Jack of All Trades, Master of Some:
Improving BERT for Multitask Learning

Stanford CS224N Default Project
Gradescope Name: BBB or B3 or B^3: Blackity-Black Blacks, (because we're Black)
Mentor: Timothy Dai

Chijioke Mgbahurike
Department of Computer Science
Stanford University
cmgbahur@stanford.edu

Iddah Mlauzi
Department of Computer Science
Stanford University
iddah@stanford.edu

Kwame Ocran
Department of Computer Science
Stanford University
kano@stanford.edu

Abstract

In an effort to achieve high and even accuracies across all 3 sub-tasks, we investigate the effects of various architecture modifications. Performing hyperparameter tuning to each changes, we ultimately find that the combination of all our changes: Annealed Sampling, Smart Loss Regularization, Bergman Point Optimization, and Multiple Negative Ranking loss, greatly improves baseline performance whilst lowering the overall time required to finetune. Our best ensembled model achieved an accuracy of 79.8% on the test set.

1 Introduction

Language systems are a robust problem space due the relative ease at which human beings acquire and reproduce language. With seemingly no effort, human beings are able to plan, understand, and complete a variety of tasks via linguistic communication. Initial attempts at computationally replicating language systems for multitask settings relied on rule-based systems crafted by domain experts, but such systems lacked linguistic robustness, were expensive to maintain and update, and lacked generalizability to other linguistic tasks. One reason for such failures was of a lack of linguistic representation. These rule-based systems could not capture all the nuanced meaningful relationships between words, sentences, and their enormous representations. Recent advances in NLP (namely Transformer-based foundation models) paved the way towards robust and well-performing language systems via the creation of powerful word embeddings which capture greater linguistic complexities Liu et al. (2019). Bidirectional Encoder Representations from Transformers (BERT) Devlin et al. (2019)is one such foundation model. The current training paradigm finetunes a foundation model on a specific task resulting in improved task accuracy often at the cost of degraded performance for other linguistic task. As such, multitask learning seeks to improve model robustness across a range of tasks using the same model weights for each task, hopefully creating a less task-dependent model embedding.

In our report we investigate multitask learning in a minimal BERT model with three adaptations: Bregman Proximal Point Optimization (Jiang et al., 2020), SMART regularization (Jiang et al., 2020), and Multiple Negatives Ranking loss (Henderson et al., 2017). We also implement an annealed sampling training schema to prevent overfitting and underfitting from tasks with smaller and larger datasets respectively.

2 Related Work

Multitask learning has been extensively explored within NLP, aiming to improve model generalizability by leveraging shared representations across related tasks. In (Peng et al., 2020), researchers train a model using clinical and biomedical data, outperforming current SOTA models. Likewise, (Caruana, 1997) investigated various multitask learning architectures, demonstrating the potential for performance gains, but challenges still remain in balancing the learning of different tasks and preventing negative transfer from one task to another.

Overfitting and underfitting are common challenges in multitask learning, particularly when dealing with datasets of varying sizes. In (Stickland and Murray, 2019), researchers explore a unique training schema to gradually introduce all tasks during training, balancing the negative effects of different-sized datasets. This ensures more even training across tasks, particularly for those with less data.

In Jiang et al. (2020), researchers implemented Smoothness Inducing Adverserial (SMART) regularization and Bregman Proximal Point Optimization (BPPO) to reduce overfitting and improve transfer learning. Using a combination of these two methods, they consistently outperformed the base BERT model across all 8 GLUE tasks by an immense margin, demonstrating improved model generalization in multitask domain.

Lastly, researchers (Henderson et al., 2017), developed Multiple Negatives Ranking loss (MNRL) which is designed to improve the model embeddings by mapping similar sentences closer together and dissimilar sentences farther apart.

By building upon these existing approaches, our research aims to develop a multitask fine-tuning method for BERT that has even performance across the 3 tasks while still having decently short training times.

3 Approach

As a baseline, we implement a multi-task BERT model. The BERT layers are shared across all three tasks. We add a simple linear layer with dropout for each of the tasks. For tasks involving sentence pairs, we use a cross-encoder architecture which performs full-attention over the input pair following the insights from Nandan Thakur and Gurevych (2020) that this architecture outperforms a bi-encoder. To finetune our model, we use a round robin approach, cycling though the datasets sequentially.

We then develop implementations of each extension, performing hyperparameter tuning at each step to determine which parameters result in the most evenly spread improvements across all three subtasks.

3.1 Annealed Sampling

Upon evaluating our base model, we observed significant accuracy discrepancies across subtasks, with paraphrase detection achieving the highest, followed by semantic textual similarity, and sentiment classification trailing. This disparity stemmed from the varying data volumes used for training each task—141,498 examples for paraphrase, 8,544 for sentiment analysis, and 6,040 for semantic textual similarity, highlighting a limitation of the base model's round-robin training approach. This training method samples batches from each datsset in a predetermined order, but as Stickland and Murray (2019) highlights, it could negatively impact training by exhausting smaller datasets before larger ones are adequately sampled.

To address this, we considered augmented the smaller datasets with more examples but that would lead to inflated training times. We further considered truncating the larger paraphrase dataset, however we wanted to utilize as much data as possible for training.

Annealed sampling can help mitigate these issues by selecting a given task based on the size of task's dataset. Concretely we model this as:

$$p_i \ \alpha \ N_i^{\alpha}$$

where p_i is the probability of selecting a batch of examples from task i and N_i is the number of training examples in task i. We use the same definition for α as Stickland and Murray (2019):

$$\alpha = 1 - 0.8\frac{e-1}{E-1}$$

Where e is the current epoch and E is the total number of epochs. This scheme has the effect of training tasks with larger datasets towards the beginning of training, and as the current epoch increases, we select a training task with more equal probability. As such, no task overly dominates towards the end of training, helping prevent overfitting and underfitting accordingly.

3.2 Smoothness Inducing Adversarial Regularization (SMART Loss)

We observed that the base model displayed high training accuracies but significantly lower development accuracies, indicating overfitting. To mitigate this, we implemented Smoothness Inducing Adversarial Regularization (SMART) loss.

SMART loss is a regularization technique that encourages the output of the model not to change much, when injecting a small perturbation to the input. This helps to prevent overfitting and enhances the model's ability to generalise on low resource domains. To do this, the method attempts to solve an optimization problem defined by Jiang et al. (2020) as:

$$min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda \mathcal{R}(\theta) \tag{1}$$

where, $\mathcal{L}$ is the task dependent loss function already used in the base model, $\lambda > 0$ is a tuning parameter, and $\mathcal{R}$ is defined as follows:

$$\mathcal{R} = \frac{1}{n}\sum_{i=1}^{n} max_{||\tilde{x_i}-x_i||_p \leq \epsilon} l(f(\tilde{x_i}, \theta), f(x_i, \theta))$$

.

where $\tilde{x_i}$ denotes the perturbed embedding, generated by adding gaussian noise to the original embedding. Following Jiang et al. (2020), we use symmetric KL-divergence as the loss function for classification tasks and mean-squared loss for regression tasks.

Informed by findings by Hayes (2023) that the effectiveness of SMART loss diminishes due to noise from dropout layers, we opted to exclude dropout in our SMART loss implementation to preserve the integrity of the adversarial loss calculation.

3.3 Bregman Proximal Point Optimization (BPPO)

To further prevent overfitting, we implement Bregman Proximal Point Optimization (BPPO) to solve equation 1. The BPPO method introduces a trust-region-type regularization at each iteration and moderates updates by ensuring they occur within a small neighborhood of the previous iterate. This strategy helps prevent aggressive updates to model parameters and preserves the knowledge of the out-of-domain data in the pretrained model, which we hoped would be useful in the transfer learning process by balancing the need for new learning with the retention of pre-existing knowledge. We accelerate the BPPO method by introducing an additional momentum to the update. Specifically, we update the parameters θ by:

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

where $F(\theta)$ is the loss calculated in 1, μ is a tuning parameter, and D_{Breg}, the Bregman divergence is given by:

$$D_{\text{Breg}}(\theta, \tilde{\theta}_t) = \frac{1}{n}\sum_{i=1}^{n} l\left(f(x_i;\theta), f(x_i;\tilde{\theta}_t)\right).$$

l is the same as described in section 3.2 and $\tilde{\theta}_t$ is the exponential moving average, defined as:

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

where β is a tuning parameter.

Thus, Bregman stops a model from diverging too far from its old parameters in 2 ways. First, β restricts how far the current parameters can diverge with each new update, and second, μ punishes the model for diverging.

3.4 Multiple Negative Ranking Loss

Following the implementation of the above extensions, we noted marked improvements in the paraphrase detection and semantic similarity tasks. In contrast, sentiment analysis showed only marginal gains, regardless of the hyperparameters used. We aimed to refine sentiment analysis by enhancing the model's embeddings, allowing it to grasp subtler distinctions between sentence embeddings with different sentiments. To do this, we implement Multiple Negatives Ranking Loss (MNRL).

MNRL tries to fine-tune the model to better discern similar and dissimilar sentences. This is accomplished by ensuring pairs of sentences that are labeled similarly are represented by embeddings that are close together in feature space. Conversely, for pairs of sentences that are not similar, the model is trained to push their embeddings further apart.

We adapt MNRL to cluster embeddings of sentences based on their sentiment. In each training batch, we select $2K$ sentences, where K is the number of sentiment class, to then construct two matrices A, B where a_i and b_i have the same sentiment if and only if $i = j$:

$$A = \begin{bmatrix} a_1\text{'s normalized embedding} \\ \vdots \\ a_K\text{'s normalized embedding} \end{bmatrix}, B = \begin{bmatrix} b_1\text{'s normalized embedding} \\ \vdots \\ b_K\text{'s normalized embedding} \end{bmatrix}$$

We compute a scaled similarity matrix $S = CAB^\top$, with a scaling Factor, C, to prevent vanishing values. Thus, I, a $K \times K$ identity matrix, is the ideal S, such that function $\mathbf{J}(S, I)$ is the cross-entropy loss:

$$\mathbf{J}(S, I) = -\frac{1}{K} \sum_{i}^{K} I_i \log(S_i)$$

4 Experiments

4.1 Data

We use the following datasets for multitask learning:

- The Stanford Sentiment Treebank Dataset (SST) (Socher et al., 2013) consists of movie reviews (9,645 examples) from Rotten Tomatoes. Specifically, we employ fine grained sentiment analysis, SST-5, where sentiments range from 0 (negative) - 4 (positive).
- Quora Question Pair Dataset (QQP) (161,710 examples) for binary paraphrase detection where a single question pair, consisting of two sentences, are labeled as (1) paraphrases of each other or (0) not paraphrases of each other.
- SemEVAL STS Benchmark Dataset (STS) (6,903 examples) for semantic textual similarity. Given two sentences, labels range from 0 (not related) through 5 (related).

4.2 Evaluation Method

We measure the model's performance by averaging three metrics: the accuracy on SST, the accuray on QQP and the Pearson correlation for STS normalized between 0-1. We place a premium on models that exhibit consistent performance across all three tasks, rather than those that excel in one area while underperforming in others, even if their overall averages might be equivalent.

4.3 Experimental Details

For all experiments, unless otherwise stated, we finetune our model keeping fixed a learning rate of 1e-5 and using the ADAMW optimizer with a weight decay of 1e-4. We use no dropout in the classification heads following the observation from Hayes (2023) that this improves performance across the three tasks. We report the best dev accuracy over 10 epochs.

4.3.1 Batch Size and Epoch Steps for Annealed Sampling

We first experimented with batch size and epoch steps for a finetuned base BERT model with annealed sampling.

Batch Size	Epoch Steps	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
8	1000	0.848	0.522	0.932	0.766
16	1000	0.868	0.521	**0.934**	0.774
8	2000	0.859	0.522	0.933	0.770
16	2000	0.872	0.525	0.932	0.776
8	3000	**0.876**	**0.529**	0.928	**0.777**
16	3000	0.863	0.520	0.931	0.771
8	4000	0.871	0.520	0.930	0.774
16	4000	0.875	0.520	0.927	0.774

Table 1: Effect of epoch steps, batch size on dev model accuracy

We find that using a using a batch size of 8 with 3000 steps per epoch results in a similar model accuracy and training time as using a batch size of 16 with 2000 steps per epoch. While the configuration with a batch size of 8 performed the best for QQP and SST, it significantly underperformed for STS. Ultimately, we choose to use a batch size of 16 with 2000 steps per epoch as this configuration consistently ranked amongst the top three in terms of accuracy. This choice also drastically reduced the finetune time from 10 hours for the baseline model, to 90 minutes.

4.3.2 SMART Loss λ

Fixing a batch size of 16 with 2000 steps per epoch, we implement Jiang et al. (2020)'s SMART loss. Using their recommended 1 sampling step, $\epsilon = 1e-6$, $\eta = 1e-3$, $p = \infty$, we experiment with different choices for λ.

Lambda	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
1	0.875	**0.536**	0.927	0.779
3	0.876	0.533	0.935	0.781
5	0.877	0.523	0.937	0.779
7	**0.878**	0.522	0.938	0.780
10	0.877	0.531	0.937	**0.782**
15	0.876	0.532	**0.939**	**0.782**

Table 2: Effect of lambda in SMART Loss regularizer on dev model accuracy

As shown in Table 2, we find that $\lambda = 10$ and $\lambda = 15$ perform well. Upon retraining with these λ values for 15 epochs, we observe, as seen in Figure 3, that the configuration utilizing $\lambda = 15$ outperformed the alternative across all three tasks. This led us to adopt $\lambda = 15$ for future experiments in anticipation of extending training durations when experimenting with the MNRL-finetuned model.

Lambda	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
10	0.874	0.524	0.938	**0.781**
15	0.874	**0.528**	**0.939**	**0.781**

Table 3: Effect of lambda in SMART loss regularizer on dev model accuracy over 15 epochs

4.3.3 Multiple Negative Ranking Loss Models

Fixing a λ of 15, we use annealed sampling and SMART loss to finetune the BERT model that we further pretrained on the SST dataset using MNRL. We anticipate an improvement in sentiment analysis using this model.

In our experiments, we observe the highest dev accuracy in epoch 10. This observation led us to theorize that extending the training duration would improve the accuracy levels. Consequently, we further finetuned our MNRL-finetuned model for 15, 25, and 50 epochs respectively. An interesting point to note is that our MNRL-finetuned model had never performed inference for any of SST, STS, and QQP tasks.

Epochs	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
15	0.877	0.536	0.940	0.784
25	**0.887**	**0.546**	0.935	**0.792**
50	0.877	0.542	**0.941**	0.787

Table 4: Effect of epochs in MNRL-finetuned finetuning

Although training for 25 epochs increased our model accuracy, we saw diminished results when finetuning for 50 epochs.

4.3.4 Bregman Proximity Point Optimization

We implement BPP Optimization from base BERT and finetune for 10 epochs with a 1e-4 learning rate. We experiment with different values of μ and β. As seen in table 5, the model accuracy decreased. Thus, we implement a learning rate scheduler that decreases the learning rate by a factor of .5 after every three consecutive epochs with increased total losses. We observe a slight improvement in accuracy, however, the model still underperforms in comparison to non-BPPO variants.

LR Scheduler	μ	β	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
0	0.3	0.75	0.876	0.512	**0.940**	0.777
0	3	0.3	0.875	0.499	0.938	0.771
1	0.3	0.75	0.876	**0.520**	0.938	**0.778**

Table 5: BPPO using base BERT model

Since BPPO aims to stop the model parameters from deviating too much from their original configuration, we posit that initiating the process with parameters already proven to excel across the three tasks would optimize BPPO's effectiveness. Consequently, we select our most successful model to date—the model fine-tuned for 25 epochs with MNRL—as the starting point for BPPO. Again, we vary μ and β.

We initially chose high values of 3 for μ and 0.99 for β in the hopes that this would stop the model from deviating too much from the original parameters which we had proven to be effective. We find that this leads to a slight improvement in the model accuracy however, the loss increases rapidly during each epoch, indicating potential overfitting. We thus experiment with lower values of β and μ to mitigate this. We find that reducing μ improves model performace and observe no differences when keeping μ constant while reducing β.

In all experiments we use a learning rate scheduler and reduce the number of steps per epoch to 1000 in the hopes that training less in each epoch would help reduce overfitting. In the final experiment, we reduce the intial learning rate to 1e-6, enabling the model to proceed with more cautious updates. We further extend the training duration to 25 epochs following the prior observation that an increased number of epochs improved the results. We observe the best results with this configuration.

Epochs	Learning Rate	μ	β	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
10	1e-5	3	0.99	0.888	0.548	0.943	0.793
10	1e-5	1	0.9	**0.892**	0.551	0.944	0.796
10	1e-5	1	0.75	**0.892**	0.549	0.944	0.795
25	1e-6	1	0.75	0.891	**0.553**	**0.946**	**0.797**

Table 6: BPPO using the 25-epoch-finetuned MNRL model

4.4 Results

In our final step, we ensemble all the models developed to this point. For the SST, QQP, and STS tasks, we apply accuracy thresholds of over 0.53, 0.88, and 0.87, respectively, including only models that surpass these benchmarks. We report the results in Table 7.

Model	Dev QQP Acc	Dev SST Acc	Dev STS	Model Acc
Baseline Bert	0.884	0.504	0.85	0.746
Baseline +SMART	0.865	0.524	0.936	0.775
Annealed	0.872	0.525	0.932	0.776
Annealed +SMART	0.876	0.532	0.939	0.782
MNRL + Annealed + SMART	0.877	0.546	0.935	0.792
MNRL + Bregman	0.891	0.553	**0.946**	0.797
Ensemble	**0.895**	**0.560**	0.945	**0.800**

Table 7: Evaluation of Model performance

We evaluate our ensemble on the test set and observe the following results

QQP Acc	SST Acc	STS	Model Acc
0.895	0.556	0.944	0.798

Table 8: Evaluation of Ensemble on the Test S

5 Analysis

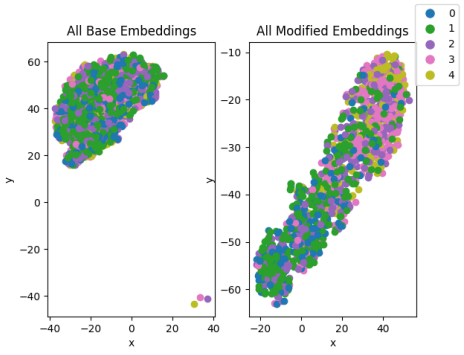

Figure 1a: Embedding Graph

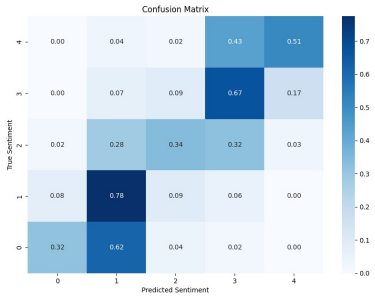

Figure 1b: Final Model's Normalised Confusion Matrix

5.1 MNRL

To analyze the effect of MNRL, we used t-SNE to visualize all the sentence embeddings from the baseline BERT model and a model fine-tuned with only MNRL. We projected all the development

examples in the SST dataset onto a 2D graph, figure (1a). Due to t-SNE's high plotting variability, we ran t-SNE with perplexities of 30 and 50, and for each perplexitiy, we used iters of 3000, 4000, and 5000. All the plots looked similar and shared a lot of the same features, so we simply chose a graph (perplexity: 50, iters: 4000) that we thought best demonstrated the differences.

Looking at the base embeddings diagram, we see the model generally mapped sentence embeddings to the same area, regardless of sentiment, which explains the large overlap of colours. However, on the modified embeddings diagram, we see a clear shape change where the lower sentiments 0 and 1, are roughly opposite sentiments 3 and 4. We also see what may be the beginnings of the sentiments being clustered, as groupings of each sentiment seem to start separating from each other. Though some of the groupings, sentiments 1 and 3, are more pronounced than others, sentiment 2, for example.

It's imperative to keep in mind that t-SNE is quite variable and thus, no definitive claims can be drawn from the graph. However, overall these diagrams give us insight into the kinds of changes that MNRL made to the embeddings, and show that our MNRL implementation most likely did make a difference in the model's performance.

5.2 Sentiment Analysis

To assess our model's sentiment analysis accuracy, we plot a normalized anti-diagonal confusion matrix (Figure 1b). This matrix records the proportion of correct and incorrect predictions by class, highlighting which are predicted with less accuracy. Ideally, cells off the anti-diagonal, which represent misclassifications, would be close to 0, reflecting high precision.

The model tends to mix up extreme sentiments with adjacent, milder categories, misclassifying many 'negatives', (0.62 of class 0), as 'somewhat negative' and 'positives', (0.43 of class 4), as 'somewhat positive'. This pattern likely arises from the embedding overlaps, in the embedding graph (1a), observed between these adjacent sentiment categories. It also, notably struggles with neutral sentiments, correctly identifying it only 34% of the time, while misclassifying it as somewhat negative (0.28) or somewhat positive (0.32), most times. This challenge likely stems from the nuanced nature of middle-class sentiments, which are inherently less distinct. For example, the sentence "It's a stunning lyrical work of considerable force and truth," which might intuitively be classified as strongly positive (sentiment 4), by both us and the model, is actually labeled as only being somewhat positive in the dataset.

5.3 BPPO

In their paper, Jiang et al. (2020) modify β from 0.99 to 0.999 after the first 10% of epochs. However, they failed to explain why did so. Ergo, when we implemented it, we chose to keep β constant. This led to us initially seeing the loss decrease in earlier epochs after which, we saw the loss steady increase. This, made us believe that Bregman had detrimental effects on our model, until we ran Bregman on a model that had already been finetuned on MNRL. In that run, we saw the loss hit its lowest after the first of ten epochs and from there it began increasing again. Yet, that run obtained the model's highest overall, non-ensembled, score across all tasks. These results suggest that the researchers modified their β, because they knew that Bregman generally converged during the first 10% of epochs, thus, the β change ensured that the model and its parameters would remain in the neighbourhood of that minima as the epochs increased.

6 Conclusion

Ultimately, we successfully implemented Anealed Sampling, Smart Loss, BPPO, and MNRL, all of which, led to improvements in our model's final predictions (even if minor). Our final model only required 6 hours to finetune, in comparison to the base model's 10 hours. In the future, we plan to make our MNRL embeddings changes more aggressive in hopes of further boosting our sentiment accuracy. For the work breakdown, Chiji did MNRL, the base Bert implementation, and Annealed sampling. Iddah did the base multitask implementation, SMART loss and model ensembling. Kwame and Iddah worked on BPPO and the milestone paper. Kwame worked on the embedding plotting and ethics statement. Chiji and Kwame worked on the poster. Lastly, we all worked on hyperparameter tuning, final paper, and project proposal.

References

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

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

Matthew Hayes. 2023. Serbertus: A smart three-headed bert ensemble. In *Stanford CS224N Default Project*.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. volume abs/1705.00652.

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*, page 2177–2190, Online. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 4487–4496, Online. Association for Computational Linguistics.

Johannes Daxenberger Nandan Thakur, Nils Reimers and Iryna Gurevych. 2020. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

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.

Asa Stickland, Cooper and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. page arXiv:1902.02671, Online. arXiv e-prints.