minBERT and Extensions over Downstream Tasks

Stanford CS224N Default Project
Mentor: Cathy Yang

Weimin Wan
Department of Civil and Environmental Engeineering
Stanford University
weiminw@stanford.edu

Taiqi Zhao
Department of Civil and Environmental Engeineering
Stanford University
tqzhao@stanford.edu

Zherui Lin
Department of Civil and Environmental Engeineering
Stanford University
zherui@stanford.edu

Abstract

Pretrained models, such as Bidirectional Encoder Representations from Transformers (BERT), have demonstrated outstanding performance in various natural language understanding tasks. This project aims to implement the minBERT model and fine-tune it for multiple downstream natural language processing (NLP) tasks. In addition, advanced processing techniques are explored as extensions to improve the results across all downstream tasks simultaneously. Specifically, the extensions include sentence concatenation, gradient surgery, and SMART regularization. The best model achieved remarkable performance on the paraphrase detection task, with an accuracy of 87.8% on the test set, and on the Semantic Textual Similarity (STS) task, with a Pearson correlation coefficient of 0.886 on the test set.

1 Introduction

Text classification is a widely researched area in Natural Language Processing (NLP) and involves assigning predefined categories to a given text sequence. Previous work uses various neural models, such as convolution models, recurrent models, and attention mechanisms, to learn text representations.

Pretrained models on large corpora have been shown to be beneficial for text classification and other downstream NLP tasks. Word embeddings and contextualized word embeddings are often used as additional features. Sentence-level pretrained models have also been proposed and achieved state-of-the-art results on several text classification datasets.

These pretrained models have recently been useful in learning common language representations by utilizing a large amount of unlabeled data. In particular, Bidirectional Encoder Representations from Transformers (BERT) has achieved amazing results in many natural language understanding tasks, but its potential has yet to be fully explored. One benefit of applying BERT as a pretrained model is that sentence embeddings it provides can be parallelly and simultaneously manipulated for multiple downstream NLP tasks, such as sentiment analysis, paraphrase detection, and semantic textual similarity (STS) determination.

This project aims to improve the performance of multiple downstream NLP tasks by fine-tuning BERT embeddings. We also explore sophisticated processing in order to balance the improvement of the performance of these multiple downstream NLP tasks.

The preliminary work is to evaluate the effectiveness of the implemented minBERT model by conducting the downstream sentiment analysis task using the SST dataset and the CFIMDB dataset, employing both non-fine-tuning and fine-tuning processes on top of the pretrained minBERT model. Subsequently, our main work is to utilize the minBERT model across various downstream tasks in an optimal manner. To achieve this goal, several advanced strategies and state-of-the-art techniques are investigated and applied as extensions to our baseline model. As a result, our final best model demonstrates a significant improvement compared to the baseline.

2 Related Work

The downstream tasks are independent, thus optimizing the layers and processing procedures for each individual task can finally result in an enhanced overall performance across the multiple downstream tasks. Remers and Gurevych [1] propose that a siamese network architecture is well-suited for processing sentence pairs, where two sentences are input into the minBert model, generating two sentence embeddings that are subsequently engineered to produce logits for further analysis. Additionally, the original BERT paper by Devlin et al. [2] also presents an approach for handling multiple input texts (sentences), where the sentences are concatenated together, separated by a [SEP] token, and provided as a single input to the BERT model.

In this project, the multiple downstream tasks have different independent datasets, but they share the same upstream minBERT model. During the fine-tuning process of the shared minBERT model, its learned parameters may tend to optimize the performance of specific downstream tasks rather than the overall performance across all downstream tasks. Therefore, it is of paramount importance to design a mechanism that balances the training steps across multiple downstream tasks. For example, Bi et al. [3] use multi-task learning, whereby the loss of each downstream task is added to form the overall loss of the entire model. To mitigate the potential conflicts between gradients, Yu et al. [4] propose a novel approach referred to as gradient surgery. This method projects the learning gradient of a task onto the normal plane of the gradient of another task, resulting in two new gradients that no longer conflict with each other. Also known as projecting conflicting gradients (PCGrad), this algorithm modifies the gradient for each task, thus eliminating conflicts with the gradients of other tasks.

Over-fitting is one common problem during model training, and our baseline model shows over-fitting in all three tasks. One potential cause of over-fitting is aggressive fine-tuning. To address this issue, Jiang et al. [5] propose a smoothness-inducing adversarial regularization method. They also propose a Bregman Proximal Point Optimization method, that can prevent over-fitting by incorporating a penalty at each iteration. The SMART framework combines these two techniques, thus providing a solution to stabilizing the model and potentially mitigating over-fitting.

3 Approach

The baseline model utilized in this project is the minBERT model, which is a minimalist implementation of BERT that can be constructed using a portion of provided code. The minBERT serves as the upstream model for a single or multiple downstream tasks, where multiple tasks can share the same BERT architecture. To extend the functionality of the model for multiple downstream tasks, we employed several techniques, including (1) concatenating sentence pairs as a single input to perform end-to-end processing on the paraphrase detection and semantic textual similarity tasks, (2) implementing the gradient surgery strategy (PCGrad algorithm) to efficiently fine-tune the upstream minBERT model across multiple tasks, and (3) using the SMART framework to stabilize the training process and prevent over-fitting.

3.1 Baseline

Tokenization: The first building block of the BERT model is tokenization, which utilizes a WordPiece tokenizer. This tokenizer assigns an [UNK] token to unseen word pieces and pads the input sentence with a [PAD] token to achieve uniformity in sentence length.

Embedding Layer: The second building block of the BERT model is an embedding layer. In this project, the input embeddings are generated by summing the token embeddings and the position embeddings, both of which possess identical dimensions.

Transformer Layer: The main building block of the base BERT model is the transformer layer. This layer is composed of a multi-head attention unit, followed by a an additive and normalization layer with a residual connection, a feed-forward layer, and another additive and normalization layer with a residual connection. Specifically, the multi-head self-attention unit employs a scaled-dot product across multiple heads. When dealing with a single head, the BERT model calculates the dot product of

the query Q with all keys K, normalizes it by the dimension, and then applies a softmax function to obtain the weighted sum of each value V. The multi-head attention mechanism jointly concatenates the information from different representation subspaces at different positions. In addition to the attention sublayer, each transformer layer includes two linear transformations with a ReLU activation function. This feed-forward layer is also followed by a normalization layer. Finally, the BERT model applies a dropout layer to the output of each sub-layer before it is added to the sub-layer input and normalized.

Optimizer: Regarding the training of the BERT model, we employ the Adam Optimizer with decoupled weight decay regularization, commonly referred to as AdamW, which utilizes decreasing learning rates. This stochastic optimization technique estimates the first and second moments of the gradients to compute adaptive learning rates for individual parameters.

Loss Functions: Given the nature of the different tasks, the most general loss function for the classification tasks is the cross-entropy function, while the typical one for the regression tasks is the mean squared error (MSE) function. Accordingly, we employ the cross-entropy loss function for the sentiment analysis task, the binary cross-entropy loss function for the paraphrase detection task, and the MSE loss function for the STS task in this project.

Single Downstream Task: In this project, the preliminary work is to employ the embeddings generated by the upstream minBERT model to undertake the single downstream task of the sentiment analysis. The model for this downstream task is composed of a dropout layer and a linear projection layer that projects the output embeddings from the minBERT model to a set of logit values for classification. These logits are subjected to a softmax function to obtain the probability distribution for various labels. To train this model, we use the mini-batch training strategy.

Multiple Downstream Tasks: The main work of this project is to perform multiple downstream tasks utilizing the upstream minBERT model. These tasks consist of sentiment analysis, paraphrase detection, and semantic textual similarity (STS) determination. Each task has a distinct downstream structure while sharing the minBERT model. For the sentiment analysis task, the architecture is identical to that used in the single downstream task model discussed above. For the paraphrase detection and the STS tasks, each piece of input data contains a sentence pair instead of a single sentence. In our baseline model, we concatenate the two embeddings of the two input sentences generated by the upstream minBERT model to generate a single extended embedding to perform the two tasks. The embedding is then passed through a dropout layer and a linear projection layer to generate a logit. This logit is then passed through a sigmoid function to produce the probability of the binary label in the paraphrase detection task or to be rescaled to a similarity score in the STS task. To fine-tune the upstream minBERT model and balance the performance of the different downstream tasks, the whole model is trained using a round-robin strategy where each mini-batch learning step is executed on different downstream tasks in a cycled sequence. As a result, each task has an equal number of data batches during the training process.

3.2 Extensions

Sentence Pair Processing: To improve overall performance across multiple downstream tasks, it can be beneficial to optimize each task separately. Given that the paraphrase detection and the STS tasks involve sentence pairs rather than single sentences, further optimization of the interaction between the two sentences in the pair is possible. We investigated two architectures to address this: siamese networks and sentence concatenation.

The siamese network is a potential approach for processing sentence pairs. It involves feeding two sentences into the same model, whereby the minBERT model generates two embeddings that are then processed by the downstream structures for further processing. In our baseline model, we employ a simple siamese network for the paraphrase detection and the STS tasks. Reimers and Gurevych [1] propose different techniques to engineer the two embeddings for both classification and regression tasks. For the classification task, they concatenate not only the two sentence embeddings u and v but also their element-wise difference $|u - v|$ and/or their element-wise product $u * v$ before passing the new long embedding to the later softmax classifier. We incorporate this approach into our

paraphrase detection task. For the regression task, they compute the cosine similarity between the two embeddings and use it as the logit for the later regression. We adapt this approach to the STS task due to its intuitive relation between the cosine similarity and the STS task.

In this project, we also explore the sentence concatenation approach, which does not generate two embeddings but concatenates the two sentences in the pair with a special [SEP] token separating them. The concatenated sentence is then fed as input to the minBERT model, generating a single embedding, which is then processed with a dropout layer and a linear projection layer, following the same structure as the baseline model. We compare this approach with the siamese network approach for both the paraphrase detection and the STS tasks. Figure 1 illustrates the architecture of the extension model utilizing sentence concatenation.

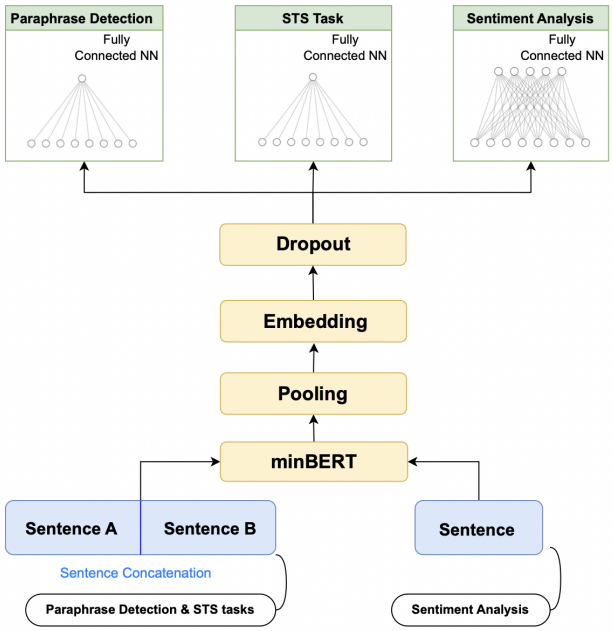

Figure 1: Extension Model Architecture

Gradient Surgery: During the training process, the upstream minBERT model is fine-tuned by the three downstream tasks. The baseline model employs a round-robin strategy to synchronously optimize the different tasks. However, optimizing each task separately can lead to conflicts between the learning gradients of the different tasks, which can impede the overall learning process. Specifically, two gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ are in conflict if their dot product $\mathbf{g}_i \cdot \mathbf{g}_j$ is negative. To mitigate the effects of conflicting gradients and ensure that the learning process progresses smoothly, Yu et al. [4] propose a gradient surgery strategy called Projecting Conflicting Gradients (PCGrad).

The fundamental idea behind the PCGrad algorithm is to project a gradient onto the normal plane of its conflicting gradient, thereby enabling the model to continue learning effectively while avoiding the adverse effects of conflicting gradients. Specifically, if two gradients, $\mathbf{g}_i$ and $\mathbf{g}_j$, are conflicting, $\mathbf{g}_i$ is replaced by

$$\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2}\mathbf{g}_j$$

This process is then repeated for all gradients from other tasks sampled in a random order from the mini-batch set. The same procedure is carried out for all tasks in the mini-batch set to calculate the corresponding projected gradients, and the final overall gradient for the entire model at the current step is the sum of all projected gradients. Algorithm 1 illustrates the PCGrad update rule.

We implement the PCGrad algorithm by adapting it from an existing codebase [1] and conduct experiments to evaluate its effectiveness in improving the fine-tuning process of the whole model.

[1] https://github.com/WeiChengTseng/Pytorch-PCGrad

Algorithm 1 PCGrad Update Rule

Require: Model parameters θ, task minibatch $B = \{T_k\}$
$\mathbf{g}_k \leftarrow \nabla_\theta L_k(\theta) \forall k$
$\mathbf{g}_k^{PC} \leftarrow \mathbf{g}_k \forall k$
for $T_i \in B$ **do**
  for $T_j \sim B \backslash T_i$ in random order **do**
    if $\mathbf{g}_i^{PC} \cdot \mathbf{g}_j < 0$ **then** $\mathbf{g}_i^{PC} = \mathbf{g}_i^{PC} - \frac{\mathbf{g}_i^{PC} \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2} \mathbf{g}_j$
    end if
  end for
end for
$\Delta\theta = \mathbf{g}^{PC} = \Sigma_i \mathbf{g}_i^{PC}$

SMART Regularization Framework: We investigate a regularization framework proposed by Jiang et al. [5] to address the issue of over-fitting that is observed in all three downstream tasks of our baseline minBERT model. To this end, we explore the feasibility of applying this regularization framework to our fine-tuning process.

The framework solves the following optimization for fine-tuning:

$$\min_\theta F(\theta) = L(\theta) + \lambda_s R_s(\theta)$$

where the $L(\theta)$ is the loss function defined as

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i; \theta), y_i)$$

and $R_s(\theta)$ is the smoothness-inducing adversarial regularizer defined as

$$R_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{||\tilde{x}_i - x_i||_p \leq \epsilon} l_s(f(\tilde{x}_i; \theta), f(x_i; \theta))$$

To solve the optimization equation, we use the Bergman proximal point method which takes

$$\theta_{t+1} = argmin_\theta F(\theta) + \mu D_{Breg}(\theta, \tilde{\theta}_t)$$

where $\mu > 0$ and the Bregman divergence is defined as

$$D_{Breg}(\theta, \theta_t) = \frac{1}{n} \sum_{i=1}^{n} l_s(f(\tilde{x}_i; \theta), f(x_i; \theta))$$

And to accelerate the training, we adopt momentum where

$$\tilde{\theta}_{t+1} = (1 - \beta)\theta + \beta\theta_t$$

The problem is solved using gradient descent with a variant of the AdamW optimizer, which is also referred to the AdamaxW optimizer.

Tuning Learning Rate and Learning Rate Decay: We conduct experiments on the impact of different learning rates and learning rate decay strategies on the training process. The decay strategy is employed to gradually decrease the learning rate after each epoch, which can facilitate faster convergence, prevent the model from being stuck in spurious local minima, and reduce oscillations during the later epochs of the training process. In particular, we focus on the exponential decay strategy as the primary approach.

4 Experiments

4.1 Datasets

In this project, we use Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset to perform the preliminary single downstream task of sentiment analysis. For the main work focused on multiple

downstream tasks, we utilize the SST dataset for the sentiment analysis task, the Quora dataset for the paraphrase detection task, and the SemEval dataset for determining the semantic textual similarity.

SST dataset: The Stanford Sentiment Treebank (SST) dataset consists of 11,855 single sentences extracted from movie reviews, and is used to perform the sentiment analysis task. The dataset is parsed by the Stanford parser and contains 215,154 unique phrases annotated by three human judges. Each sentence is assigned a label from a set of five categories: negative, somewhat negative, neutral, somewhat positive, and positive. The whole dataset is divided into train, dev, and test sets with 8,544/ 1,101/ 2,210 examples, respectively. Notably, this dataset is employed in both the single downstream task and multiple downstream tasks models.

CFIMDB dataset: The CFIMDB dataset consists of 2,434 movie reviews of high polarity, and is utilized for the the sentiment analysis task. Each movie review is assigned a binary label of either negative or positive. The whole dataset is divided into train, dev, and test sets with 1,701/ 245/ 488 examples, respectively.

Quora dataset: The Quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are a pair of paraphrase, and is used to perform the paraphrase detection task. Paraphrase detection means to determine whether particular words or phrases convey the same semantic meaning. The model processes sentence pairs as inputs and generates binary labels as outputs. The whole dataset is divided into train, dev, and test sets with 141,506/ 20,215/ 40,431 examples, respectively.

SemEval dataset: The SemEval dataset consists of 8,628 different sentence pairs, each of which is assigned a similarity score on a scale from 0 (indicating no relationship between the sentences) to 5 (suggesting identical meaning). The whole dataset is divided into train, dev, and test sets with 6,041/ 864/ 1,726 examples, respectively.

4.2 Evaluation method

Since the sentiment analysis and the paraphrase detection tasks are classification tasks, accuracy is employed as the evaluation metric. Accuracy denotes the proportion of correctly predicted labels over the total number of inputs provided.

For the STS task, researchers conventionally adopt the Pearson product-moment correlation as the metric for evaluation. This metric serves as an indicator of the magnitude of a linear correlation between two series, x and y, and is denoted as r.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}}$$

4.3 Experimental details

For the single downstream task model on the sentiment analysis task, we specify the training with 10 epochs, batch sizes of 64 for the SST dataset and 8 for the CFIMDB dataset, a dropout probability of 0.3 for the downstream task, and learning rates of 10^{-3} for non-fine-tuning and 10^{-5} for fine-tuning. The training time for non-fine-tuning is approximately 20 minutes, while the fine-tuning process needs around 25 minutes.

For the multiple downstream tasks model, we specify the fine-tuning training with 10 epochs. In order to ensure equitable distribution of batches across three tasks while accommodating a 24 GB GPU, batch sizes are adjusted to 2 for the SST dataset, 36 for the Quora dataset, and 2 for the SemEval dataset. The dropout probability is maintained at 0.3. Under the AdamW optimizer, the learning rate is set to 10^{-5}, whereas the AdamaxW optimizer features an initial learning rate of 10^{-4} and a decay factor of 0.9. Each epoch of training takes approximately 20 minutes.

4.4 Results

Single Downstream Task on Sentiment Analysis: Table 1 demonstrates the dev accuracy results for the SST and CFIMDB datasets, corresponding to both non-fine-tuning and fine-tuning training modes. The results affirm that updating and fine-tuning the upstream minBERT model can markedly enhance the efficacy of the downstream task.

Dataset	Non-fine-tuning	Fine-tuning
SST	0.380	0.505
CFIMDB	0.755	0.976

Table 1: Dev Accuracy Results for Single Downstream Task on Sentiment Analysis

Multiple Downstream Tasks: For the multiple downstream tasks model, our focus was solely on the fine-tuning training process. Our best model, the extension model, amalgamates all proposed extensions, which contribute differently to the system. Table 2 presents the accuracy and correlation metrics for the three tasks on the test set for our extension model. The average metric attains a ranking of 11 among more than 150 teams on the leader board. For comparative purposes, Table 2 includes results on the dev set for both the extension model and the baseline model. It is apparent that the metrics for all three tasks are markedly enhanced by the extensions as compared to the baseline model.

Downstream Tasks	Extension Test	Extension Dev	Baseline Dev
Sentiment Analysis	0.524	0.542	0.459
Paraphrase Detection	0.878	0.878	0.735
Semantic Textual Similarity	0.886	0.877	0.353
Average Metric	0.762	0.766	0.517

Table 2: Metric Results for Multiple Downstream Tasks over Test Set for the Extension Model and Dev Set for both the Extension and the Baseline Models

Separate Extensions: To evaluate each separate extension and its contribution to the final results, we compared the best model with the model that does not implement each specific extension.

Table 3 shows the dev set results for different strategies to process the sentence pair inputs for the paraphrase detection and the STS tasks, where siamese network generates two embeddings u and v, while sentence concatenation only produces one. Based on the results, all siamese approaches do not yield a better performance than the simple sentence concatenation process, and thus the sentence concatenation is adopted in our best model.

Downstream Tasks	Paraphrase Detection	Semantic Textual Similarity
Concat(u, v) (Baseline)	0.735	0.353
Concat(u, v, $\lvert u - v \rvert$)	0.829	-
Concat(u, v, $\lvert u - v \rvert$, $u * v$)	0.803	-
Cos-sim(u, v)	-	0.710
Sentence Concatenation	**0.878**	**0.877**

Table 3: Dev Metric Results for Paraphrase Detection and SST tasks by different strategies

Table 4 shows the dev set results for the best model (PCGrad + AdamaxW), the unextended round-robin model, and the unextended AdamW model. The results demonstrate that the AdamaxW extension contributes to the overall performance to some extent. However, the implementation of PCGrad has a minimal impact compared to the round-robin strategy.

5 Analysis

Among all implemented extensions, the sentence concatenation strategy demonstrates the most significant contribution to the overall performance. Although siamese network approaches may appear to enhance the overall model performance to some extent, the human engineering introduced

Downstream Tasks	Round-robin + AdamaxW	PCGrad + AdamW	PCGrad + AdamaxW
Sentiment Analysis	0.516	0.489	**0.542**
Paraphrase Detection	**0.880**	0.877	0.878
Semantic Textual Similarity	**0.882**	0.874	0.877
Average Metric	0.759	0.746	**0.766**

Table 4: Dev Metric Results for Multiple Downstream Tasks by different strategies

to the interaction between the two embeddings remains inexplicable. In contrast, the sentence concatenation strategy feeds both sentences simultaneously into the minBERT model, allowing the complex deep layers to better understand the relationship between the two sentences, and thus the output embedding can capture more information than the human-engineered version.

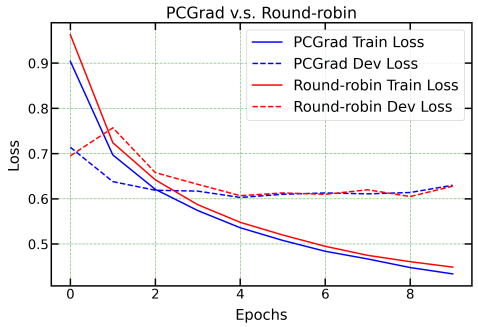

Figure 2: Loss Development for PCGrad and Round-robin algorithms

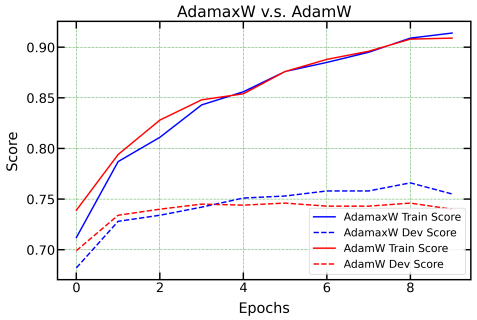

Figure 3: Average Metric Score Development for AdamW and AdamaxW optimizers

The PCGrad extension exhibits the least significant effect in the best model. Figure 2 shows the development of the training and dev losses over epochs under the round-robin and the PCGrad alorithms. It is observed that although the dev loss of the round-robin is unstable at the initial epochs, there is little difference between the two strategies during the last epochs. This phenomenon might arise from the independence among the multiple downstream tasks. As per prior research [4], gradient surgery and PCGrad are usually applied to multitask models where all downstream tasks are performed on the same input sentences. In such instances, the gradients are likely to conflict with each other since they should be jointly optimized by the same input. However, in this project, the datasets for the three downstream tasks are independent, and one sentence or one sentence pair must not be input for multiple tasks. Thus, gradients in each mini-batch may not significantly conflict, leading to little enhancement by the PCGrad algorithm.

The SMART regularization framework, utilizing the AdamaxW optimizer with a decreasing learning rate, contributes to the overall results. Figure 3 displays the development of the overall metric score over epochs on the train set and the dev set under the AdamW and AdamaxW optimizers. Although the metric score of the train set does not exhibit a significant difference, during the last epochs, the AdamaxW optimizer steadily outperforms the AdamW optimizer with a higher metric score. This trend suggests that the AdamaxW optimizer can mitigate over-fitting.

6 Conclusion

In this project, we fine-tune the minBERT model for multiple downstream tasks and incorporate three extensions, including sentence concatenation, gradient surgery, and SMART regularization, to improve its performance. Our results demonstrate that the sentence concatenation technique significantly enhances the performance of the model on the paraphrase detection and STS tasks, outperforming siamese network approaches. This implies that sentence concatenation can enable the minBERT model to better capture the relationship between sentence pairs. Additionally, we observe that the use of gradient surgery does not have a substantial impact on the training process or prediction results. Finally, we conclude that the SMART regularization method can help mitigate the over-fitting issue to some extent.

References

[1] Nils Reimers and Iryna Gurevych. 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, 2019.

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

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

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

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