SMART Fine-tuning

Stanford CS224N Default Project

Zikui Wang
Department of Electrical Engineering
Stanford University
zikuiw@stanford.edu

Abstract

Natural Language Processing (NLP) encompasses various downstream tasks. Traditional methods often rely on fine-tuning pre-trained language models, which can lead to overfitting and poor generalization. This study aims to enhance multi-task learning by integrating advanced techniques. I employed the Bidirectional Encoder Representations from Transformers (BERT) model, incorporating semantic similarity analysis, multitask training scheduling, and SMART regularization across three tasks: sentiment classification (SST dataset), paraphrase detection (QQP dataset), and semantic textual similarity (STS dataset). My optimized model, **Concat+SMART+Pretrain**, achieved significant performance improvements, with an overall accuracy of **0.800**, ranking second on the Dev leaderboard, and **0.798**, ranking third on the Test leaderboard.

1 Key Information to include

- Mentor: Chaofei Fan
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Natural Language Processing (NLP) encompasses a wide array of tasks, including sentiment classification, question answering, language translation, paraphrase detection, semantic similarity analysis, and more. Inspired by works such as Radford et al. (2018) and Devlin et al. (2018), contemporary approaches often leverage pre-trained language models, which are trained on large text corpora and then fine-tuned for specific tasks to achieve impressive results. However, aggressive fine-tuning can lead to overfitting, where the model performs well on the training data but fails to generalize to validation and test datasets.

Moreover, while this fine-tuning process is effective for single tasks, challenges arise when aiming to develop a single model that performs well across multiple downstream tasks. Different tasks can update gradients in conflicting directions during training. If these gradients diverge significantly, they can undermine the overall gradient update, hindering effective learning.

To address the issue of overfitting from aggressive fine-tuning, Jiang et al. (2020) introduced two frameworks: Smoothness-inducing regularization and Bregman proximal point optimization, collectively known as SMART. Additionally, Yu et al. (2020) proposed gradient surgery to alleviate the impact of conflicting gradients.

In this paper, I utilize the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model and apply both the SMART method and gradient surgery to three downstream tasks:

Sentiment Classification, Paraphrase Detection, and Semantic Textual Similarity. My aim is to achieve high performance across these tasks simultaneously. I also explore strategies to handle datasets of varying lengths and evaluate the performance of my proposed model.

3 Related Work

Before the advent of the Transformer architecture proposed by Vaswani et al. (2017), most NLP tasks were accomplished using Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM) networks, as introduced by Rumelhart et al. (1986) and Hochreiter and Schmidhuber (1997), respectively. However, the sequential computation requirements and difficulties in capturing long-term dependencies posed significant challenges for these methods. The introduction of the Transformer architecture addressed these issues and has since become the dominant framework for NLP tasks. Additionally, transfer learning has proven effective in NLP, as highlighted by Conneau et al. (2017).

Inspired by the success of the Transformer model and transfer learning techniques, the Generative Pre-trained Transformer (GPT) was proposed by Radford et al. (2018). GPT suggests pre-training a model on large text corpora and then fine-tuning it on task-specific datasets, leveraging the abundance of unlabeled data and the scarcity of labeled data for specific tasks. Following this, Devlin et al. (2018) introduced the Bidirectional Encoder Representations from Transformers (BERT), which uses masked language modeling to train a bidirectional language model, enhancing context awareness compared to unidirectional models.

Several techniques have been proposed to enhance model performance during both the pre-training and fine-tuning phases. For instance, Gao et al. (2021) proposed contrastive learning of sentence embeddings for both labeled and unlabeled datasets. This technique uses pairs of sentences with similar meanings, applying different masking probabilities for unlabeled data. The objective is to maximize the similarity between similar sentences and the dissimilarity between different sentences, thereby creating a more uniform embedding space. A similar approach was proposed by Reimers and Gurevych (2019), which measures cosine similarity between sentence embeddings during training and evaluation.

Despite these advancements, challenges remain, particularly during the fine-tuning stage. Aggressive fine-tuning can lead to overfitting, a problem addressed by Jiang et al. (2020) through the use of SMART regularization. Additionally, in multi-task training, different tasks may produce gradients in conflicting directions. To mitigate this, Yu et al. (2020) designed gradient surgery, a technique that projects each gradient onto the normal plane of the other. This method prevents gradient interference, ensuring that gradients are not inadvertently applied across tasks.

4 Approach

To address three downstream tasks, I first implemented a baseline model using the pre-trained minBERT model proposed by Devlin et al. (2018). For each specific task, the last layer implementation is in Appendix A.1. During training, the model is sequentially trained on various tasks independently (in the sequence of paraphrase detection, semantic textual similarity, and sentiment analysis) without further training on previously completed tasks, while allowing all parameters to be updated. This methodology ensures comprehensive utilization of all datasets and maximizes the model's capacity for fine-tuning each task effectively, rather than solely focusing on the last layers.

To further enhance my model, I implemented several advanced techniques: Task-Specific Learning (Section 4.1), Multitask Training Scheduling (Section 4.2), and Regularization (Section 4.3). These improvements aim to refine my approach and ensure high performance across all tasks. Additionally, I employed **ensembling**, a technique also used by Liu et al. (2019) in their BERT implementation to achieve higher accuracy. Individual trained classifiers typically exhibit low variance but high bias. Ensembling combines several models to reduce this high bias. In my implementation, I aggregated the outputs of 5 models to reduce their bias. Specifically, I used the mode to select the predicted value

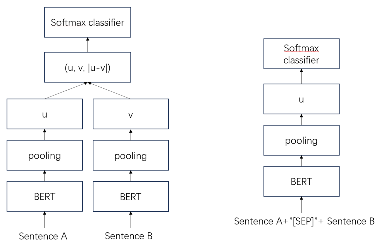

Figure 1: Architecture for Cosine Similarity (Left) and Sentence Concatenation (Right)

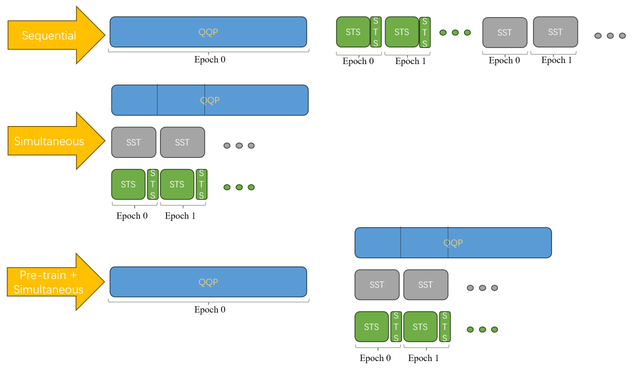

Figure 2: Three Training Schema: Sequential Training, Simultaneous Training, Pre-train+Simultaneous Training

among the classification results of the 5 models trained under different parameters and the median to select the predicted score among the regression results of these 5 models.

4.1 Semantic Similarity Analysis

Cosine Similarity Many NLP tasks, such as paraphrase detection and semantic textual similarity, aim to determine the correlation between two sentences. A traditional approach to this problem involves passing two sentences through BERT separately and concatenating their [CLS] tokens with a linear layer. However, this method may not perform well because it does not incorporate any global information about the two sentences. Moreover, comparing one sentence to all others to identify the most semantically similar sentences can be computationally intensive. Reimers and Gurevych (2019) proposed using the cosine similarity method to construct the embedding space for each sentence. Specifically, after obtaining sentence embeddings u and v from the BERT encoder, the embeddings are concatenated as u, v, $|u - v|$ and passed through a softmax classifier if it is a classification problem, as illustrated in Figure 1. For regression problems, the similarity score $\frac{u^T v}{\|u\| \cdot \|v\|} \in [-1, 1]$ is calculated and rescaled accordingly.

Sentence Concatenation Another method to capture semantic similarity information was proposed by Devlin et al. (2018). This approach treats the two sentences being compared as a single input sequence, concatenated with the [SEP] token as depicted in Figure 1. The concatenated sentence is then passed through the BERT model to obtain an embedding or the [CLS] token. The subsequent steps are similar to the traditional method, where the embedding is passed through a linear layer with a different number of neurons depending on whether the task is classification or regression.

4.2 Multitask training scheduling

Sequential Training One approach to address the challenges in multitask training is to sequentially train each task as dipicted in Figure 2. This method works well if each task only updates the weights specifically used for its own operations. For example, in my case, if I do not update the weights used by the encoder but only update the parameters used in the last layer for each task, sequential training can be effective. However, this approach poses a significant problem if tasks share common parameters, such as those in the encoder. When the model updates these shared parameters according to the most recent task, it is highly likely that such changes will adversely affect the performance of previously trained tasks. This parameter interference can lead to a deterioration in the model's overall performance across all tasks.
Simultaneous Training Simultaneous training offers a potential solution to the issues inherent in sequential training. In this approach, the model is trained by performing inference on all three tasks and calculating their respective losses separately as illustrated in Figure 2. The total loss is then computed as the sum of these individual losses ($\mathcal{L}loss = \mathcal{L}para + \mathcal{L}SST + \mathcal{L}STS$), and the model parameters are updated simultaneously. This method helps mitigate the "forgetting" problem that can occur with sequential training. It is also important to note that different tasks may have datasets of varying lengths. For instance, in my example, the SST dataset contains 8,544 samples, which is slightly more than the STS dataset with 6,040 samples, but significantly fewer than the QQP dataset with 283,010 samples. To fully utilize each dataset and balance the overfitting problem, I propose treating the complete training of the SST dataset as one epoch. Within each epoch, I cycle through the STS dataset and, across all epochs, cycle through the entire QQP dataset. This approach ensures that all datasets are effectively utilized and the model is well-tuned across different tasks.
Gradient Surgery Despite the advantages of simultaneous training, Yu et al. (2020) hypothesized that different tasks may result in gradient update directions that conflict with each other. For instance, suppose I have two tasks with gradient update directions g_i and g_j. If these gradients are updated in opposite directions, it could lead to destructive interference when updating them together. To address this issue, Yu et al. (2020) proposed projecting each gradient onto the normal plane of the other if the angle between them exceeds 90 degrees, as illustrated in Equation 1:

$$g_i = g_i - \frac{g_i^T g_j}{\|g_j\|^2} g_j \tag{1}$$

In my implementation, I utilized the GitHub repository[1].
Further Pre-training In my example, the QQP dataset is significantly larger than the other datasets. To fully leverage this larger dataset, I proposed initially training the model solely on the QQP dataset as shown in Figure 2. After this pre-training phase, I then train the model on the multi-task downstream tasks simultaneously. In this way, the QQP dataset serves as an additional pre-training resource. Empirical studies have shown that this method can substantially improve model performance.

4.3 SMART Regularization

Jiang et al. (2020) highlighted that aggressive fine-tuning during the fine-tuning stage often leads to overfitting, where the model performs well on the training dataset but fails to generalize to unseen data. To address this issue, Jiang et al. (2020) proposed two techniques: Smoothness-inducing regularization and Bregman proximal point optimization.
Smoothness-inducing regularization This proposed method is used to control the model complexity. The motivation is that if I want to add small perturbation to the model input, I hope not to see large change in the output. Mathematically speaking, it can be computed by the following objective:

$$\min_\theta \mathcal{F}(\theta) = L(\theta) + \lambda_s R_s(\theta) \tag{2}$$

Here, $L(\theta)$ is the task specific loss function defined as:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i) \tag{3}$$

where $\ell(\cdot,\cdot)$ depends on the task and n is the total number of data points.
The $\lambda_s R_s(\theta)$ is the newly added regularization. Specifically, λ_s is the control parameter used to

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

balance task objective and regularization objective. $R_s(\theta)$ is the regularization objective defined as below:

$$\mathbf{R}_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_s(f(\tilde{x}_i;\theta), f(x_i;\theta)) \tag{4}$$

In this equation, we perturbed x_i by the range defined by the tuning parameter ϵ and would want to make the hugely perturbed output $f(\tilde{x}_i;\theta)$ to be as similar as original output $f(x_i;\theta)$. The way to achieve such a goal is through loss function $\ell_s(;\theta)$, which is defined as KL divergence $\ell_s(P,Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P)$ if the underlying task is a classification problem where the output is a probability simplex and is defined as Mean Squared Error $\ell_s(p,q) = (p-q)^2$ if the underlying task is a regression problem.
Bregman proximal point optimization This is the second technique introduced by Jiang et al. (2020). Different from the smoothness-inducing regularization which helps to control the surrounding smoothness of the optimal point. Bregman proximal point helps to control the the trained parameter not too far away from the parameter obtained from the pre-training stage. In each iteration, the parameter is updated in the following way:

$$\theta_{t+1} = \arg\min_{\theta} \mathcal{F}(\theta) + \mu \mathcal{D}_{Breg}(\theta, \theta_t) \tag{5}$$

where $\mu > 0$ is a tuning parameter, and $\mathcal{F}(\theta)$ entails the original objective and smoothness-inducing regularization. $\mathcal{D}_{Breg}(\theta, \theta_t)$ is defined as follows:

$$\mathcal{D}_{Breg}(\theta, \theta_t) = \frac{1}{n}\sum_{i=1}^{n} \ell_s(f(x_i;\theta), f(x_i, \theta_t)) \tag{6}$$

with θ_0 be the parameter obtained from pre-training and ℓ is defined the same as above regularization. This ensures that the updated θ not far away from θ_0 and thus making sure it not over-fitting on the specific task.

5 Experiments

5.1 Data

I used three different dataset for finishing three different downstream task. Specifically, for sentiment analysis, I used **Stanford Sentiment Treebank** (SST) dataset Socher et al. (2013). The dataset comprises 11,855 unique sentences sourced from movie reviews. Each phrase, extracted from parse trees, has been individually annotated by three human judges. These annotations categorize each phrase into one of five sentiment labels: negative (0), somewhat negative (1), neutral (2), somewhat positive (3), or positive (4). For the paraphrase detection, **Quora Question Pairs** (QQP) dataset[2] is selected. It encompasses more than 404,298 question pairs. Each pair is annotated with a binary value indicating whether the two questions are paraphrases of each other. Finally, I wil use **Semantic Textual Similarity**(STS) Agirre et al. (2013) for semantic textual similarity task, in which it comprises 8,628 distinct sentence pairs with varying degrees of similarity, rated on a scale from 0 (unrelated) to 5 (equivalent meaning). The exact training validation and test splits can be refereed in Table 1

Table 1: Train, validation, and test splits for three different datasets.

Task	Dataset	Train	Validation	Test
Sentiment Analysis	SST	8,544	1,101	2,210
Paraphrase Detection	QQP	283,010	40,429	80,859
Semantic Textual Similarity	STS	6,040	863	1,725

5.2 Evaluation Method

For the sentiment analysis and semantic textual similarity tasks, both of which are essentially classification problems: Sentiment Analysis: I will use accuracy as the primary metric for evaluation.

[2]`https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

Accuracy measures the proportion of correctly classified instances over the total number of instances. Semantic Textual Similarity (STS): I will employ the Pearson correlation coefficient as the primary metric for evaluation. The Pearson correlation coefficient measures the linear correlation between predicted and true similarity scores, providing a measure of how well the model's predictions align with the ground truth similarity scores.

5.3 Experimental details

I maintained consistent configurations across all tasks, including a learning rate of 1e-5 , a batch size of 8, and typically running for 10 epochs. However, for the paraphrase detection task, which utilized the larger Quora dataset, convergence was achieved within just 2 epochs. I employed the AdamW optimizer with parameters set to Beta (0.9, 0.999) and Epsilon 1e-6. For the minBert model, a hidden dropout probability of 0.3 was utilized. Additionally, I set the tuning parameter for SMART regularization to 1 (λ_s in Equation 2) for convenience.

5.4 Results

Table 2: Dev Accuracy of Different Methods on 3 Tasks

Model	Training Method	SST	QQP	STS	Overall
Baseline	Sequential	0.506	0.673	0.224	0.597
Cos-Sim		0.507	0.787	0.394	0.664
Concat		0.494	0.795	0.833	0.735
Concat+SMART		0.530	0.782	0.771	0.733
Concat+SMART*	Simultaneous	0.510	0.883	0.874	0.777
Concat+SMART+Pretrain		0.530	0.906	0.873	0.791
Concat+SMART+Surgery+Pretrain		0.533	0.904	0.864	0.790

5.4.1 Model Comparison

In this section, I will elaborate on the performance of different models as illustrated in Table 2.
Cosine-Similarity(Cos-Sim) One of the initial enhancements implemented to improve accuracy is the use of cosine similarity. With the integration of the cosine similarity method, I observed a notable increase in accuracy by 15% for both the QQP and STS datasets. This improvement is expected since both tasks involve comparing pairs of sentences, which is where cosine similarity is applied. Furthermore, this method outperforms the baseline approach by providing a more nuanced description of the relationship between sentences.
Concatenation(Concat) In addition to utilizing cosine similarity for assessing sentence relationships, another approach involved concatenating two sentences using the [SEP] token. This modification yielded a significant improvement over the Cos-Sim method, with a 40% increase in accuracy observed on the STS dataset. The effectiveness of this approach can be attributed to the BERT model's pre-training on large corpora text, which has equipped it with strong capabilities in processing multiple sentences, leveraging the [SEP] token for next sentence prediction during pre-training. As a result, transfer learning has played a pivotal role in preparing the model for the new task.
SMART Regularization (SMART) In response to the observed severe overfitting issue with the SST dataset, SMART regularization was employed. Initially, I set the tuning parameter (λ_s in Equation 2) to 1, indicating equal treatment of task-specific loss and regularization loss. This adjustment resulted in a notable increase in accuracy by 3.5%, demonstrating the effectiveness of the SMART method in mitigating overfitting. The subsequent drop in performance on both the QQP and STS datasets aligns with expectations. Given the sequential training approach implemented, where SST is the final task trained on, the incorporation of regularization loss suggests a potential "forgetting" phenomenon.
Simultaneous Training Up to this point, each task has been trained sequentially, resulting in a noticeable drop in accuracy when subsequent tasks are trained. This "forgetting" phenomenon prompted me to devise a new training scheme for multi-task learning. One solution involves aggregating the losses and updating weights simultaneously. Notably, the Concat+SMART* method demonstrated improvements in both the QQP and STS tasks, which were forgotten in the sequential training.
Further Pretrainig Despite the fact that the QQP dataset is larger than the other two datasets, there is still a need for my model to fully utilize the training data to achieve high scores. To address this

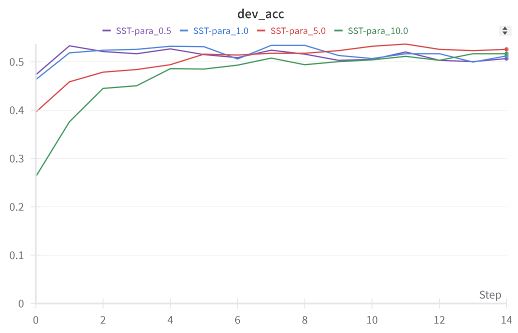

Figure 3: Performance of Different Regularization Parameter on SST Dataset. (para_0.5 represents the tuning parameter λ_s in Equation 2 is 0.5)

issue, one approach is to further pretrain the model on the QQP dataset for one epoch (since the Bert model was able to learn from it in just 1 to 2 epochs). This technique not only demonstrates improved performance in its task but also aids in enhancing the SST task, possibly due to additional benefit of transfer learning.

Gradient Surgery (Surgery) I attempted to mitigate gradient conflict issues by deploying the gradient surgery method. However, I did not observe a clear improvement from this implementation. One possible reason for this lack of improvement is that there may not be significant gradient direction conflicts in my multi-task learning setup. Given that gradient surgery did not yield substantial improvements and consumed considerable GPU memory to run, I made the decision not to adopt this method in subsequent tasks.

Table 3: Performance of Differernt Dropout Probability under λ_s in 1 and 5

Model	λ_s	Dropout Probability	SST	QQP	STS	Overall
Concat+SMART+Pretrain	1	0.1	0.536	0.905	0.884	0.794
		0.2	0.546	0.900	0.888	0.797
		0.3	0.530	0.906	0.873	0.791
		0.4	0.536	0.900	0.888	0.793
		0.5	0.537	0.897	0.884	0.792
Concat+SMART+Pretrain	5	0.1	0.544	0.904	0.893	0.798
		0.2	0.530	0.905	0.895	0.794
		0.3	0.540	0.902	0.889	0.796
		0.4	0.544	0.905	0.889	0.798
		0.5	0.530	0.903	0.884	0.792

Table 4: Dev/Test Accuracy of Ensemble Method with Parameter Tuning on 3 Tasks

Model	Dev/Test	SST	QQP	STS	Overall	Ranking
Concat+SMART+Pretrain	**Dev**	**0.545**	**0.908**	**0.894**	**0.800**	**2**
	Test	**0.540**	**0.908**	**0.893**	**0.798**	**3**

5.4.2 Parameter Tuning & Ensemble Result

To achieve optimal performance, I conducted parameter tuning for two key parameters: BERT Dropout Probability (the dropout probability for all fully connected layers in the embeddings, encoder, and pooler) and the SMART Regularization parameter (parameter λ_s in Equation 2). For the SMART Regularization parameter, I tested it directly on the SST dataset (refer to Figure 3) and determined that a tuning parameter of 5 provided the best performance. For the BERT Dropout Probability, I trained the model with various dropout probabilities and evaluated performance on the Dev dataset with regularization parameters equal to 1 and 5. According to Table 3, I found that a Dropout Probability

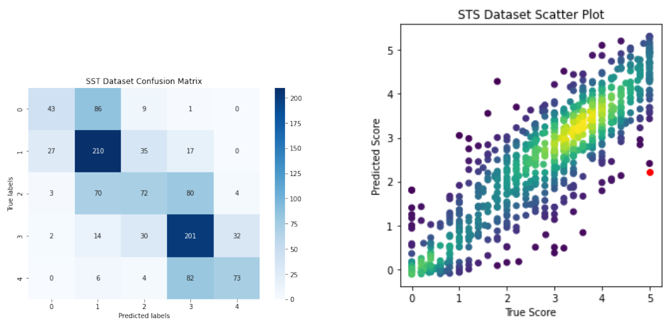

Figure 4: Visualization of Best Model Performance on SST and STS Dataset, on the Right, the Yellower the Color, the Greater the Density

of 0.1 and 0.4 with $\lambda_s = 5$ yielded the best results on the Dev dataset. To further enhance the results, I utilized the ensemble method, and the result is shown in Table 4. This approach achieved my best performance on the DEV leaderboard (ranked 2) and Test leaderboard (ranked 3).

6 Analysis

I have visualized my best model performance on the SST and STS datasets, as shown in Figure 4. On the left, there is a confusion matrix of true labels and predicted labels for the SST dataset. This confusion matrix clearly shows that the model performs well in predicting sentences with sentiments classified as somewhat negative (1) and somewhat positive (3). However, it struggles to accurately predict negative (0), neutral (2), and positive (4) sentiments. The results indicate that the model tends to predict sentences with intermediate sentiment levels: somewhat negative (1) and somewhat positive (3), but it is less effective at distinguishing between strong sentiments and neutral sentiments. Furthermore, I examined a sentence labeled as negative (0) but predicted as somewhat positive (3) by the model: "This one is definitely one to skip, even for horror movie fanatics." While humans can easily understand this sentence, the model seems to focus more on "movie fanatics" rather than "skip." This might be due to the model's inability to grasp the meaning of "even for." On the right, the visualization shows a clear relationship where the model performed well overall. However, it is still important to identify where the model failed. The red dot on the graph represents the case where the predicted score is most different from the ground truth score. In this case, the two sentences are: "it's also a matter of taste," and "it's definitely just a matter of preference," with a predicted score of 2.22 compared to the ground truth score of 5. Upon examining the training set, I found that the most related training examples are "It's also a matter of taste" and "It's mostly a matter of taste," both with a score of 5. There are no examples showing a relationship between "taste" and "preference." I believe that the model's inability to understand this relationship makes it difficult to predict correctly. By changing the word "preference" to "taste," the overall score improved to 4.56.

7 Conclusion

In this study, I tackled the complexities of multi-task learning in Natural Language Processing (NLP) through the application of the BERT model. My comprehensive approach, integrating multitask training scheduling, SMART regularization, and gradient surgery, yielded promising results across diverse downstream tasks including sentiment classification, paraphrase detection, and semantic textual similarity analysis. **Remarkably, my model secured second and third place rankings on the DEV and Test leaderboards respectively, highlighting its effectiveness in addressing complex NLP challenges.** My findings underscore the significance of pre-training language models and the incorporation of regularization techniques for enhancing performance in downstream tasks.
Furthermore, while my model demonstrated strong performance across various tasks, limitations persist, particularly in its performance on the SST dataset. Future research efforts may focus on refining model structures or fine-tuning strategies specifically tailored to address challenges in this dataset, thereby enhancing overall model robustness and effectiveness.

8 Ethics Statement

Language models are powerful tools derived from human language, encapsulating diverse perspectives. When utilized by humans, they inevitably influence language and opinions, impacting societal discourse. Consequently, careful consideration must be given to mitigate several issues that arise during their training. In my project, one significant concern is the potential **reinforcement of biases** present in the datasets. For example, comments in the SST dataset may include biases related to gender, race, and other sensitive attributes. If not addressed, this bias can persist in the trained model, amplifying societal inequalities. Strategies to counteract this phenomenon include modifying training data to interchange gender-specific terms and conducting post-training evaluations for bias detection.

Another challenge is the **dissemination of misinformation** through these models. Some datasets may contain inaccuracies or illicit content, necessitating thorough screening to ensure the model's training data integrity. Even with high-quality datasets, limitations of deep learning can still cause issues. For example, in the semantic textual similarity task, a person might compare "Paris is the capital of France" with "Shanghai is the capital of China" and find a high correlation between these sentences. This could lead to the mistaken belief that Shanghai is the capital of China. Mitigation efforts involve identifying and quantifying misinformation within datasets, followed by appropriate adjustments or replacements. Moreover, ongoing efforts are required to ensure the language generated by these models remains appropriate and aligned with ethical standards post-training.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

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*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

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*.

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

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

A Appendix

A.1 Last Layer Implementation

- Sentiment Classification: The given sentence is passed through minBERT, and the [CLS] output is fed into a single linear layer with five neurons (corresponding to the number of classes). The cross-entropy loss is calculated as the loss function.
- Paraphrase Detection: Two sentences are passed through minBERT separately, and their [CLS] outputs are concatenated. This concatenated output is then fed into a linear layer with one neuron. Binary cross-entropy loss is used as the loss function.
- Semantic Textual Similarity: The structure is the same as for Paraphrase Detection, but with a different loss function: mean squared error, since this is a regression problem.