ConCATenation Curiosity: Evaluating Multitask Performance of minBERT

Stanford CS224N Default Project

Prithvi Krishnarao
Department of MS&E
Stanford University
prithvik@stanford.edu

Emily Redmond
Department of Computer Science
Stanford University
eredmond@stanford.edu

Abstract

Our goal is to implement key functions of the minBERT architecture and apply the model to perform well across NLP downstream tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity. We conducted experiments to analyze improvements on the baseline performance of the minBERT model and saw differing degrees of success in applying a combined loss function, cosine similarity, concatenation of input sequence pairs before feeding them into the model, SMART fine-tuning with regularized optimization, and projected attention layers (PALs). Our strongest model was merging the combined loss, concatenated inputs, and PALs architectures. We also discuss ethical considerations of our model, including the implications of data privacy and biased training data, and address challenges in applying Multitask fine-tuning and Multiple Negatives Ranking Loss Learning and gradient surgery.

1 Key Information to include

- TA mentor: Chaofei Fan
- External collaborators, External mentor, Sharing project: No for all
- Team contributions: Both Emily and Prithvi contributed to the core architecture for the baseline minBERT model. For the model extensions, they collaborated on SMART and concatenating inputs, Emily led the implementation of MNRL, and Prithvi led the implementation of Cosine similarity, Multitask fine-tuning, and PALs. They both contributed to the written report and to check-ins with their project mentor.

2 Introduction

We implemented a version of BERT, minBERT, using multi-head self-attention mechanism and transformer layers based on the original BERT architecture [1]. Our main focus was understanding and implementing the self-attention mechanism, layer normalization, and residual connections [2]. We utilized pre-trained BERT weights for embedding initialization and fine-tuned the model on three specific tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS).

BERT sits on the cutting edge of natural language exploration. Thus, implementing it and extensions of it represent meaningful steps in the frontier of natural language. Current issues with BERT are its massive computational intensity and lack of generalization to different tasks. As such, we seek to explore extensions that can address these issues.

Building on the base BERT multitask model, we experimented with a few targeted extensions and the targeted tasks after each in brackets:

- Combining Loss Functions [Task: All]

- Cosine similarity [Task: STS]
- Concatenation of input sentence pairs before feeding them into the model [Task: STS]
- SMART fine-tuning with regularized optimization [Task: Paraphrase detection]
- PALs: Projected Attention Layers [Task: All]
- Multiple Negatives Ranking Loss Learning [Task: Paraphrase detection] (attempted)

In our results, we find concatenating input sentence pairs before feeding them into the model, cosine similarity, and SMART fine-tuning to deliver the best improvement on model accuracy, in relative order, for the defined tasks and standard project datasets described in Data. We expected concatenation of inputs and cosine similarity to deliver accuracy improvements, as both methods logically increase similarity context and calculation. SMART under-delivered on our hypothesis, likely because its benefits are best observed in real-world data and user-input scenarios, which we did not target.

While we also expected Multitask fine-tuning and Multiple Negatives Ranking Loss Learning to also provide model improvements, we faced challenges in achieving success with these approaches due to complexity in integrating them within our base model architecture.

3 Related Work

Research on language model architectures and their applications has rapidly evolved, significantly influenced by the development of the Transformer model, which uses self-attention mechanisms for various NLP tasks [2]. BERT (Bidirectional Encoder Representations from Transformers) further revolutionized this field by introducing a pre-trained model that effectively captures deep bidirectional contexts [1]. This model forms the foundation of our minBERT implementation.

We use our own vanilla minBERT implementation as a baseline, but our extensions, outlined in the Appraoch section, were inspired by related work in the field.

We were inspired by a combined loss function stated in a 2022 paper [3] that illustrated how such a function allows a model to share learnings from one task to another. Further, we were intrigued by the description of using cosine similarity for word embeddings as a "natural" way to signify similarity in a 2019 paper [4]. Further, a 2020 paper [5] introduced SMART, Smoothness Inducing Adverserial regularization, as a method to encourage models to be less sensitive to small perturbations in the input data.

In 2017, a group from Google introduced the use of multiple negative examples during training to good and bad response[6]. This inspired us to attempt MNRL to reduced bias towards common responses observed in our baseline model. Finally, a paper discussing multitask fine-tuning [7] illustrated the potential in using low-dimensional multi-head attention mechanisms to increase multi-task training. A later paper [3] also mentioned gradient surgery works as an effective method to mitigate gradient conflicts that arise through such multitask fine-tuning.

4 Approach

We followed standard architecture denoted in the Default Final Project instructions for our BERT model and ascribe credit to the many authors and citations of this handout as well as the starer code provided. Below are details on each of our attempted extensions.

4.1 Combining Loss Functions

Inspired by a loss function mentioned in a paper out of Renmin University of China, [3] we saw how a combined loss function:

$$L_{total} = L_{task1} + L_{task2} + L_{task3}$$

forces a model to learn representations that are beneficial to all tasks. This combined loss function also serves as a regularization factor that prevents the model from overfitting on any specific task.

4.2 Cosine similarity

Building upon the Sentence-BERT (SBERT) methodology [4], our approach uses cosine similarity to measure semantic relatedness between sentence pairs in the STS-B dataset.

The formula for calculating cosine similarity is below, where u and v are the vector representations of the two sentences.

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

By fine-tuning the model on this task, we optimize the embedding space such that the cosine similarity between the vector representations of two sentences reflects their semantic similarity. Given the inherent semantic richness of BERT embeddings, we hypothesize that semantically similar sentences will naturally cluster closer together in this optimized embedding space. To align the cosine similarity values with the 0-5 scale of the STS-B dataset, we apply a scaling transformation to the output of our cosine similarity function.

4.3 Concatenating input sequences

In our baseline model, we implemented a naive embedding structure such that each sentence, even in the pair sentence dataset, is processed independently through the BERT model to get individual outputs, then concatenated the outputs afterwards. By concatenating the input sequences before feeding them into the model, the model can directly learn the relationship between the two sentences as part of its contextual understanding. This can potentially lead to a better understanding of how the sentences relate to each other, since the model sees them as a continuous text during training and inference. This is particularly relevant to the semantic textual similarity task, since the model can better learn the relationship between the sentences, strengthening its understanding of sentiments similarity. We ascribe credit for this idea from CS224N TA, Jingwen Wu.

We thus expect concatening the input sentences before feeding them into the model to have a notable improvement on the sentiment similarity task, as this should improve the model's holistic context and understanding.

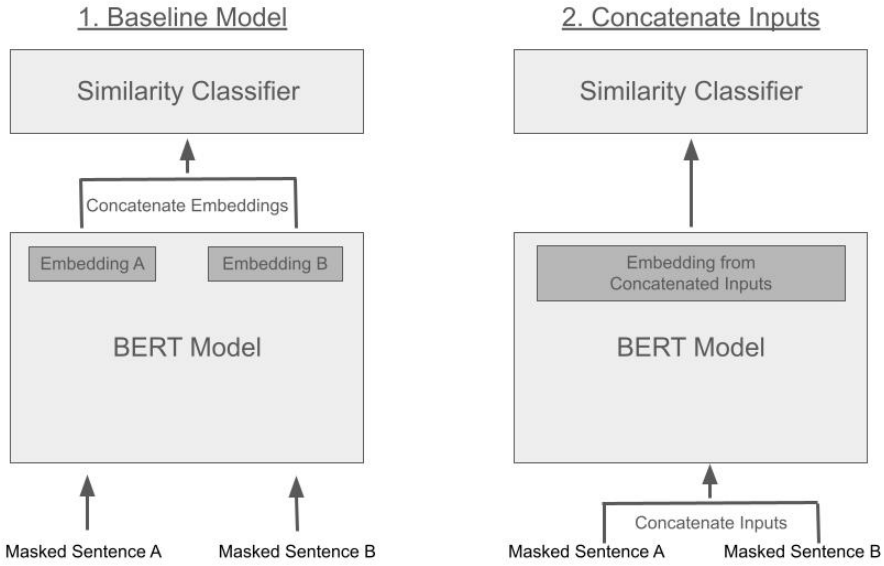


Figure 1: Baseline vs. Concatenate Inputs Model Architecture

4.4 SMART fine-tuning with regularized optimization

The SMART (Smoothness-Inducing Adversarial Regularization) method, as outlined by Jiang et al., is designed to enhance the generalization of neural networks by promoting smoothness in the model's

predictions [5]. SMART modifies the standard training objective by adding a smoothness-inducing regularizer to the loss function. We apply it within the context of fine-tuning our minBERT model for the paraphrase prediction task.

Objective Function:

$$\min_\theta \left(L(\theta) + \lambda_s R_s(\theta) \right)$$

Here $L(\theta)$ is the loss and $R_s(\theta)$ is the smoothness regularizer. λ_s is a hyperparameter for the strength of the regularization.

Empirical loss:

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

Smoothness-Inducing Regularizer:

$$R_s(\theta) = \frac{1}{n} \sum_i \max_{\|x_i' - x_i\|_p \leq \epsilon} l(f(x_i'; \theta), f(x_i; \theta))$$

Loss for Smoothness Regualrizer:

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

This regularizer encourages the model to produce similar outputs for slightly perturbed versions of the input to make the model output less sensitive to small variations in the input, which can help in mitigating issues like overfitting.

In the context of BERT, the baseline model fine-tunes for specific tasks by adjusting its parameters slighly based on the task's related dataset by minimizing a loss function that measures how far a model's predictions are from actual labels. Adding SMART changes this goal such that in addition to optimizing for the model to make correct predictions, these predictions should also not vary much if the input data is slightly altered by introducing perturbations to input embeddings and penalizing model sensitivity. This is relevant to real-world, noisy data that can vary in small ways and should not be treated differently by the model. We are using an external library to implement SMART[8].

We hypothesize this approach will increase the robustness of our model, thus improve the handling varied inputs, and the reliability of our model on 'real-world' data and operating conditions.

4.5 Multiple Negatives Ranking Loss Learning (MNRL)

MNRL is a method to fine-tune a model with "negative" sentence pairs in addition to the "positive" sentence pair training examples. To do so, it employs a loss function to minimize the distance between a positive sentence pair (a_i, b_i) while simultaneously maximizing the distance between a non-correlated or "negative" sentence pair (a_i, b_j) where $i \neq j$. We see this for a single batch:

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

where θ represents the word embeddings and S is a scoring function [6].

We expected this to significantly improve our accuracy results as it seems like it would hone our similarity representations, but in practice, we struggled to implement this method effectively in our model architecture, as described in Results.

4.6 PALs: Projected Attention Layer

PALs consist of low-dimensional multi-head attention mechanisms that are added in parallel to the existing layers of the model [7]. They work by projecting the high-dimensional hidden state of the model into lower-dimensional space using learnable projection matrices. This allows the multi-head attention matrix to operate in this lower-dimensional space, allowing for task-specific adaptation while preventing a vast increase in the number of parameters. Finally, PALs output is projected back to the original hidden state dimension and combined with the output of the corresponding BERT layer.

As mentioned in our related work section, we were also inspired by a paper out of Renmin University of China [3] that employed multi-task learning with gradient surgery to mitigate gradient conflicts that may arise among different tasks[9].

Gradient surgery works by modifying the gradient for each individual task so each gradient does not interfere destructively with the gradients of other task. This is done by projecting each task's gradient onto the normal plane of any other task's gradient that conflicts with it.

For task i, let g_i bet its gradient and for task j, let g_j be its gradient.

$$g_i^{PC} = g_i - \frac{g_i \cdot g_j}{||g_j||^2} g_j$$

$g_i^P C$ is the projected gradient for task i that no longer conflicts with the gradient of task j.

As we explain in the results section, although we were able to get a working implementation of PALs, we struggled to implement gradient surgery using an external library. [9]

5 Experiments

5.1 Data

For the sentiment analysis task, we used the Stanford Sentiment Treebank (SST) dataset [10] with 11,855 single sentences extracted from movie reviews and the CFIMDB dataset with 2,434 highly polar movie reviews for sentiment analysis. For paraphrase detection, we trained on the Quora dataset of 404,298 question/answer pairs [11] and to perform semantic textual similarity, we trained on the SemEval STS dataset of 8,628 sentence pairs of varying similarity [12].

5.2 Method & Experiments

Our primary metric for evaluating our model is reporting and comparing the dev accuracy for each of the 3 tasks. We tested each extension individually or in small combinations to isolate and identify the largest individual factors in model improvement.

We used the BERT-base configuration with 12 layers, 12 attention heads, and hidden size of 768. We used a learning rate of 1e-5 for full-model fine-tuning and 1e-3 for last-linear-layer fine-tuning. Our training time averaged 4 hours 43 minutes before we added in a combined loss function and 1 hour 12 minutes for the models with a combined loss function.

5.3 Results

In the table below, LLL refers to Last Linear Layer and FM refers to Full Model training. An asterics indicates an important note on the results, and bold values indicate our best model and the best performance across all models for each of the tasks.

Model	Dev Sentiment Accuracy	Dev Paraphrase Accuracy	Dev STS Correlation
LLL - Baseline	0.265	0.632	0.046
FM - Baseline	0.316	0.632	0.210
FM - Combined Loss (CL)	0.506	0.748	0.369
FM - CL & MNRL*	0.312	0.669	0.069
FM - CL & Cosine Similarity	**0.518**	0.701	0.669
FM - CL & Concat. Inputs (CI)	0.498	**0.749**	**0.876**
FM - CL, CI, & SMART	0.513	0.739	0.869
FM - CL, CI, & PALs	0.510	0.746	0.873

Table 1: Experiment Results

	Test Sentiment Accuracy	Test Paraphrase Accuracy	Test STS Correlation	Overall Score
Best Model:	0.531	0.745	0.874	0.738

Table 2: Best Model Results

Our best model is Combined Loss + Concatenated Inputs + PALs.

5.4 Analysis

As expected, our Last Linear Baseline model performed quite poorly. Given the lack of fine-tuning for this model, this is expected. Our full model baseline had significantly improved STS correlation, indicating that fine-tuning likely helped capture semantic similarity.

Our Combined Loss model experienced substantial gains across the board. This is likely because the model learned representations that are useful across different tasks, which in turn acted to regularize and prevent overfitting on any one specific task. Further, our implementation of combined loss ensured uniform weighting across each of the tasks, promoting balanced learning.

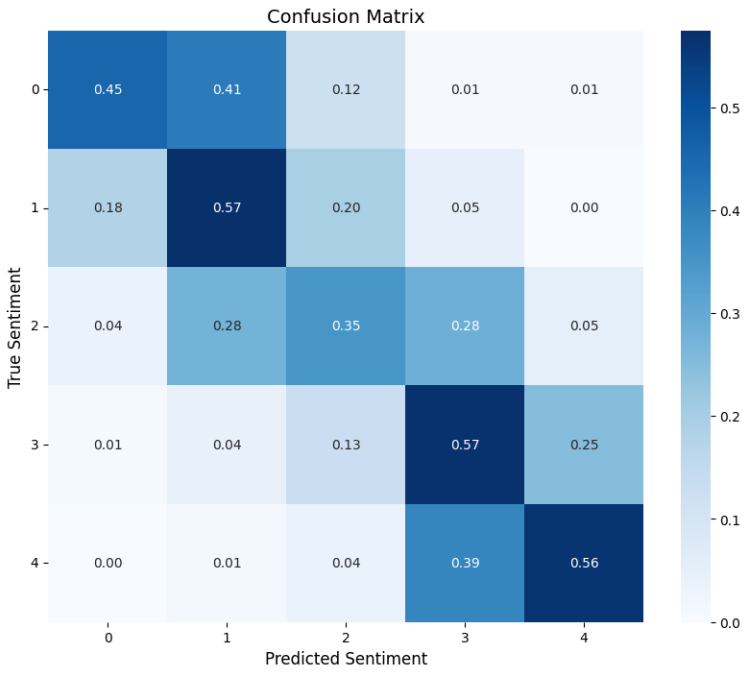
Our Cosine Similarity model with combined loss highly outperformed our original combined loss model in regards to the STS Correlation. This is because Cosine Similarity directly captures the similarity between different sentence embeddings more naturally than a linear layer followed by MSE loss.

We note significant improvements in the STS task compared to the baseline model using the concatenated input architecture, as expected in the Approach section. This is likely due to the model benefiting from enhanced context directly within the BERT embeddings, rather than only for the similarity calculation as in the baseline model. As such, having direct interaction between the embeddings also led to our concatenated architecture performing better than even our Cosine Similarity architecture because the model can directly learn interactions between the individual word embeddings instead of having to rely on a similarity metric.

We do not observe notable improvements using SMART, perhaps because we did not set up the problem to fully realize or take advantage of SMART's stabilization offerings for "real-world" data. Given our carefully engineered and standard datasets, the SMART affordances covered in the Approach section could not shine through, though may have given more real-world variable user input. Further, given the high run-time we experienced with SMART, we were unable to sufficiently hyper-parameter tune. Given more time, we would have liked to see how the model would have performed with different hyperparameters.

We also did not observe notable improvements using PALs. Although sentiment accuracy slightly increased as compared to the model with combined loss and concatenated inputs, paraphrase accuracy and STS correlation dropped. We prescribed the increase in sentiment accuracy as potentially being due to better contextual representations within BERT layers. However, the decrease we see in the other tasks could have been due to conflciting gradients between layers. Although we sought to implement gradient surgery [9], we were not able to succesfully implement this in our code without running into RAM issues with our cloud GPU.

While our best performing model (CL, CI, PALs) performed quite well, we saw that our sentiment scores were lower than the leading models on the leaderboard. To finish our analysis, we will briefly analyze our sentiment performance.



Analyzing a confusion matrix of sentiment scores from our model, we see that the model struggles with differentiating between negative and somewhat negative sentiments. Similarly, we also see the model confuses somewhat positive and positive sentiments. This is because our dataset is imbalanced, with having higher prevalence of somewhat negative and positive sentiments than any other sentiments by a significant margin. As such, this discrepancy could be addressed in future work by undersampling from majority classes and oversampling from minority classes.

* NOTE: We tried to apply MNRL to the STS task, but our implementation resulted in our STS correlation score dropping to 0.07. We hypothesize that this significant drop is due to the MNRL assumption that all sentence pairs (a_i, b_i) in the MNRL equation are highly positive, while in reality our dataset included a continuous scale from 0 to 5 for sentence pairs and each example pair was not necessarily highly correlated.

We attempted many fixes to circumvent this disparity, such as creating a new data class that only includes sentence pairs with a strong sentence correlation (ie. ≥ 0.4) and pretraining using MNRL on this data, trying the same filtering approach in the main training loop, and applying MNRL to the paraphrase detection task. Unfortunately, none of these fixes improved the abnormally low results, thus indicating a likely bug in the data or label processing that we could not uncover despite many sessions with TAs. We are including this result and analysis for transparency and learning opportunity.

6 Conclusion

"Curiosity killed the cat, but satisfaction brought it back" is a well-known English idiom. In the case of our BERT model, "cat" or con-CAT-enation served our model well and definitely brought us satisfaction, providing the largest jump in single-task accuracy. We found that the increase concatenation provided for STS correlation was higher than that provided by implementing cosine similarity.

As expected, implementing a combined loss function lifted our performance across all tasks. While we had successful implementations of SMART and PALs, these extensions did not improve our performance as much as we thought they would, although they showed some interesting findings. Our best model was an ensemble model combining input concatenation, a combined loss function, and PALs, leading to an overall test score of 0.738.

6.1 Ethical Considerations

There are several ethical challenges and societal risks to consider in developing a model to perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks.

Broadly, the use of large language models in processing sensitive or personal data can lead to privacy concerns, such as inadvertent exposure of personal information through model outputs. More specifically, some of our input examples may include personal data such as names that could also influence similarity scores. This may be okay in paraphrase detection, like if two similar sentences cite the same person, or may cause inadvertent problems in sentiment similarity if the underlying sentiment is supposed to be the same but the model is confused or biased by different cultural names in the sentences.

The potential for bias in model predictions is also significant, especially in sentiment analysis. This bias can arise from skewed training data that does not adequately represent diverse viewpoints or demographic groups, or worse, inherently puts down certain groups, leading to unfair treatment or discrimination in application. This could lead certain cultural groups to be labelled as similar or correlated to particular actions.

To mitigate the risk of privacy violations, one approach is to implement strict data handling policies that include de-identification of personal data before it is processed by the model. This could help minimize the risk of exposing individual data or comparing individuals. Addressing the risk of bias requires a nuanced approach of ensuring the training data covers a wide range of demographics and viewpoints. Bias can also be addressed by tuning the model using fairness-aware metrics to detect and correct bias in predictions; for example, Equal Opportunity scoring, which is a fairness criterion used to control the equality of true positive rates among different groups when categorizing people into classes (or more specific to our tasks, categorizing sentences and the groups that are the subject or object of these sentences as similar or their sentiment as negative or positive) [13].

6.2 Limitations & Future Work

We chose not to conduct a hyper-parameter search to save time and resources in exploring the effectiveness of non-trivial model extensions. We recognize the limitation of not conducting a hyper-parameter search in achieving the best possible model accuracy but stand by the resource trade-off for the purpose of this report. In the future, we would like to conduct such a search.

In future work, we would also like to design an effective method for negative example sampling for both the sentiment similarity and paraphrase detection tasks to use in Multiple Negatives Ranking Loss Learning (MNRL),

as touched on in the Approach section. Extending beyond the cited work on (MNRL), we hypothesize there is potential for positive results in the sentiment similarity in including more training data specialized in idioms and common expressions; for example, including more "positive examples" with high-similarity sentence pairs like "Good luck tonight!" and "Break a leg tonight!" and intentional "negative examples" with low similarity sentence pairs like "I hope she breaks her leg" and "I hope she breaks a leg tonight at the performance!"

Inspired by our strong results for the STS task in concatenating the input sentences before passing them into the BERT model, we urge future work to consider creative embeddings structures that provide the model with as much context as possible for the task at hand, while balancing overfitting. Specifically, paraphrase detection is an obvious next application to experiment with concatenating input embeddings, given the shared goal of similarity detection as in STS. Furthermore, given the imbalance we found in our sentiment dataset, we also believe there is significant value in oversampling and undersampling within each dataset to create a more balanced model for each task.

References

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

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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

[4] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

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

[6] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

[7] Asa Cooper Stickland and Iain Murray. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR, 09–15 Jun 2019.

[8] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *ACL*, 2020.

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

[10] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 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, 2013.

[11] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*, pages 45–52, 2008.

[12] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM)*, pages 32–43, 2013.

[13] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.