BERT's Odyssey: Enhancing BERT for Multifaceted Downstream Tasks

Stanford CS224N **Default** Project

Haoming Zou
Department of Electrical Engineering
Stanford University
zhm2023@stanford.edu

Minghe Zhang
Department of Statistics
Stanford University
minghez@stanford.edu

Abstract

We aim to extend the capabilities of pre-trained BERT(Devlin et al., 2019) model for Sentiment Analysis, Paraphrase Detection, and Semantic Text Similarity by fine-tuning it with task-specific heads, incorporating strategies such as SMART regularization, Multi-task learning, MultipleNegativesRankingLoss, Triple-Bert single task finetuning, architecture modification, and hyperparameter tuning, to enhance task-specific performance while maintaining its broad applicability and generalization. Our Ensemble model, which attained a notable and so far the **highest overall test score of 0.797 (SST accuracy of 0.557, PARA accuracy of 0.896 and STS correlation of 0.876)**, has achieved the **top 2** performance on the test leaderboard by March 16th.

1 Key Information To Include

Team contributions: We contribute equally to the project and do all the implementation, experimentation, analysis and writing together.

Project Mentor: Olivia Lee

2 Introduction

We start our BERT's Odyssey here. Advancements in pre-training language models have significantly improved unsupervised and fine-tuning methods. While models like ELMo faced limitations with its feature extraction, it also demonstrated the promising aspects of transfer learning from large datasets. BERT, building on top of these developments, is therefore our project's foundation, employing a novel approach to pre-training and fine-tuning to enhance task performance across different domains.

Despite these advancements with pre-training on high-resource domains, given the scarcity of target task data, overfitting phenomenon emerged as a serious question to consider. Various regularization techniques, including SMART, were designed to overcome such limitation, aiming to enhance the model's robustness and generalizability. Acknowledging the novel approaches along with the limitation, our implementation efforts focused on constructing a minimal BERT encoder, experimenting with various hyperparameters, investigating the impact of different architectures including input strategies and head layers, and utilizing extensions such as SMART for regularization, MultipleNegativesRankingLoss as contrastive learning to improve embedding, and Multi-task learning for richer representation. The combination between various multi-task learning strategies and SMART regularization is our **original** contribution.

Through extensive experimentation, we aim to refine our understanding of how to best utilize BERT's capabilities, balancing between task-specific optimization and the broader goal of generalizable, robust language understanding. This paper presents our findings, detailing the challenges encountered, solutions proposed, and the implications of our work for future NLP research.

3 Related Work

The research in pre-training general language representations has made great improvements in unsupervised feature-based approaches and fine-tuning approaches. While models like ELMo generalizes word embedding, extracting context-sensitive features from two opposite directional language models, their bidirectionality were limited. Also, transfer learning from supervised data indicates that pre-trained models on large datasets can effectively aid across different tasks. Therefore, as the foundation for our project, BERT (Bidirectional Encoder Representations from Transformers), was introduced as a combination of these developments, utilizing a novel pre-training and fine-tuning approach. As it conditions on both left and right context, BERT improves its behavior for natural language processing tasks much better than unidirectional pre-training models that used to be prevalent, by using a masked language model(MLM). Alongside MLM, BERT also utilizes next sentence prediction to further improve its understanding of sentence-level context and relationships. Since BERT has already learned rich and context-aware representations of language, it can be fine-tuned with just one, or minimal additional output layers, without performing extensive task-specific adjustments, to achieve state-of-the-art results across various of NLP tasks, including question answering and language inference. Therefore, BERT simplifies the requirement for architectural complexity that is demanded to achieve high performance for downstream tasks.

Despite the development of two-stage models of pre-training high-resource domains and fine-tuning on target tasks with limited data, including BERT, fine-tuning stage can be challenging because of the risk of overfitting due to high complexity of pre-trained models and limited data from target domain. SMART (Jiang et al., 2020) solves the overfitting problem and accounts for the tuning efforts and robustness for improving model's performance for the second fine-tuning stage. It proposes Smoothness-inducing Adversarial Regularization. Inspired by local shift sensitivity on robust statistics, this regularization adds a perturbation to the input, which effectively control the change of the output, enforcing the smoothing of the model. As a result, it successfully manages the complexity of the model for avoiding overfitting. However, SMART acknowledges the limitations of computational resources and time–more training time and more memory usage, affecting the scope of experiments with BERT.

Moreover, instead of fine-tuning BERT on individual tasks, adding up each task's losses as one total loss is another approach to update BERT, as proposed in multi-task learning (Bi et al., 2022). Since the learning objectives for some tasks have synergies, by employing multi-task learning with such auxiliary tasks, multi-task learning takes advantage of the richer representation of the auxiliary tasks. Moreover, with one total loss for all tasks to update the parameters of the model simultaneously, the model would find representations that can generalize well across tasks, achieving a more robust performance on future data.

4 Approach

4.1 Implementation of Bert model and downstream tasks

For the **first part** of our project, we implemented a minimal BERT encoder by integrating a multi-headed self-attention mechanism, layer normalization, and position-wise feed-forward networks, and utilized pre-trained weights loaded from the Hugging Face's web, passing the sanity check. We also completed the implementation of AdamW optimizer and passed the test. After applying the pooled layer and then projecting it using a linear layer with CrossEntropy loss for Sentiment Analysis task, our work passed the pre-described reference accuracies, aligning with the project's requirements.

For the **second part**, we designed 3 different heads for the three tasks, Sentiment Analysis (SST), paraphrase detection (PARA), and semantic textual similarity (STS), which took the embedding as input and outputted the logits for binary classification (PARA) and regression (STS), with BinaryCrossEntropy loss and MSE loss respectively.

4.2 Multi-Bert

Our **baseline model** is extended from the baseline SST task by adding baseline PARA and STS tasks, where we update the parameters of one BERT model with multi-task learning. We call our baseline model **Multi-Bert**, which enhances the model's generalization and robustness(Bi et al., 2022).

Given the comparison nature of paraphrase detection, in the hope of taking advantage of contrastive learning between positive and negative samples and improving the robustness and generalization of the model for paraphrase detection, we considered another extension, **Multiple Negatives Ranking Loss Learning (MNRLoss)**(Henderson et al., 2017). It minimizes the distance between similar pairs while maximizing the distance for dissimilar pairs, as shown in equation (1). We implemented the loss from scratch and applied it on Multi-Bert: positive pairs in one batch (batchsize=64) are used to calculate MNRLoss, where K is the number of positive pairs. K negative samples are generated by rolling i position in a batch each time, for $i = 1, ..., K$. S was obtained by the output logits of the model, with tanh function to project the value into [-1, 1] as similarity and a scale of 20.

$$\mathcal{J}(x, y, \theta) = -\frac{1}{K}\sum_{i=1}^{K} \log P_{approx}(y_i|x_i) = -\frac{1}{K}\sum_{i=1}^{K}\left[S(x_i, y_i) - \log\sum_{j=1}^{K} e^{S(x_i, y_j)}\right] \tag{1}$$

4.3 Triple-Bert

Because of the lack of robustness of our naive baseline model, and because we were concerned about the negative interaction and influence between different tasks, we considered fine-tuning these three tasks independently, which is called **Triple-Bert**. Accordingly, Triple-Bert is designed as a composite model integrating three separate BERT models, each individually fine-tuned for optimal performance on distinct NLP tasks, and we aim to achieve specialized optimization for SST, PARA, and STS. Compared to baseline model, Triple-Bert mitigates negative influences between three tasks during fine-tuning, but may decrease its generalization ability.

To improve the Triple-Bert's performance on Sentence Pair comparison tasks, we then tried **different input strategies as our extension** for PARA and STS (Sentence Pair data): combining two sentences with Bert's [SEP] token as inputs for one Bert, or inputting two sentences into two shared-weights Berts(Reimers and Gurevych, 2019). We found the former strategy yielding much better performance, possibly because of the cross-attention of pair sentences that extracts the relationship information between sentences.

4.4 SmartTri-Bert

Due to the limited number of training data we can access, we would like to improve the generalization ability of our model by using regularization method, SMART, to avoid overfitting. We then implemented **SmartTri-Bert**, where we incorporated SMART into the Triple-Bert model for downstream tasks with different smart weights. To be specific, for our Triple-Bert model $f(\cdot; \theta)$ with n data points $\{(x_i, y_i)\}_{i=1}^{n}$, where x_i's represents the input sentence embeddings from the Embedding layer of the Bert model and y_i's are the corresponding labels, we focused on optimizing the fine-tuning process:

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

where λ_s is the smart weight, and $L(\theta)$ is the loss function:

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

and R_s is the smoothness-inducing adversarial regularizer:

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

For classification task SST and PARA, $f(\cdot; \theta)$ outputs a probability distribution, and l_s is chosen as the the symmetrized KL-divergence:

$$l_s(P, Q) = D_{KL}(P||Q) + D_{KL}(Q||P)$$

For regression task STS, $f(\cdot; \theta)$ outputs a scalar, so l_s is the squared loss:

$$l_s(p, q) = (p - q)^2$$

The adversarial regularizer promoting smoothness essentially evaluates the local Lipschitz continuity of the function f with respect to the given metric l_s. This feature of inducing smoothness is crucial for mitigating overfitting and bolstering the model's ability to generalize, especially in scenarios with limited resources for these three tasks. Besides implementing SMART regularizationfrom scratch, we tried different smart weights for each task in SmartTri-Bert model, imposing different levels of regularization to these three tasks.

4.5 SmartMul-Bert

After building a more robust and generalizable model with improved input strategy and SMART regularization, we would like to examine again the transfer learning ability between different tasks, exploring whether training model on one task can be helpful for other tasks, and we call the corresponding model **SmartMul-Bert**.

Firstly, given the synergies of sentence pair comparison objectives of PARA and STS, and the huge dataset of PARA compared to that of STS, we hope that what the model has learned from PARA data would help improve STS's prediction performance by training one Bert for these two tasks in each epoch sequentially. As a result, we saw an upward trend of the model's performance on STS task.

Another way for transfer learning we considered is Multi-task learning for PARA and STS, which also utilizes synergies between the learning objectives for the two tasks. In each batch, we added together the losses on the tasks of PARA and STS, $loss = loss_{PARA} + loss_{STS}$, built a single added optimization objective, and then updated the parameters of the single Bert for the two tasks. Within each batch, the way we combined STS data and PARA data is at a ratio of 1:25, ensuring that both datasets can be entirely iterated and run through under the same number of batches.

4.6 Architecture and Hyperparameter Tuning

We also performed **architecture and hyperparameter tuning** to further analyze and improve the performance of models and to do ablation studies on them. We did ablation studies on **Bert hidden layer dropout probabilities**, **head layer dropout probabilities**, **number of head layers**, and **weight decay of AdamW optimizer**.

5 Experiments

5.1 Data

For Sentiment Analysis, we used Sentiment Treebank (SST) dataset, annotated by three human judges. We used 8,544 examples for training, 1,101 examples for dev, and 2,210 examples for testing. We also used CFIMDB dataset consisting of 2,434 highly polar movie reviews. For Paraphrase Detection, we used Quora dataset consisting of 400,000 question pairs with labels. We used 141,506 examples for training, 20215 examples for dev, and 40431 examples for testing. For STS, we used SemEval STS Benchmark dataset with 8,628 different sentence pairs of varying similarity. We used 6041 examples for training, 864 examples for dev, and 1726 examples for testing.

5.2 Evaluation method

We used accuracy (or best accuracy) as the metric for SST and PARA tasks, and we utilized Pearson correlation of the true similarity values against the predicted similarity values for the STS task.

5.3 Experimental details

For the **Sentiment Analysis classifier in Part 1**, we used le-3 learning rate, batch size = 8, Bert hidden layer dropout probability = 0.1 for pretraining and 1e-5 for finetuning on Stanford Sentiment Treeband and CFIMDB datasets. Notice that in "pretrain" mode, the pretrained parameters of the BERT model are frozen, and the only changes/updates of parameters occur in the model's heads for each specific task, which is called "linear probing", examining the generalization ability of pretrained Bert model.

For our baseline **Multi-Bert** model, we used same learning rates=1e-5, hidden dropout probability = 0.1, and batch size = 64 for finetuning, each with a single linear head layer. We also tried more head layers for PARA (two linear layers and a ReLU activation, hidden size=768), and the motivation is the extreme large number of examples in Quora dataset, compared to the others.

For Multi-Bert model with **MNRLoss** for PARA task, the scale is 20, and K depending on the number of positive samples in a batch, and output logits as the S score. All other hyperparameters remain the same as in Multi-Bert model.

Triple-Bert model is an ensemble of three independent Bert models, each with task-specific head. All other hyperparameters remain the same as in baseline model. For Triple-Bert model with **improved 2-input-structure**, we used two combined sentences pair with Bert's [SEP] token as Bert input for PARA and STS. All other hyperparameters remain the same as in baseline model.

For our **SmartTri-Bert** model, an ensemble of three independent Bert models with SMART regularization, we used smart weight $\lambda_s = 0.5, 1, 2, 4, 5, 6, 7, 8$ for an ablation study and fixed the weight decay equalling to 1e-4 for SST, 0 for PARA, and 1e-3 for STS. We utilized best hidden dropout probability equaling to 0.2 for this model. Because of the better performance of improved 2-input-structure, we applied it into our SmartTri-Bert model and all the models below. All other hyperparameters remain the same as Triple-Bert model.

For our **SmartMul-Bert** model, for all attempts, we fixed the smart weight to 2 for SST, 6 for PARA and 8 for STS, which are the best parameters after the ablation study. All other hyperparameters remain the same as in SmartTri-Bert.

For **Architecture and Hyperparameter Tuning**, we conducted the **hidden layer dropout probability** ablation study on STS task with Triple-Bert, ranging from 0 to 0.5; **head layer dropout probability** on SST and STS with SmartTri-Bert from 0 to 0.9; **head layers** on all three tasks with SmartTri-Bert; and **weight decay** on STS from 0 to 1e-2.

5.4 Results

For **part 1**, we reached 0.423 and 0.816 as the pretrain dev set accuracy and 0.525 and 0.963 as the finetune dev set accuracy for both datasets.

For **part 2**, the finetuning accuracy on dev sets are listed in Table 1. **Multi-Bert baseline model** did not perform well on the three tasks, and one possible reason is that the tasks cannot be transferable learned with one backbone properly. When we **add more head layers** of PARA, we saw an increase of the PARA accuracy from 0.654 to 0.802 because of the improvement of the ability to analyze and compare the difference in embedding of sentences pair.

For Multi-Bert with **MNRLoss**, the model suffered from a performance degradation, which may be due to our naive design of MNRLoss and improper hyperparameter choices (e.g. scale and tanh), and we also found that maybe it is because MNRLoss is sensitive to the size and diversity of dataset.

For **Triple-Bert**, we can see an improvement compared to the baseline model, possibly because of the independence between three tasks, but accordingly, it requires more storage. For **Triple-Bert+2-input-structure**, it achieved **overal dev score = 0.773** and has a large improvement on PARA and STS. It demonstrates the positive effect of cross-attention between sentences.

Table 1: Performance comparison of different models

Model	SST dev accuracy	PARA dev accuracy	STS dev correlation	Overall dev score
Multi-Bert (Baseline)	0.478	0.654	0.371	0.606
Multi-Bert+more head layers in PARA	0.499	0.802	0.364	0.661
Multi-Bert+MNRLoss	0.294	0.426	0.038	0.413
Triple-Bert original	0.511	0.790	0.362	0.660
Triple-Bert+2-input-structure	0.511	0.883	0.850	0.773
SmartTri-Bert ($\lambda_s = 2, 6, 8$)	**0.550**	**0.897**	0.870	0.794
SmartMul-Bert (PARA+STS)	/	0.885	0.875	/
SmartMul-Bert (PARA+STS, one-loss)	/	0.866	0.874	/
SmartMul-Bert (SST+PARA+STS)	0.526	0.888	**0.877**	0.784
Ensemble	**0.550**	**0.897**	**0.877**	**0.795**

For **SmartTri-Bert**, we added SMART regularization to all three tasks of Triple-Bert, and firstly did an ablation study on the smart weight. As shown in Table 2, we can find that $\lambda_s = 2$ and $\lambda_s = 1$ achieve the best dev accuracy 0.550, which is also much better than that from Triple-Bert+2-input-structure model, so we can say that SMART plays an important role in avoiding overfitting and improving generalization and robustness. As shown in Figure 1, we can see a downward trend of

training accuracy when smart weight increases, which is because the level of regularization increases. For PARA and STS tasks, we found that $\lambda_s = 6$ and $\lambda_s = 8$ achieved the best results respectively, outperforming the Triple-Bert+2-input-structure model, demonstrating the broad applicability of the SMART method.

Table 2: Performance metrics for different smart weight λ_s on SST

λ_s	**Dev set accuracy**
0.5	0.532
1	**0.550**
2	**0.550**
4	0.542
5	0.542
6	0.546
7	0.536
8	0.537

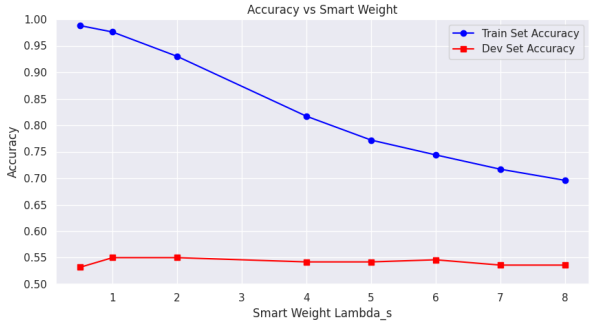

Figure 1: Dev set and Train set accuracy for different smart weight λ_s on SST

For **SmartMul-Bert for PARA and STS**, with the implementation of SMART regularization to avoid overfitting (as shown in the promising results with SmartTri-Bert), while PARA accuracy is only 0.885, STS correlation increases to 0.875, becoming higher than all results from previous models. Such quantitative results align with our expectations that training on PARA data with a large scale can indeed benefit the STS task. With synergies of the sentence pair comparison objectives of PARA and STS, the results indicate successful transfer learning from PARA to STS.

After fine-tuning for **SmartMul-Bert for PARA and STS with one loss**, we obtained the PARA dev set accuracy (0.866) and STS correlation (0.874). Notice that the multi-task learning approach, where the losses from both tasks were combined, did not outperform the sequential training procedure in SmartMul-Bert (PARA and STS). Although we hope to benefit from the richer representations derived from auxiliary tasks, we might need more powerful or nuanced optimization that can differentiate while optimize between the detailed specific needs for each task. Another observation we have for SmartMul-Bert (PARA and STS) with one loss as shown in Figure 3 is the overall gradual increase in both train and dev accuracies over the epochs. Such growth as well as the relatively steady difference between the train and dev set performances, without significant divergence, may suggest that the model is not overfitting, and moreover, it's learning effectively just at a gradual pace.

For **SmartMul-Bert for SST, PARA and STS**, as shown in Table 1, we also saw a performance degradation on SST and PARA compared to SmartTri-Bert, which may be due to the bad transfer learning between one-sentence-input task (SST) and two-sentence-input tasks (PARA and STS). However, the correlation of STS increases to 0.877, which is the best performance. It demonstrates that STS needs transfer learning from PARA (maybe also from SST), possibly due to the lack of data and diversity. Meanwhile, if we compare this model with our baseline Multi-Bert model, we can find a huge improvement. They are both trained on three tasks, with one Bert backbone, and so we can find the effectiveness of SMART and 2-input-structure to the transfer learning ability of the model.

Table 3: Test set performance metrics

SST test accuracy	**PARA test accuracy**	**STS test correlation**	**Overall test score**
0.557	0.896	0.876	0.797

Our **Ensemble** model used SmartTri-Bert for SST and PARA, and SmartMul-Bert(SST+PARA+STS) for STS to achieve the best ensemble performance on each task. As shown in Table 3, our Ensemble model achieved SST test accuracy=0.557, PARA test accuracy=0.896, STS test correlation=0.876, and overall test score=0.797, which ranks 2^{nd} on the test leaderboard by March 16th.

We also performed various experiments for architecture and hyperparameter tuning. Firstly, for **BERT hidden dropout ablation study**, we found that the best performance occurs when dropout=0.2, displaying a best generalization ability and avoiding overfitting. Secondly, after trying different **head layer dropout probabilities**, with dropout probability being 0 for both STS and SST did we achieve the highest dev set accuracy, 0.870 and 0.550 for each as shown in the following Table 5.

Meanwhile, we observe that as head dropout probability increases, both SST accuracy and STS correlation decreases monotonically, and the train performance and dev performance increases or decreases in the same direction. Two plots displaying this tendency are attached in appendix. Thirdly, we did **experiment for both one-layer and two-layer head** for all tasks with SMART regularization and other hyperparameters selected in advance, and gained the following results. One-layer yields accuracy of 0.536 and 0.894, as a better architecture for SST and PARA than the two-layer head (with 0.526 and 0.890). For STS task, with no weight decay for AdamW, two-layer structure renders a higher correlation (0.867) than that with one-layer (0.860); however, when we set the weight decay to be 1e-3, one-layer yields higher correlation (0.866) when compared with the two-layer's (0.861). Lastly, we experimented **different weight decays for AdamW optimizer**, with both 1e-4 and 1e-3 yielding the relative higher accuracy, 0.866. Detailed analysis are in the next section.

Table 4: Performance comparison of different Bert hidden dropout probabilities

Dropout Prob	Correlation (max)
0.0	0.839
0.1	0.846
0.2	**0.848**
0.3	0.843
0.4	0.821
0.5	0.732

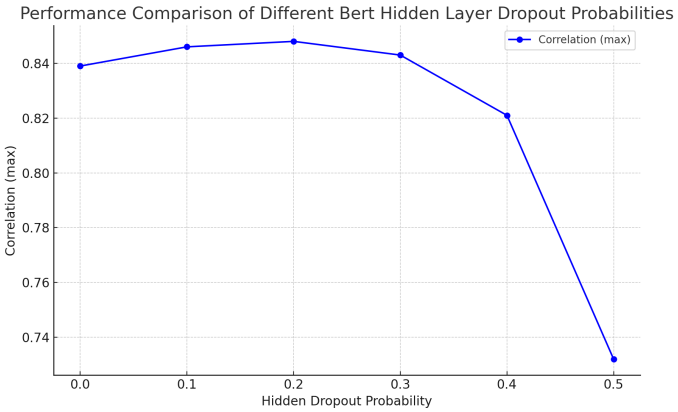

Figure 2: Correlation comparison plot of different Bert hidden layer dropout probabilities for STS

Table 5: Dev set STS correlation and SST accuracy for varying head dropout probabilities

Head Dropout	Dev sts corr	Dev sst acc
0.9	0.838	0.523
0.7	0.858	0.534
0.5	0.861	0.533
0.3	0.864	0.540
0.1	0.865	0.542
0	**0.870**	**0.550**

Table 7: STS Task Performance for Various Weight Decay Values

weight_decay	Dev set accuracy
0	0.860
1e-4	**0.866**
1e-3	**0.866**
1e-2	0.860

Table 6: Model Performance Across Different Head Layers and Parameters

Head Layer	SST smart_weight=8, weight_decay=0	PARA smart_weight=10, weight_decay=0	STS smart_weight=10, weight_decay=0	STS smart_weight=10, weight_decay=1e-3
one-layer	**0.536**	**0.894**	0.860	**0.866**
two-layer	0.526	0.890	**0.867**	0.861

6 Analysis

For our baseline **Multi-Bert** model, it is sequentially trained on three tasks, so the performance of the model highly depends on the similarity between tasks and the ability of the model to conduct transfer learning. At that time, without SMART, 2-input-structure, and hyperparameter tuning, our baseline model cannot adapt different tasks easily and lacks the generalization ability. Generalization plays a significant role in a model's resistance to differences in dataset distributions, including the distribution differences between training and test sets, as well as among different task datasets (SST,

PARA and STS). Therefore, after **adding SMART, 2-input-structure, and better hyperparameters** to our model, with the same task domain, **SmartMul-Bert** performed much better than Multi-Bert, and showed a better transfer learning ability between tasks, with a more generalizable and robust embedding generated by the Bert. Obviously, we can observe that regularization methods like SMART not only improve the development set accuracy of individual tasks but also enhance robustness to different dataset distributions, thereby boosting the capability of transfer learning across various tasks.

We found that **MNRLoss** didn't perform well, and there are some possible reasons. Firstly, there might be flaws in our implementation of MNR Loss. For the current dataloader, we opt to select positive/negative samples within a batch, leading to an inconsistent number of samples K, which could cause instability in the loss function, resulting in poor performance. Secondly, the efficacy of MNR Loss heavily depends on the choice of negative samples. If the selected negative samples are too similar to the positive ones (hard negatives), the model might overly focus on minor differences, reducing its generalization capacity. Conversely, if the negative samples are too dissimilar from the positive ones (easy negatives), the model might fail to learn effective features to distinguish between positive and negative samples. Additionally, employing MNR Loss may require the model to capture more complex data relationships, necessitating greater model capacity. Our model might suffer from insufficient capacity or a lack of diversity in the training data.

In addition, it is a big challenge to train **SmartMul-Bert with one single loss**. It turned out that we only have one objective for multiple tasks, which requires even stronger transfer learning ability of the model. Given that the gradients are back-propagated by one additive loss for different tasks, maybe in the future we need to set different loss weight for each task's loss and add them up with different weights. We can possibly take advantage of the reciprocal of the norm of the gradients of each task's loss, to calculate each loss weight, for more balance training between tasks.

With our Triple-Bert model, the decreasing correlation score when the **hidden layer dropout probability** becomes larger indicates possible missing information and underfitting. While dropout is a form of regularization that helps prevent overfitting by randomly deactivating a subset of neurons during training, its application as the **head dropout layer** on SmartTri-Bert suggests a different performance pattern than the hidden layer dropout. From Table 5, since increasing dropout leads to worse performance for both SST and STS, it may suggest that the model is already well-regularized at lower dropout rates, and further dropout only just deprives the model's ability to learn well. Therefore, this can be the results of the SmartTri-Bert, which already contains SMART implementation for the model to regularize. Our study on the **number of head layers** indicates that a two-layer structure, which is more complex, does not always guarantee superior performance over a single-layer approach. As described in results section, single-layer head could achieve slightly higher accuracy for all tasks with some pre-defined hyperparameters. This outcome indicates that it is possible to overcomplicate the model with no obvious benefits, so we need to be cautious and pay attention to balanced architecture when trying to extend or modify the model for better performance. Moreover, we investigated the **impact of the AdamW optimizer's weight decay** on STS task, and the empirical results suggest that with some weight decay values, higher dev set accuracy can be achieved, by moderately penalizing large weights during optimization. All of the architecture and hyperparameter tuning experiments underscore the intricate and delicate balance when either applying regularization techniques or adding more architectural complexities, thus making it crucial to find the "sweet spot" where the model is sufficiently regularized but without overcomplication.

7 Conclusion

Our BERT's Odyssey ends here. We tried different methods to tackle the difficulties in the journey, including SMART regularization, MultipleNegativesRankingLoss, Multi-task learning, Triple-Bert single task finetuning, architecture modification, and hyperparameter tuning. Finally, we build a robust Bert model, which not only achieve excellent results on SST, PARA, and STS tasks, but also shows fantastic ability of generalization and transfer learning between tasks. However, our BERT's Odyssey has some limitations, such as the lack of transfer learning ability between one-sentence-input task and two-sentence-input task, and also some failures on some extensions, MNRLoss and PALs-Bert (which is not shown on the paper).

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

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.

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

A Appendix

A.1 Additional Tables and Figures

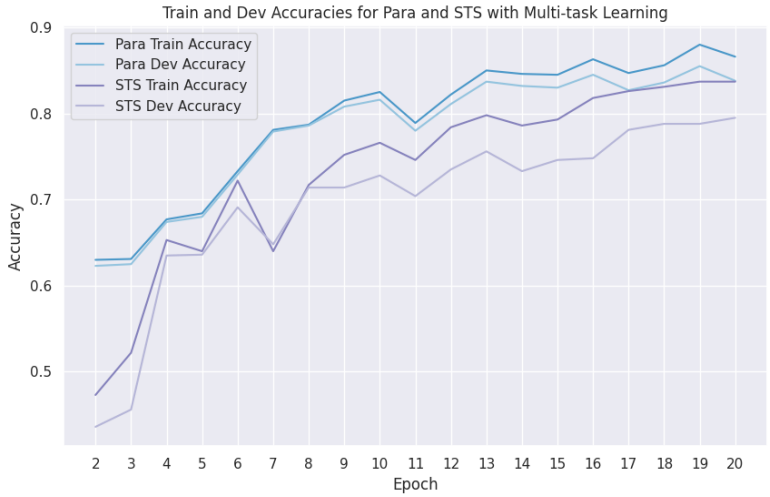

Figure 3: Train and Dev Accuracies for Para and STS (one-loss, without SMART)

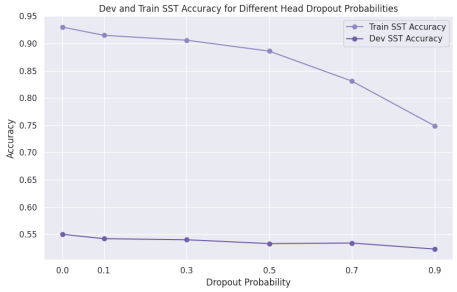

Figure 4: Train and dev SST accuracy with different head layer dropout probabilities

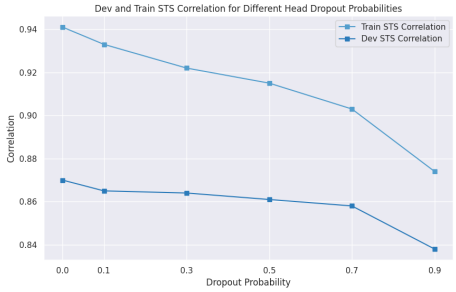

Figure 5: Train and dev STS correlation of different Bert hidden layer dropout probabilities