Enhancing Multi-Task Learning on BERT

Stanford CS224N Default Project

Yiming Ni
Department of Computer Science
Stanford University
yimingni@stanford.edu

Paris Zhang
Department of Computer Science
Stanford University
parisz@stanford.edu

Abstract

Leveraging transformer-based models like BERT for multi-task learning poses challenges due to the diversity of task objectives. In this project, we enhanced BERT's performance through multi-task learning on three downstream tasks, sentence sentiment classification (SST), paraphrase detection (PARA), and semantic textual similarity (STS). Our approaches include model architecture extensions, strategic fine-tuning regularization, data augmentation and additional training datasets, and ensemble methods. These approaches resulted in significant improvements, achieving an overall score of 0.773 on the test leaderboard. Despite these advancements, challenges remain in overfitting and difficulty in capturing nuanced linguistic meanings, suggesting future directions in refining sentence relationship encoding and exploring more effective regularization techniques.

1 Introduction

The advances of transformer-powered large pretrained models have revolutionized the field of natural language processing (NLP). Specifically, BERT (Devlin et al., 2019) exhibits remarkable capabilities in encoding deep sentence contexts through extensive pre-training, which, when fine-tuned with task-specific data, often results in impressive performance on downstream tasks. Multi-task learning has shown its promise for enhancing model robustness and generalizability by utilizing a broader spectrum of annotated data (Ruder, 2017). However, the process of optimizing shared model parameters across diverse tasks with diverging objectives can lead to optimization conflicts, making it difficult to excel at multiple tasks simultaneously (Liu et al., 2019).

In this project, we investigated multi-task fine-tuning on the BERT for three downstream tasks: sentence sentiment classification (SST), paraphrase detection (PARA), and semantic textual similarity (STS). Our approaches involved a series of strategic techniques, including implementing model architecture extensions, incorporating SMART regularization techniques (Jiang et al., 2020), expanding training data via data augmentation and additional datasets, and applying ensemble methods. These approaches aimed to strike a balance between task-specific demands and the model's robust and generalizable learning. Our enhanced model configuration showed a significant improvement of more than 0.1 on the dev set compared to baseline models and achieved an overall score of 0.773 on the test leaderboard. Finally, we qualitatively analyzed our model's performance, highlighting its strengths, acknowledging its limitations, and suggesting potential future works.

2 Related Work

BERT (Devlin et al., 2019) BERT is a large pre-trained model with bidirectional transformer encoders trained using Masked Language Model (MLM) and Next Sentence Prediction (NSP) objectives. For the MLM task, 15% of the original input is either replaced by random tokens from the vocabulary, masked, or unchanged, and the model needs to predict the original tokens for these modified ones. In the NSP task, on the other hand, the model is given two sequences where the second sequence is either the true next sequence or randomly selected from the source. The task objective is then predicting whether the pairs are adjacent in the original document. However, later studies show that pre-training with MLM only can achieve comparable or better performance.

Robust Finetuning To extend the capability of the large pre-trained models, various fine-tuning methods have been proposed. Direct fine-tuning relies on hyperparameter tuning to get the best performance. Typical techniques involve a smaller learning rate or a decayed learning rate for each transformer layer. Among them, Jiang et al. (2020) took a more systematic approach by introducing a smoothness-inducing regularization loss. They adversarially add a perturbation to each input and penalize the model for large changes in the output.

Data Augmentation Data augmentation techniques artificially increase the size and diversity of the training data through different transformations on the raw data, promoting model resilience against typical data alterations. As introduced in Raghu and Schmidt (2020), common data augmentation techniques in NLP include directly modifying the input sequences and sequence-to-sequence transformation, which is also known as back translation (Sennrich et al., 2015). When directly modifying the input sequence, popular perturbations usually involve randomly deleting a token, swapping sequence tokens, and replacing a token with its synonym, which is usually guided by word embeddings (Wang and Yang, 2015) or contextualized word embeddings (Kobayashi, 2018).

3 Approach

3.1 Baseline minBERT model

Our baseline model followed the original BERT model to add three task-specific heads after minBERT, one for each evaluation task. Each task head consists of a dropout layer followed by a linear layer. We used the pooled representation of the input, which is the final hidden state for the [CLS] token, as the contextual embedding for the sentence. For tasks involving sentence pairs, specifically PARA and STS, we separately passed the two input sentences through the BERT backbone and concatenated their embeddings as the input to the task heads.

We conducted two baseline experiment settings:

1. Jointly fine-tuning BERT and the three heads across three datasets for three tasks. This setting is analogous to the training configuration employed for our subsequent model extensions, thereby serving as a good comparative baseline.
2. Separately fine-tuning BERT for each task using the corresponding dataset. This experiment aimed to explore the potential upper bounds of the model performance on individual tasks.

3.2 Model Architecture Extensions

Multi-layer Task Heads To increase the expressive power of our task heads, we expanded beyond the initial configuration of a single dropout and linear layer. Each task head now has two additional linear layers with GeLU activation functions. We believe a more complex architecture within the task heads will facilitate a deeper understanding and more adept handling of task-specific nuances.

Inductive Bias for PARA and STS Given that both PARA and STS involve predicting sentence similarities, direct indicators of sentence relations, such as their differences or similarities, could significantly inform the prediction outcomes. Therefore, we introduced explicit inductive biases by:

1. Calculating the difference between the [CLS] token embeddings of the sentence pairs, and concatenating this difference vector with the original embeddings.
2. Computing the cosine similarity of the [CLS] token embedding, which is then concatenated with the embeddings.

We believe by adding the difference or cosine similarity as input features, the task heads of PARA and SST will be equipped with direct and salient indications of sentence relationships, thereby enhancing their ability to discern and quantify the degrees of similarity or paraphrase between sentence pairs.

Shared Task Heads between PARA and STS Observing a large difference in the data volumes between PARA and SST tasks, with PARA possessing 400,000 sentence pairs while SST only has 8,628, we adopted a strategy of sharing the intermediate layers in the task heads for the two tasks.

In this architecture, the two tasks used the same hidden linear layers, which captured the general semantic structures in comparing sentence pairs, and had their individual final output layers.

The shared task head aims to leverage the inherent similarities between the two tasks, aiming to construct a more adaptable and robust task head. At the same time, it tackles the challenge posed by skewed data distribution by allowing shared components to learn from a wider range of data. We believe this approach will enhance the model performance on both tasks by fostering a deeper and more generalized understanding of sentence similarity concepts.

Split BERT Layers into Task Heads Inspired by our previous finding that a more complex architecture of task heads can lead to better prediction results, We further increased this complexity by taking advantage of the capability of pre-trained transformers. Specifically, we allocated the last two layers of the BERT model to the individual task heads. With the strong contextual learning ability of the attention mechanism, we hope to drive the model to learn more nuances specific to each task while still maintaining the model's general linguistic ability.

3.3 Fine-tuning BERT with SMART methods

Because of the relatively small dataset size, our fine-tuning is prone to overfitting. As a solution, we incorporated the methods introduced in the SMART paper (Jiang et al., 2020) for robust and efficient fine-tuning. To mitigate excessive deviation during fine-tuning, we integrate a smoothness-inducing adversarial regularization term into the task loss function, formulated as: $\min_\theta = L(\theta) + \lambda_s R_s(\theta)$, where $R(\theta)$ is the regularization term defined by $R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{||\tilde{x_i}-x_i||_p \leq \epsilon} l(f(\tilde{x_i};\theta), f(x_i;\theta))$. Here, $\epsilon > 0$ is a tuning parameter, and l_s denotes the symmetrized KL-divergence for classification tasks and mean square loss for regression tasks.

Additionally, we incorporate Bregman Proximal Point Optimization methods, mitigating aggressive updates during fine-tuning. This adjustment foster more stable optimization steps through trust-region-type regularization, with the pre-trained model serving as the initialization point $f(\cdot;\theta_0)$. Consequently, the parameter update is defined as $\theta_{t+1} = \arg\min_\theta F(\theta) + \mu D_{Breg}(\theta, \theta_t)$, where $\mu > 0$ is another tuning parameter and $D_{Breg}(\theta, \theta_t) = l_s(f(\tilde{x_i};\theta), f(x_i;\theta_t))$.

We implemented SMART methods referencing the original paper's repo and a reimplementation repo.

3.4 Data Augmentation and Additional Datasets

Data Augmentation Recognizing the limitations imposed by the dataset's size, we adopted data augmentation strategies to enrich the training data both in volume and variety. Our approach involves introducing perturbations into the dataset by substituting tokens with their synonyms defined in WordNet (Miller, 1994) with a probability of 0.3 for any given token. To assess the impact of data augmentation volume, we experimented with augmenting 30%, 50%, and 100% of our dataset.

We implemented our data augmentation from nlgaug pacakge.

Additional Datasets To further diversify and enlarge our data, we incorporated two additional datasets as detailed in Section 4.1. By training on a more varied set of data with consistent tasks and annotations, the model is anticipated to develop a more profound and resilient understanding of the tasks at hand, thus performing better in terms of generalizability and robustness.

3.5 Ensemble

One of the crucial challenges in our project is that with limited data for tasks that are fundamentally different, the model is prone to overfitting. Our experiment showed that different model architectures and training configurations could improve on one task but likely with a trade-off with other tasks. For example, for PARA and STS, increased task head complexity gave an edge but the SST accuracy decreased for the same model. In fact, for SST, our separately finetuned BERT with a single linear layer had the best performance. To make these models complementary, we ensembled the outputs of different models.

Specifically, as illustrated in Figure 1, we passed the outputs of our best-performing models with different architectures to an ensemble network to get the final predictions. Because of the relatively

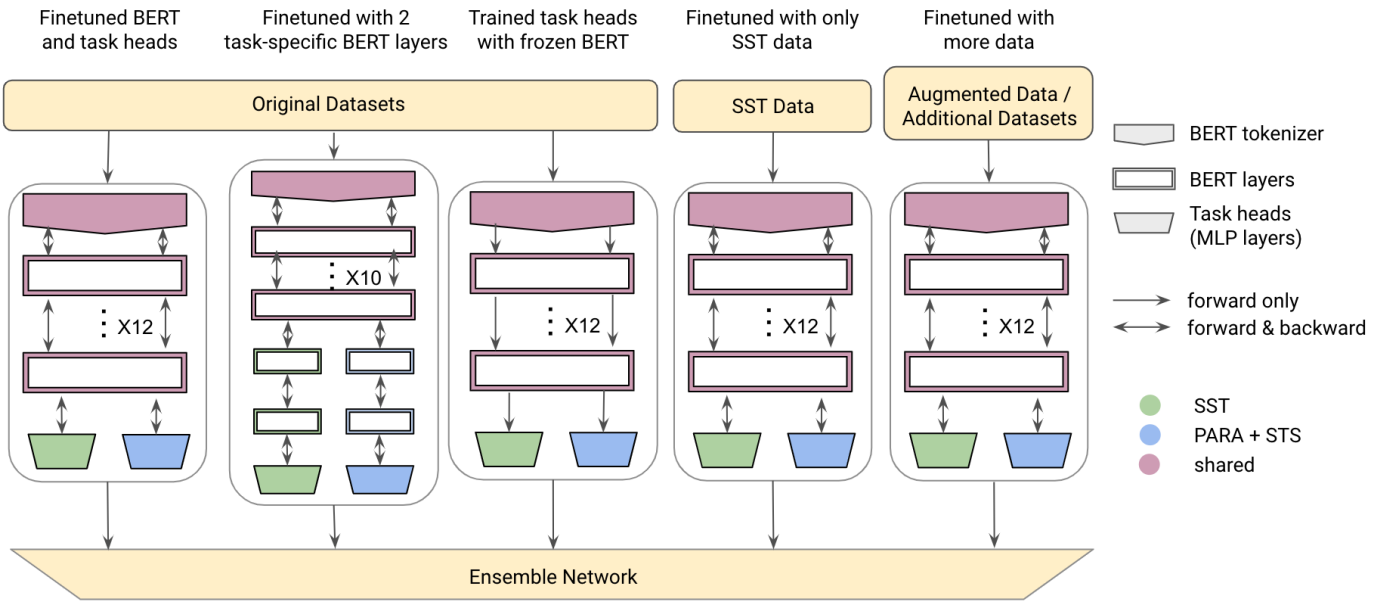

Figure 1: Ensemble model architecture

complicated architecture of BERT and task heads, we designed the ensemble network to be simple. We experimented with one linear layer, which was equivalent to a weighted average of the individual models, and two linear layers with a GELU activation.

4 Experiments

4.1 Data

We utilized five datasets, three of which were originally provided and the other two are additional.

The three originally provided datasets are Stanford Sentiment Treebank (SST) (Socher et al., 2013) for sentiment analysis, Quora dataset (Quo, 2017) for PARA, and SemEval STS Benchmark dataset (Agirre et al., 2013) for STS. The SST dataset comprises 11,855 sentences from movie reviews, further divided into 215,154 unique phrases annotated with sentiment labels ranging from negative to positive. The Quora data features 400,000 question pairs labeled for paraphrase identification. The SemEval dataset consists of 8,628 sentence pairs rated for similarity.

We added two additional datasets. For PARA, we added Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) with 5,801 sentence pairs annotated for paraphrase presence. For STS, we added SemEval SICK2014 dataset (Sem, 2014), which comprises 10,000 sentence pairs annotated for semantic relatedness. Both datasets are very similar to the original datasets in terms of task annotations and sentence contexts. Therefore, we hope adding them will provide a balanced mix of task specificity and sentence diversity, without compromising on the precision of task-tailored performance evaluations.

4.2 Evaluation method

The evaluation was mainly conducted on the dev sets of the three datasets. For sentiment analysis and paraphrase detection tasks, we used accuracy percentages. For STS, we used the Pearson correlation coefficient between predicted and true similarity scores. The overall performance was calculated by averaging the three scores after normalizing STS correlation.

4.3 Experimental details

All the experiments were run with our minBERT and AdamW implementations from part 1. The different model architectures are shown in Figure 1, with experiment details specified in each

subsection in Section 4.4. We set the learning rate as $1e-5$ for fine-tuning, and $1e-3$ for training only the task heads. Each individual model was run for 40 epochs, and the ensemble network was run for 20 epochs.

4.4 Results

4.4.1 Model Architecture Extensions

We conducted extensive experiments with our model architecture extensions, including multi-layer task heads, adding sentence differences or cosine similarities as input features, sharing task heads between PARA and STS, and splitting BERT layers into each task head.

As shown in Table 1, the introduction of multi-layer task heads significantly improved performance on PARA and STS tasks with a slight decrease in SST accuracy. This discrepancy may be attributed to the inherently straightforward nature of sentiment analysis, which aligns well with sentence embeddings, so single-layer classifiers are sufficient and less prone to overfitting. Conversely, PARA and STS entail more complexity due to their two-sentence inputs, where multiple layers prove more effective in discerning underlying relationships.

Furthermore, concatenating the sentence embedding differences to the task-head inputs significantly boosted STS performance, as they are direct indicators of sentence relations and highly correlated with textual similarity. However, the addition of the cosine similarity measure did not yield further performance gains, likely due to its redundant nature when compared to the more informative sentence difference vector.

Sharing the task heads between PARA and STS resulted in a notable improvement in STS performance without compromising PARA accuracy. These findings suggest that finetuning BERT and the task head across various tasks and datasets enhances its general sentence encoding capabilities, thereby improving its overall performance.

However, the approach of splitting two BERT layers into each task head did not lead to performance enhancements, potentially due to overfitting issues. It is also important to highlight that the baseline model achieved the best accuracy on the SST task when BERT was fine-tuned solely with a single-layer task head on the SST dataset. This observation validates our hypothesis that different tasks may require different optimal model configurations, implying that enhancing performance in one task may come at the expense of others.

Experiment	SST Acc	PARA Acc	STS Corr	Overall score
single-layer, separate (baseline)	**0.523**	0.785	0.383	0.667
single-layer, joint (baseline)	0.496	0.771	0.344	0.646
multi-layer, separate (baseline)	0.510	0.800	0.370	0.665
multi-layer, joint	0.503	0.813	0.402	0.673
multi-layer, joint + diff	0.494	**0.815**	0.643	0.710
shared-multi-layer, joint + diff	0.505	**0.818**	**0.852**	**0.750**
shared-multi-layer, joint + diff + cosine	0.508	0.798	**0.850**	0.744
shared-multi-layer w/ split BERT, joint + diff	0.498	0.812	0.844	0.744

Table 1: Results on architecture extensions

4.4.2 Data Augmentation and Additional Datasets

Based on the most effective model architecture in Section 4.4.1, we further explored the impact of data augmentation and the integration of real datasets on model performance.

As shown in Table 2, data augmentation improved PARA accuracy the most. Since we augmented the data by randomly substituting words with their synonyms, it effectively generated a wider array of paraphrasing examples, enhancing the model's ability to recognize paraphrased content. Nevertheless, the benefits of data augmentation diminished when applied excessively, with a 100% or

50% augmentation rate adversely affecting performance on SST and STS tasks. This decline is likely due to the additional noise introduced by the synthetic data into the training process.

The inclusion of real datasets yielded mixed results. Incorporating MRPC led to improvements in PARA accuracy while adding SICK increased STS performance. These outcomes align with expectations, given the relevance of these datasets to their respective tasks. However, improvements on one task were often counterbalanced by declines in others. In addition, simultaneously adding both MRPC and SICK resulted in a significant decrease in PARA accuracy. This suggests that optimizing BERT and task heads with a large volume of external data may cause the model to diverge from its optimal state for the original datasets.

Experiment	SST Acc	PARA Acc	STS Corr	Overall score
shared-multi-layer, joint + diff (base setting)	**0.505**	0.818	**0.852**	0.750
base + 30% augmented data	**0.507**	0.846	**0.847**	**0.759**
base + 50% augmented data	0.496	**0.858**	0.835	**0.757**
base + 100% augmented data	0.474	0.842	0.832	0.744
base + MRPC dataset	0.486	0.851	0.827	0.750
base + SICK dataset	0.499	0.844	**0.850**	**0.756**
base + MRPC & SICK dataset	0.495	0.809	**0.850**	0.743

Table 2: Results on data augmentation and additional datasets

4.4.3 SMART Fine-tuning Methods

We applied SMART fine-tuning regularization methods to some experiment settings in previous sections. As shown in Table 3, SMART slightly reduced performance, especially for models trained with original datasets. For the model trained with 50% augmented data, SMART had no significant effect. We suspect that even though SMART encourages a more robust fine-tuning by mitigating aggressive updates, our complex task-specific model architecture and the constrained size of the datasets limit the effectiveness of such regularization techniques in this context.

Experiment	SST Acc	PARA Acc	STS Corr	Overall score
multi-layer, joint	0.503	0.813	0.402	0.673
+ SMART	0.497	0.808	0.392	0.667
shared-multi-layer, joint + diff	0.505	0.818	**0.852**	0.750
+ SMART	0.510	0.813	0.484	0.688
shared-multi-layer, joint + diff + 50% data	0.496	**0.858**	0.835	**0.757**
+ SMART	0.495	0.852	0.837	**0.755**

Table 3: Results on SMART techniques

4.4.4 Ensemble

We trained ensemble networks to combine various model configurations as illustrated in Figure 1.

Our initial ensemble consisted of 4 distinct model configurations trained with original datasets: (1) a base setting of fine-tuned BERT with shared-multi-layer task heads, jointly trained on three datasets; (2) the base setting fine-tuned with additionally 2 BERT layers allocated to the task heads; (3) a variant where only the task heads were trained with BERT model frozen; and (4) a model fine-tuned only on SST data with single-layer task heads. Within the ensemble framework, we experimented with both a single linear layer and two linear layers with a hidden size of 64. As illustrated in Table 4, this ensemble approach yielded a significant improvement, increasing the overall score by 0.01 over our previously best-performing model in Table 1.

Experiment	SST Acc	PARA Acc	STS Corr	Overall score
models w/ original data, linear layer	**0.519**	0.834	0.859	0.761
models w/ original data, 2 layers, dim 64	**0.515**	0.838	0.858	0.761
+ models w/ augmented data, 2 layers, dim 64	0.513	0.843	**0.861**	0.762
+ models w/ augmented & real data, linear layer	0.512	0.856	**0.860**	**0.766**
+ models w/ augmented & real data, 2 layers, dim 16	0.509	0.856	**0.865**	**0.766**
+ models w/ augmented & real data, 2 layers, dim 32	0.510	**0.863**	**0.865**	**0.768**
+ models w/ augmented & real data, 2 layers, dim 64	0.514	0.856	**0.864**	**0.767**

Table 4: Results on ensemble

Expanding upon this foundation, we further enriched the ensemble with models trained on both augmented data and additional datasets. This incorporation led to consistent performance enhancements for the PARA and STS tasks over all the previous individual models, underscoring the robustness and effectiveness of the ensemble approach. Upon comparing different numbers of layers and hidden sizes, we found that 2 linear layers with 32-dimension hidden size worked slightly better, possibly due to a balance between expressiveness and a reduced risk of overfitting.

4.4.5 Test Leaderboard Result

Our performance on the test leaderboard is shown in Table 5. We chose our best-performing ensemble setting "+ models w/ augmented & real data, dim 32" setting from Table 4).

SST Acc	PARA Acc	STS Corr	Overall score
0.531	0.86	0.859	0.773

Table 5: Test leaderboard result

5 Analysis

We evaluated our model's performance qualitatively by examining the dev set predictions.

The leftmost plot in Figure 2 reveals a notable trend in SST task where most of the errors are minor, typically deviating by only one class. The model exhibits a conservative stance in sentiment classification, often opting for "somewhat positive" or "somewhat negative" labels instead of extreme sentiments. This cautious approach somehow mirrors human behavior in uncertain situations, where a more neutral classification is preferred. However, some examples also suggest that the model struggles with more complex linguistic features such as sarcasm or double negation, leading to significant misclassifications. Those sentences require a more comprehensive understanding of the full sentence instead of individual words. For instance, the sentence *"If Steven Soderbergh 's ' Solaris ' is a failure it is a glorious failure ."* has a positive sentiment but is predicted as "somewhat negative". The model probably made the prediction based on the word *"failure"* but failed to capture the sentiment of the whole sentence.

For the PARA task, as demonstrated in the middle plot of Figure 2, errors predominantly occur in non-paraphrase classifications, particularly when sentence pairs differ only slightly but enough to alter the meaning completely. For example, *"Who is the most intelligent person alive today?"* is predicted as a paraphrase for *"Who are the most intelligent people in the world?"*. Similarly, amongst the false negatives, a lot of the sentence pairs have different keywords and tones despite the same meaning. For example, the sentences *"Why does the skin on our palms and soles of the feet have less pigmentation than other parts of the body?"* and *"Why are our palms and the soles of our feet fairer than the rest of our body?"* are paraphrases. However, our model predicted it wrong, probably because the first sentence uses more scientific terms whereas the latter is more conversational.

From the rightmost plot in Figure 2, the model's predictions on STS also demonstrate a reliance on word overlap. For sentence pairs with the same label, the predictions can deviate by at most 2.5. For

example, *"Work into it slowly"* and *"It seems to work"* have no semantic textual similarity, while our model predicted a score of 2.5. Such errors underscore the model's challenge in capturing the deeper semantic relationships beyond mere word presence, especially in sentences with minimal context or strong word associations. This is similar to its reasoning on PARA predictions, which is expected because the two tasks share the same intermediate layers.

Across the three tasks, the model tends to prioritize word-level meanings and associations over comprehensive contextual interpretation. While pre-trained BERT parameters provide a foundation for sentence processing, the model's effectiveness is diminished in scenarios requiring intricate linguistic comprehension or when faced with limited training examples. This suggests a need for further refinement in model training and architecture to better accommodate complex sentence structures and nuanced language use.

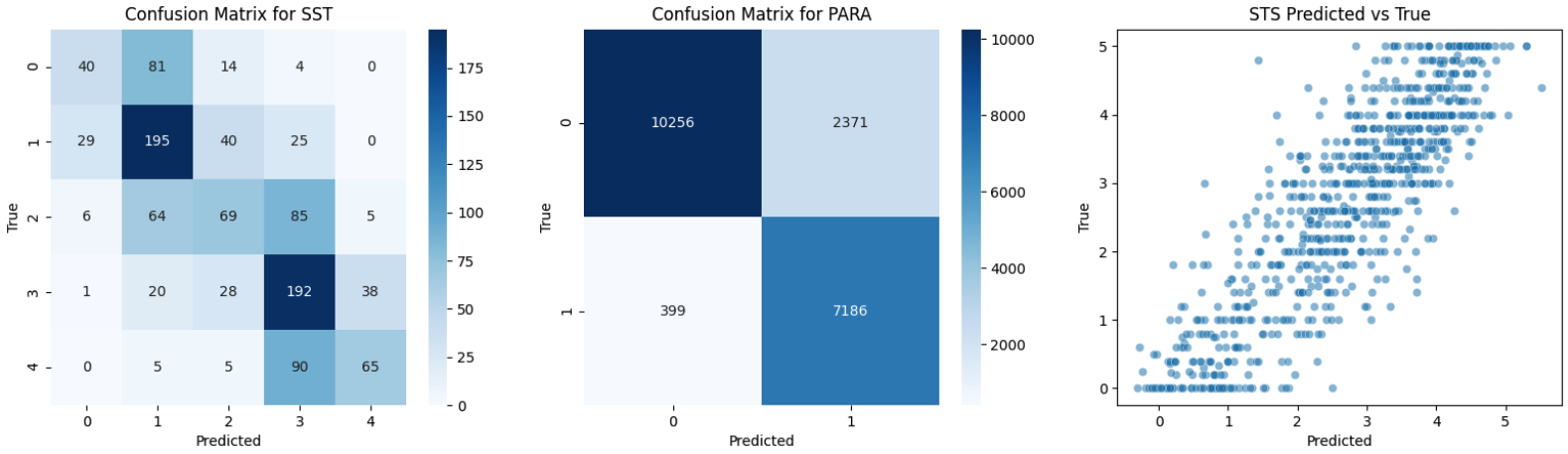

Figure 2: Qualitative analysis

6 Conclusion

In this project, we systematically investigated BERT's performance in multi-task learning and proposed several methods to enhance its performance over three different tasks: SST, PARA, and SST. Through a series of strategic extensions including multi-layer task heads, inductive biases for sentence relationships, and shared task heads between tasks, we significantly improved over the baseline by 0.08. To address the challenge of insufficient task-specific training data, we expanded the dataset through augmentation and incorporating new datasets, which yielded another improvement of 0.01. Furthermore, our ensemble method integrated various model configurations and contributed to an overall performance boost of 0.01.

However, we frequently observed the pattern of overfitting during our fine-tuning. SMART regulation was less effective in preventing it due to the constrained data and relatively large model size. We also identified some limitations of our model, such as its conservative sentiment classification, its difficulty with complex linguistic features, and an overreliance on word-level cues over deeper contextual understanding. Potential future work includes refining sentence relationship encoding, expanding training datasets, and exploring more effective regularization techniques to mitigate overfitting.

7 Other Key Information

Mentor: Anirudh Sriram. No external collaborators or sharing project.

Team Member Contributions:

- Yiming: BERT and AdamW implementations, Multi-layer task heads, Split BERT layers into task heads, Ensemble methods
- Paris: Inductive bias, Shared task heads, SMART implementations, Data augmentation & additional datasets, Running all experiments

References

2014. Task description: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment.

2017. First quora dataset release: Question pairs.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

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

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

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.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*.

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*.

George A. Miller. 1994. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Maithra Raghu and Eric Schmidt. 2020. A survey of deep learning for scientific discovery. *arXiv preprint arXiv:2003.11755*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.