4.1 Model Completion - Part 1

After downloading the base project from Dai (2024), we complete some key parts implementation of the BERT model, including optimizer, embedding layer and transformer layer. In our project, we load the weights from the pretrained BERT model with masked language modeling and next sentence prediction, and perform further fine-tune on given datasets, as the figure shown in 1.

We use Adam Optimizer based on Decoupled Weight Decay Regularization (Ilya and Frank, 2019) for model optimizer, shown in 2. The weight decay approach is decoupled from the learning rate, which improves the performance of popular Adam optimization.



Algorithm 1 Adam algorithm. g_t^2 indicates the element-wise square $g_t \odot g_t$. All operations on vectors are element-wise. With B_1^t and B_2^t, we denote B_1 and B_2 to the power t.

```
Require: $\alpha$ : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize time step)
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective function at timestep $t$)
      $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
      $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
      $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
      $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
      $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$
  return $\theta_t$ (Resulting parameters)
```

Note that, at the expense of clarity, there is a more *efficient version* of the above algorithm where the last three lines in the loop are replaced with the following two lines: $\alpha_t \leftarrow \alpha \cdot \sqrt{1 - \beta_2^t}/(1 - \beta_1^t)$ and $\theta_t \leftarrow \theta_{t-1} - \alpha_t \cdot m_t/(\sqrt{v_t} + \epsilon)$.

Figure 2: BERT Embedding Layer (Ilya and Frank, 2019)

4.1.1 Model Architecture

Based on paper Devlin et al. (2019), we implement the embedding layer which is the sum of token embedding, segment embedding and position embedding, shown in Figure 3.

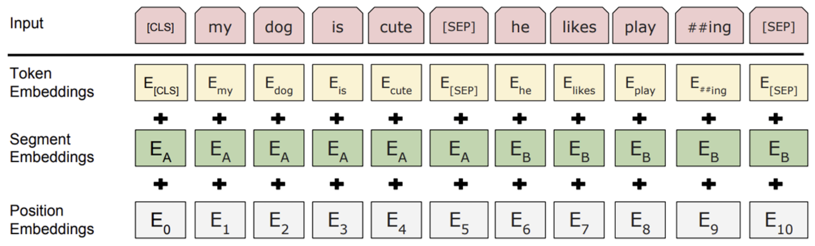

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3: BERT Embedding Layer (Devlin et al., 2019)

The BERT model includes 12 encoder transformer layers (Ashish et al., 2017) consists a multi-head self-attention and a position-wise fully connected feed-forward layer with a additive and normalization layer, shown in Figure 4.

Multi-head self-attention Multi-head self-attention uses multiple scaled dot-product attention to capture input semantic information from different representation subspaces, shown in Figure 4. Mathematically, the scaled dot-product attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

where Q is queries matrix, K and V represents keys and values matrix. d_k is the each key's dimension. With multi-head attention joins multi scaled dot-project attention together, is computed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^o$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_I^K, VW_i^V)$$

Where W_i^Q, W_i^K, W_i^V and W^o are parameter matrices.

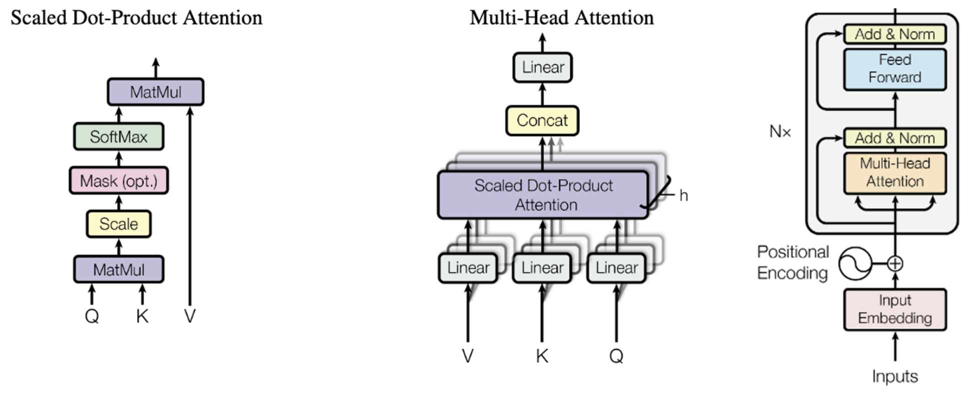


Figure 4: Multi-Head Attention (Ashish et al., 2017) (left), Encoder Transformer Layer (Ashish et al., 2017) (right)

4.2 Improvement Exploration - Part 2

Based on the BERT model implemented in Part 1, we extend it to perform three downstream tasks, sentiment analysis, paraphrase detection and semantic textual similarity, spontaneously. The following section list our baseline model as well as extension methods that we experiment to optimize the performance for multi-task learning.

4.2.1 Multi-task model Architecture

To perform three different task simultaneously, we use one prediction head for each of the task on top of the completed base BERT model in Part 1, inspired by Xiaodong et al. (2019). On top of the BERT model, we use a linear layer to project the pooled output from BERT model to the three task outputs with hidden size setting to 768. Since the input of paraphrase detection and semantic similarity tasks is a pair of sentence, the pooled output of two sentences are concatenated before projection. The output is 5 sentiment classes for sentiment analysis, a binary classification for paraphrase detection and a similarity level for semantic similarity. The architecture is shown in Figure 5.

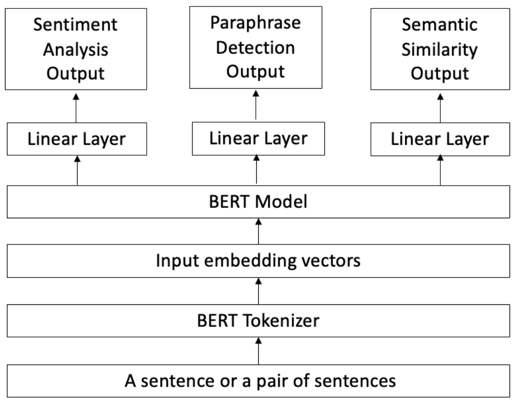

Sentiment Analysis Output
Paraphrase Detection Output
Semantic Similarity Output
Linear Layer
Linear Layer
Linear Layer
BERT Model
Input embedding vectors
BERT Tokenizer
A sentence or a pair of sentences

Figure 5: Multi-task Architecture

4.2.2 Multi-task Training

The input sentence (for sentiment analysis task) or a pair of input sentences (for paraphrase detection and semantic textual similarity tasks) is embedded using pretrained BERT Tokenizer "bert-base-uncased" downloaded from HuggingFace (Hug). We use cross entropy to get the sentiment classification loss, binary cross entropy loss for paraphrase detection task, and mean square loss for semantic similarity task. We also experimented with training three datasets sequentially and together at once, and training on three tasks together shows much better results.

Pooling Strategy In addition to use CLS-token as the pooled output from the BERT model, we explore using the mean of all output vectors, inspired by paper Reimers and Gurevych (2019). The MEAN pooling strategy provides significant improvement on semantic task, indicating using mean of the full sentence comprehends the semantic meaning better than just using a single final token.

Cosine-Similarity Inspired by paper Reimers and Gurevych (2019), in addition to the approach of classification problem on similarity from two sentences, we also explored the approach of taking cosine similarity loss from two embeddings when fine-tune on semantic similarity task. Sentences that are equivalent have a cosine-similarity of 1 and unrelated sentences get a 0 similarity score. We scale the cosine-similarity to range [0, 5] and using mean square loss between the scaled cosine similarity value and similarity ground truth labels.

Gradient Surgery Since multi-task training has been a challenging optimization problem comparing to learning each task individually. To improve the performance among multi-task learning, we also implemented the gradient surgery method to mitigate the conflict of gradients from different tasks. As described in Tianhe et al. (2020), the main idea is that if two gradients are conflicting, we alter the gradients by projecting each onto the normal plane of the other, preventing the interfering components of the gradient from being applied to the network.

We adapt the PCGrad code from Github Tseng (2020) into our model, where gradient surgery and its wrapper functions are implemented. For each epoch, we loop through each batch of the combined dataset and sum the loss for all the tasks in the batch. Gradient surgery is then applied through back propagation using AdaW optimizer. The pseudocode is shown in 6, where g_i represents gradient for task i.

```
Algorithm 1: PCGrad
Input: Model parameters $\theta$, task minibatch $\mathcal{B} = \{T_k\}$
1  $\boldsymbol{g}_k \leftarrow \nabla_\theta \mathcal{L}_k(\theta)\ \forall k$
2  $\boldsymbol{g}_k^{PC} \leftarrow \boldsymbol{g}_k\ \forall k$
3  for $T_i \in B$ do
4      for $T_j \overset{uniformly}{\sim} \mathcal{B} \backslash \mathcal{T}_i$ in random order do
5          if $\boldsymbol{g}_i^{PC} \cdot \boldsymbol{g}_j < 0$ then
6              //Substract the projection of $\boldsymbol{g}_i^{PC}$ onto $\boldsymbol{g}_j$
7              Set $\boldsymbol{g}_i^{PC} = \boldsymbol{g}_i^{PC} - \frac{\boldsymbol{g}_i^{PC} \cdot \boldsymbol{g}_j}{||\boldsymbol{g}_j||^2} \boldsymbol{g}_j$
8          end
9      end
10 end
11 return update $\Delta\theta = \boldsymbol{g}^{PC} = \Sigma_i \boldsymbol{g}_i^{PC}$
```

Figure 6: Gradient Surgery (Tseng, 2020)

SMART Regularization Inspired by Jiang et al. (2020), we also add a regularization loss to the original loss of each task. Mathematically, the total loss for each task is

$$\text{Loss} = L(\theta) + \lambda_s R_s(\theta)$$

where $L(\theta)$ is the original target task loss, $\lambda_s > 0$ is a tunning parameters and $R_s(\theta)$ is the smoothness-inducing regularizer defined as

$$R_s(\theta) = \frac{1}{n}\sum_{i}^{n} \max_{||\overline{x_i} - x_i||_{p \leq \epsilon}} l(f(\overline{x_i};\theta), f(x_i;\theta))$$

where $\epsilon > 0$ is a tinning parameter and l_s is chosen as symmetrized KL-divergence. Intuitively, the method stables the output by giving a small perturbation and ensures no big change in output.

We adapt the smart-pytorch algorithm code from archinetai (2022) and apply this method for all the three downstream tasks.

Other training techiques To prevent model from overfitting on training data and achieve more generalize capabilities, we also used other training techiques such as dropout and weight decay. We also used learning rate decay and warm up during training to make global optimization more efficient especially when gradients become small.

5 Experiments

5.1 Data

For semantic analysis task, we use the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), consisting of 111855 single sentences from movie reviews and labeled with 5 classifications from negative to positive. In part 1, we also use CFIMDB dataset for semantic analysis. It consists 2434 highly polar movie reviews. For paraphrase detection task, we use Quora Dataset (DataCanary, 2017), consisting of 400000 questions pairs and labeled with whether different questions are paraphrases of each other. For semantic textual similarity task, we use SemEval STS Benchmark Dataset, consisting of 8628 sentences pairs and labeled with a similarity scale from 0 to 5.

5.2 Evaluation method

For sentiment classification and paraphrase detection tasks, multi-class and binary classification accuracy is used to evaluate the model. For semantic textual similarity task, we use the Pearson correlation of the true similarity values against the predicted similarity values as Agirre et al. (2013) to evaluate the performance. The score of the model will be compared with the baseline model score.

We use the model described in the previous section to train on three tasks' database sequentially for each epoch and sum the loss for all tasks for optimization. We use this method as our baseline model.

5.3 Experimental details

The experiment starts with loading weights from pretrained BERT model downloaded from HuggingFace with model 'bert-base-uncased' (Hug). There are two options to train the model: pretrain or finetune. Pretrain freezes the weights of BERT model and only updates predication heads. The learning rate for pretrain is 0.001. For finetune, both BERT model weights and prediction heads parameters get update with a learning rate of 0.00001. For Part 1, we use both pretrain and finetune to train on SST and CFIMDB dataset. For Part 2, we use finetune method on three dataset. For each dataset, 70% data is used for traing and 10% for dev. The rest 20% is used for test. We trained with batch size 16 for 10 epochs, initial learning rate 1e-5 and gradually decays to 1e-7 with 1 epoch of warm up. We tested different dropout rates of 0, 0.1, and 0.3. We also tested different weight decay value of 1e-3 and 1e-5.

5.4 Results

We selected some evaluation results on dev set to compare between different models and training settings, as shown in Tabel 1. From the results of baseline and baseline with GS (Gradient Surgery), gradient surgery method indeed has improvements on multi-task training. The initial design of optimizing semantic textual similarity as a classification task shows poor results on STS datasets. However changing to optimizing the cosine similarity between embeddings has large improvement on this task. Based on the cosine similarity optimization, adding SMART regularization is also improving the overall performance especially on sentiment classification of SST dataset. The training curves are shown in Figure 7

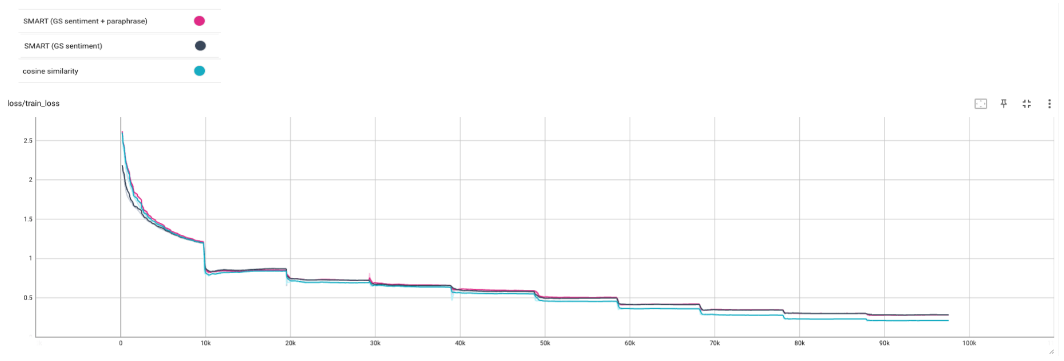


Figure 7: Training loss

	overall	SST	Quora	STS
Baseline	0.635	0.453	0.789	0.324
Baseline (GS)	0.654	0.492	0.777	0.383
Cosine similarity (GS)	0.741	0.500	**0.786**	**0.874**
Cosine similarity (GS + dropout)	0.744	0.516	0.781	0.869
Cosine similarity (SMART + GS + dropout)	**0.746**	**0.520**	0.782	0.873

Table 1: Comparison of performance on three datasets

Effect of SMART regularization We also compared dev accuracy curves during training under different model settings, as shown in Figure 8. Here we listed the comparison of three different setting, cosine similarity, adding SMART on sentiment classification task, and adding SMART on both sentiment classification and paraphrase detection tasks. We can observe that adding SMART regularization can achieve higher accuracy scores and adding on both tasks has even better performance.

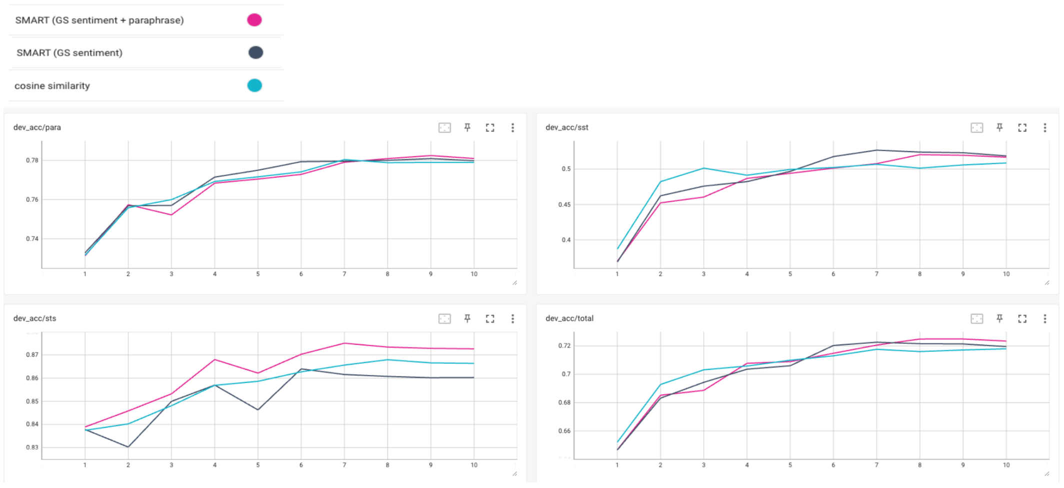


Figure 8: Dev accuracies

Effect of dropout We tested training with different dropout rate settings. As shown in 9, having dropout rate 0.1 is showing advantage over no dropout training. However larger dropout rate at 0.3 is getting worse results, one possible explanation is that it loses too much information.

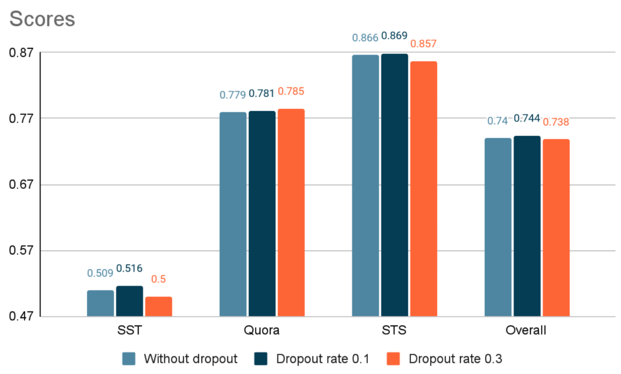


Figure 9: Scores on dev set under different dropout setting

6 Analysis

There are lots of challenges in multi-tasks learning such as model structure and objective design, balancing among different objectives, pretrained model selection, preventing from overfitting and ensuring generalization abilities.

In this project, we at first tested training three tasks sequentially but the result isn't ideal. There is catastrophic forgetting issue on previously trained tasks after switching to a new task. Therefore, training with multiple tasks on several different dataset is a more reasonable option. However, it's critical to balance the optimization among different objective functions and prevent from overfitting to only some of them. Our initial design soon reached to bottleneck of improving performance, our understanding is that the difficulties in optimizing the text similarity task is having the major impact on the regression, since large gradients has larger effects on the parameter updates. Therefore we started to use cosine similarity for text similarity objective, it shows significant boost on both similarity accuracy and overall performance. Also we explored the Gradient Surgery approach in order to balance the training of different tasks and it also shows great improvements.

During training we also notice that training accuracy can get very high over 0.9 on all three tasks while evaluation on dev set can be much lower. This is a overfitting problem where model is weak in generalizing to data it hasn't been familiar with. This will cause the model perform much worse on unseen data comparing to the performance during training. Our efforts in improving the model's generalization ability mainly include experimenting with dropout, weight decay, and SMART regularization approach. Including the SMART regularization indeed help improve the performance on dev dataset. We also did some analysis on different training setting and suitable parameters like dropout rate 0.1 is showing better results.

7 Conclusion

The project enhances our understanding on BERT model architecture and comprehend methods to improve multi-task learning performance, specifically on three downstream tasks, sentiment analysis, paraphrase detection and semantic textual similarity. We significantly improves our model performance from baseline by using learned techniques such as gradient surgery, SMART regulation, AdamW optimizer, different pooling strategy and loss calculation method. This project has the limitation of exploring only several methods mentioned in this report. We lack of exploring architecture modification and further parameter tuning, which will be the focus of future work.

References

Bert base model (uncased). HuggingFace.

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, Atlanta, Georgia, USA. Association for Computational Linguistics.

archinetai. 2022. archinetai/smart-pytorch. GitHub.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, N. Gomez Aidan, Kaiser Lukasz, and Polosukhin Illia. 2017. Attention is all you need. volume abs/1706.03762.

Timothy Dai. 2024. minbert-default-final-project. GitHub.

Lili Jiang Meg Risdal Nikhil Dandekar tomtung DataCanary, hilfialkaff. 2017. Quora question pairs. Kaggle.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Loshchilov Ilya and Hutter Frank. 2019. Decoupled weight decay regularization.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Bi Qiwei, Li Jian, Shang Lifeng, Jiang Xin, Liu Qun, and Yang Hanfang. 2022. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics*, pages 2663–2669, Online. Association for Computational Linguistics.

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, Hong Kong, China. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Yu Tianhe, Kumar Saurabh, Gupta Abhishek, Leveine Sergey, Hausman Karol, and Finn Chelsea. 2020. Gradient surgery for multi-task learning. In *34th Conference on Neural Information Processing Systems*, Online. NeurIPS 2020.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad. GitHub.

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