Implementing minBERT and Extensions for Multi-Task Learning

Stanford CS224N Default Project

Jiani Wang
Department of Computer Science
Stanford University
`jianiw@stanford.edu`

Qinchen Wang
Department of Computer Science
Stanford University
`qinchenw@stanford.edu`

Yan Wang
Department of Computer Science
Stanford University
`yan12@stanford.edu`

Abstract

This project studies how well a pretrained Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) model generalizes to different downstream tasks, and explores improvements to enable better generalization. BERT is known as a pretrained model that when finetuned on specific tasks with just one additional output layer, achieves outstanding results. Its power to generalize to multiple tasks all at once is not studied in the original paper and is the primary focus of our project. We introduce custom output layers and loss functions for each task, and then merge the gradient updates into a single model. Our results from the ablation study show how well these techniques work on multi-task learning.

1 Key Information to include

- Mentor: Anuj Nagpal
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Multi-task learning has emerged as a promising approach to address the challenges of deep learning models in natural language processing (NLP) tasks. The technique enables a single model to learn multiple related tasks simultaneously, leveraging shared representations and improving the generalization capability of the model. However, the optimization process of multi-task learning can be complex, and interference between different tasks can pose significant challenges to achieving efficient multi-task learning. In this project, we implemented minBERT, a lightweight version of the popular BERT model, and explored its extensions for multi-task learning on downstream tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity. We developed a simple method to conduct multi-task learning based on minBERT, which aimed to preserve the best performance of every single task while leveraging shared representations using multi-task techniques. Through experiments and ablation studies, we demonstrated the effectiveness of our method. Our approach almost preserved the best performance of every single task while improving the generalization capability of the model on all three tasks. This demonstrates the potential of minBERT and its extensions for efficient and effective multi-task learning in NLP tasks.

3 Related Work

BERT BERT demonstrates the importance of bidirectional pretraining for language representations(Devlin et al., 2018). BERT introduces Masked LM (MLM) to prevent each word from indirectly "seeing itself" in bidirectional models. BERT is the first work that uses masked language models to enable pretrained deep bidirectional representations. Such pretrained representations could be generalized and used in different kinds of downstream tasks. It is the first finetuning-based representation model that outperforms other models in both sentence-level and token-level tasks. For example, BERT pretrained on Next Sentence Prediction (NSP) tasks to improve sentence-level downstream tasks. Later, a few other models, such as ALBERT(Lan et al., 2019), RoBERTa(Liu et al., 2019), and ELECTRA(Clark et al., 2020) improve based on the original BERT model.

Undoubtedly, BERT is one the most important works in the NLP area that uses pretraining to improve performance. However, BERT mainly focuses on pretraining, and we still need to finetune the model for specific downstream tasks. BERT, as a pretrained model, is targeting general applications. In our default project, we need to involve different extensions of BERT to get better performance on tasks like sentiment analysis, paraphrase detection, and semantic textual similarity.

Sentiment analysis Sentiment analysis is an NLP task to classify the polarity of a given text (i.e., whether the tone of the text is positive, negative, or neutral). Sentiment analysis can be utilized to determine individual feelings towards particular products, politicians, or within news reports. Medhat et al. (2014) summarized different methods to approach this task, including both machine learning-based methods and lexicon-based methods. The emergence of language models such as BERT also helps the performance of sentiment analysis in different scenarios. For example, Sun et al. (2019) utilized BERT for aspect-based sentiment analysis, which is a subtask of sentiment analysis aiming to identify fine-grained opinion polarity towards a specific aspect. Sousa et al. (2019) applied BERT to stock market sentiment analysis. Seeing so many successful uses of BERT for sentiment analysis, we believe BERT is the proper choice for our sentiment classification task: classifying the sentiment of movie reviews.

Paraphrase detection Paraphrase detection is the task of finding paraphrases of texts in a large corpus of passages. Paraphrases are "rewordings of something written or spoken by someone else", so paraphrase detection is basically determining whether two texts convey the same meaning (Fernando and Stevenson, 2008), which is mostly a binary classification task. BERT embeddings are also widely used in the paraphrase detection task. For example, Wahle et al. (2021) compared the performance of paraphrase detection using BERT, RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020). Although BERT is not the best among the three, it still has very high accuracy in identifying paraphrases.

Semantic textual similarity The semantic textual similarity (STS) task measures the extent to which some texts are similar to others. This task is different from paraphrase detection in that it's no longer limited to a binary classification task, but outputs a quantitative value that represents how similar the two texts are. For example, we can use a number between 0 and 5 to represent the similarity. The number can be an integer or a continuous value. The problem is how should we measure this similarity. Reimers and Gurevych (2019) proposed to finetune the BERT model with cosine embedding loss, because cosine similarity is a one of the best ways to measure the similarity between two embeddings. Henderson et al. (2017) introduced the idea of using multiple negatives ranking (MNR) loss to finetune the embeddings.

Multi-task learning Multi-task learning has become a promising approach for sharing structure across multiple tasks to enable more efficient learning. However, multi-task learning is also a difficult optimization problem. In order to efficiently merge all the tasks together, multi-task learning need to combine different data distribution and loss function in a meaningful way. Intuitively, several multi-task RL algorithms have been using independent training as a subroutine of the algorithm before training on multiple tasks (eg, Parisotto et al. (2016)). Another kind of solution is to gain knowledge of the multi-task optimization landscape and resolve gradient conflicts in the landscape. Yu et al. (2020) proposed gradient surgery, which could avoid such interference between task gradients, and merge the gradient information in an efficient manner.

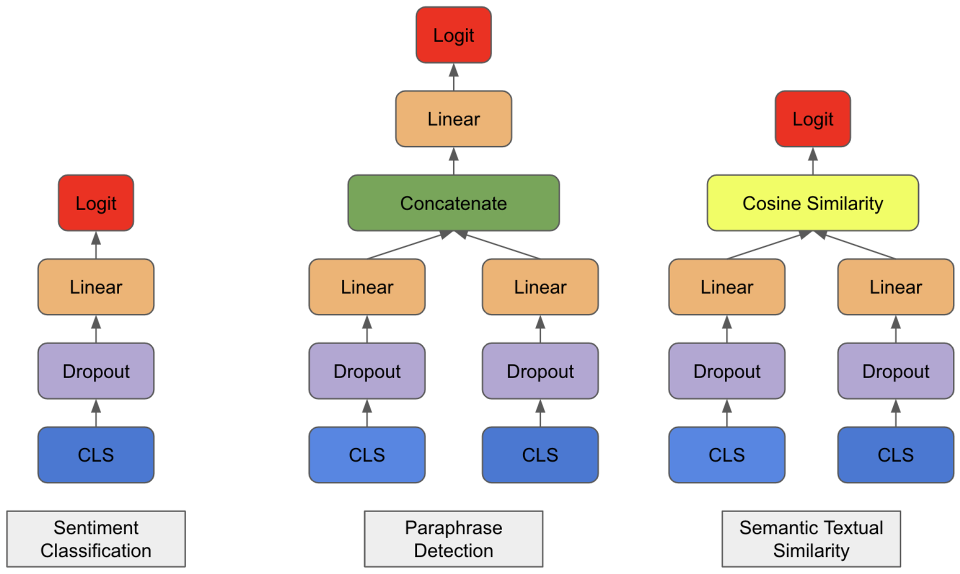

Figure 1: Model architecture

4 Approach

4.1 BERT

We implemented a baseline BERT model built up from the starter code [1]. The main contribution of BERT is incorporating bidirectional context in a pretrained language representation embedding. The attention mechanism, BERT layer, and BERT model are key components that enabled this bidirectional context visibility. Our implementation of these is a realization of the model structure and equations described by the original BERT paper (Devlin et al., 2018). We implemented the "efficient version" of Adam optimizer following instructions from the starter code (Kingma and Ba, 2014).

4.2 Downstream tasks architecture

In order to perform the three tasks at the same time, we implement different prediction heads for each of them. For sentiment classification, we use the default architecture as described in the project handout. First, we encode the sentences using BERT and obtain the pooled representation of each sentence (CLS token) The class will then classify the sentence by applying dropout on the CLS token and then projecting it to 5 logits using a linear layer for the classification task.

For the paraphrase detection task, we start with applying dropout on the CLS token and project it to a lower dimension representation for each sentence. Then we concatenate the two embeddings and pass them through another linear layer to get the output.

For the semantic textual similarity task, we build something similar. It starts with applying dropout on the CLS token and projecting it to a lower dimension for each sentence. Instead of concatenation, we compute the cosine similarity between the two embeddings and scaled the output to the range of 0 to 5 to match the dataset labels. Figure 1 presents the architecture for the three prediction heads.

4.3 Loss function

Binary Cross-Entropy Loss Binary Cross-Entropy (BCE) loss is a good loss function for binary classification tasks, such as paraphrase detection for our project. Equation 1 shows the definition of the BCE loss function. The label y_i is either 0 or 1 and $\hat{y}_i$ is the predicted probability of being

[1]https://github.com/gpoesia/minbert-default-final-project

classified as 1, so $\hat{y}_i$ is a number between 0 and 1.

$$Loss_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{1}$$

Cosine Embedding Loss The semantic textual similarity task usually represents the similarity between two sentences with the cosine similarity between two embeddings, so we want to take cosine similarity as a part of our loss computation (Reimers and Gurevych, 2019). We come up with two ways to implement this cosine embedding loss, one for a regression task and one for discrete labels. Because the similarity score of the dataset we used for this semantic textual similarity task is continuous values, we think it would make more sense to perform a regression task. Therefore, in the prediction head, we calculate the cosine similarity of two embeddings. However, the cosine similarity is between -1 and 1 while the range of the similarity score is between 0 and 5, we scale the cosine similarity score to the range of 0 to 5 in order to match the labels. Finally, we apply the mean squared error (MSE) loss, shown in Equation 2, where y_i is the label and $\hat{y}_i$ is the scaled cosine similarity. In this setup, sentences that are the equivalent have a cosine similarity of 1 while those that are unrelated have a cosine similarity of 0.

$$Loss_{MSE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

Multiple Negatives Ranking Loss In addition to cosine embedding loss, another loss function that optimizes for embedding space similarity between similar sentences is the multiple negatives ranking loss. In our baseline, we used the cosine similarity score as the output to the Textual Similarity task, and to determine whether two sentences are close enough to be classified as paraphrases of each other in the paraphrase detection task. Since the quality of vector similarity measure plays such an important factor in both tasks, we will experiment with adding another similarity score measure directly to the loss at the fine tuning stage. The loss function we want to consider is Multiple Negatives Ranking Loss, which brings sentences that are similar close to each other in the embedding space, while pushing those that are dis-similar far apart(Henderson et al., 2017). The MNR loss is calculated as in Equation 3 where K is the number of sentence pairs that are labeled as similar and S is the cosine similarity function. The loss function minimizes the distance between a_i and b_i while it simultaneously maximizing the distance between a_i and b_j where $i \neq j$.

$$Loss_{MNR} = \frac{1}{K}\sum_{i=1}^{K}\left[S(x_i - y_i) - \log\sum_{j=1}^{K}e^{S(x_i, y_j)}\right] \tag{3}$$

4.4 Training approach

Baseline The baseline model is only trained on the sentiment classification task but evaluated on the three tasks. The loss function we used for the baseline model is the cross-entropy loss.

Round robin training While our three tasks have different structures for the final prediction layer, they share the same BERT model body, and the finetuning process updates a large body of shared weights. Our approach of enabling the model to predict for all three tasks is finetuning the model on each of the three tasks in a round robin manner. In each epoch, the model first trains on all batches of the SST dataset, then on a subsample of the Quora paraphrase dataset, and finally on the STS dataset. We choose a subset of the Quora paraphrase dataset with a sample size of 6000 for most of our training - due to limited computation power, we decide to sample a size similar to the SST and STS dataset.

Finetune + Pretrain In our implementation and training, we combine the pretraining and finetuning process. In order to reflect the different importance of different tasks, our training process is divided into 2 passes: The first pass is to “pretrain" the Bert model on paraphrase tasks and the whole Quora dataset, with BCE loss function specified. We choose paraphrase tasks because the size of Quora dataset is relatively larger than the other dataset, which could reflect more language features to the model. In the second pass, we train on all 3 tasks using the round robin strategy, with MNR loss function for semantic textual similarity and paraphrase detection, and cross-entropy loss for sentiment classification.

Gradient surgery Yu et al. (2020) proposes gradient surgery. In our implementation, we use the code from Tseng (2020), which follows the original implementation from the paper. In the implementation of gradient surgery, it projects the gradient of the i-th task g_i onto the normal plane of another conflicting task's gradient g_j:

$$g_i = g_i - \frac{g_i * g_j}{||g_i||^2} * g_j \tag{4}$$

5 Experiments

5.1 Data

We use different datasets for different tasks of our project. For sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013). The SST dataset consists of 11,855 single sentences extracted from movie reviews. The dataset includes 215,154 phrases, each with a label of a number from 1 to 5. 1 means negative while 5 means positive. For paraphrase detection, we use the Quora dataset (Shankar Iyer and Csernai, 2017), which consists of 400,000 question pairs with labels indicating whether the two questions are paraphrases of each other. For semantic textual similarity, we use the SemEval STS Benchmark dataset (Agirre et al., 2013), which consists of 8,628 different sentence pairs with different similarities on a scale of 0 to 5 while 0 means unrelated and 5 means having equivalent meaning.

5.2 Evaluation method

We use different evaluation metrics for different tasks. Because sentiment analysis and paraphrase detection are classification tasks, we use accuracy as our main evaluation metric, as it can measure the ratio of correct predictions by our model. For semantic textual similarity, we use Pearson correlation as our main evaluation metric, as it can measure the correlation between the true similarity score and the predicted similarity score. A higher correlation indicates better performance on the STS task.

During multi-task training, we need an evaluation method to measure how the model performs across the 3 tasks at the same time. We decided to simply use the average of the evaluation scores of the 3 tasks (the accuracy of sentiment classification, the accuracy of paraphrase detection, and the Pearson correlation of Semantic Textual Similarity), as this is also the main metric on the project leaderboard.

5.3 Experimental details

We trained our models on AWS with Tesla T4 GPU. The batch size is 32 and the learning rate is 1e-5 across all experiments. The baseline model is trained only on the SST dataset using the cross-entropy (CE) loss and we used the same loss for the sentiment classification across all the multitask training experiments as well. The dropout rate for the baseline model is 0.5.

Then we trained a total of 5 experiments to better understand the performance of multitask training with different configurations. For these 5 experiments, we trained each model for 10 epochs and the dropout rate is 0.3. Because the Quora dataset is very large, we randomly sampled 6000 data points for each experiment. To study the effect of the loss function on paraphrase detection and semantic textual similarity, we conducted a total of four training combinations. We used binary cross-entropy (BCE) loss or multiple negatives ranking (MNR) loss for the paraphrase detection task and cosine similarity with mean squared error (MSE) loss or MNR loss for the semantic textual similarity task. The results for each combination of these loss functions are shown in rows 2 to 5 in Table 1. In addition, to study the effect of the dropout rate on the model performance, we ran another experiment, shown in row 6 in Table 1, where we set the dropout rate to 0.3. We use the combination of loss functions to represent the experiment.

Finally, after seeing the results from the above 6 experiments, we decided to train a new model, shown in row 7 in Table 1that is first pretrained on the paraphrase detection task on the full Quora dataset for 10 epochs and then finetuned with multitask learning for another 10 epochs.

The training time for Experiment 1 is around 20 minutes, the training time for Experiments 2 to 6 is around 2 hours, and the training time for the last experiment is around 12 hours.

Experiments	Sentiment Score		Paraphrase Score		STS Score		Overall
	Train	Dev	Train	Dev	Train	Dev	
Baseline	0.644	0.511	0.473	0.469	0.222	0.250	0.410
CE+BCE+COS	0.925	0.490	0.757	0.729	0.953	0.454	0.558
CE+BCE+MNR	0.979	0.511	0.777	0.743	0.529	0.503	0.586
CE+MNR+COS	0.806	0.519	0.388	0.392	0.924	0.491	0.467
CE+MNR+MNR	0.953	0.500	0.371	0.382	0.622	0.614	0.499
CE+BCE+MNR Dropout 0.3	0.975	0.507	0.749	0.736	0.441	0.459	0.567
Pretrain, CE+MNR+MNR	0.996	0.505	0.813	0.761	0.608	0.571	0.612

Table 1: Experiment results

5.4 Results

The results of our experiments are presented in Table 1. The baseline model works poorly on the paraphrase detection and semantic textual similarity task. This is as expected because it's not even trained for those two tasks. Comparing the results from Experiment 2 to 5, we see that cosine embedding loss leads to overfitting on the STS dataset and BCE loss performs well for the paraphrase detection task. The MNR loss is very robust against overfitting and is better than cosine embedding loss on the STS task, but performs worse than the BCE loss for the paraphrase detection task. This results align with our expectation. Then, in Experiment 3 and 6, we basically compared the effect of dropout rate on the performance of different models. The results demonstrate that a larger dropout rate can effectively reduce the overfitting, which also aligns with our expectation.

The best model is achieved from Experiment 7, where we first pretrained our model on the full Quora dataset and then finetuned the model through multitask training on the 3 tasks. We did this because we only used 6000 samples from the Quora dataset, but didn't want to train on the whole Quora dataset during the multitasking training. During the finetuning stage, we again used 6000 samples as in previous experiments and used MNR loss for both paraphrase detection and semantic textual similarity since MNR loss is robust against overfitting. The results, as expected, are the best among all 7 experiments.

6 Analysis

Overfitting Rows 2 to 5 in Table 1 show that cosine similarity with mean squared error loss leads to significant overfitting for the semantic textual similarity task. On the training set, the CE+BCE+COS loss function combination achieves 0.953 in correlation score, but only a score of 0.454 on the dev set. On the other hand, using MNR for the semantic textual similarity task resolves the problem while improving the dev set accuracy. With the CE+BCE+MNR loss function combination, we achieve a score of 0.529 on the training set of the semantic textual similarity task, and a score of 0.503 on the dev set. Similar analysis can be made when comparing training and dev set results for the loss function combinations CE+MNR+COS vs. CE+MNR+MNR.

This is likely because using cosine similarity in the loss function is only good at bringing similar sentence embeddings close together. In many cases this is not enough - imagine if all sentence embeddings sit close to each other in the embedding space. MNR loss, on top of bringing similar sentences close together, is able to break dis-similar sentences far apart. We do not observe such overfitting effect using binary cross-entropy loss on the paraphrase detection task, which strengthens our speculation that the issue comes from using cosine similarity to train for the sentence embeddings.

When overfitting appears on some task, it means our model fails to generalize on unseen data, which implies that we finetuned too aggressively or the model is too complex. One way to handle overfitting is to use dropout layers in the model. We started by setting the dropout rate to 0.3, but the model is heavily overfitting on the STS task, so we try to increase the dropout rate to 0.5, which reduces the overfitting a lot. Therefore, the dropout layer does help with reduce or prevent overfitting, but that's not the only thing we can try in the future. To deal with overfitting, we can also consider some regularization methods. For example, we can use smoothness-inducing regularization to manage the complexity of the model or Bregman proximal point optimization to prevent aggressive updating. Both methods are proposed by Jiang et al. (2019).

Importance of multi-task learning During the training process, we observed that improving the performance of the paraphrase detection task often leads to a decrease in the performance of the Semantic Textual Similarity (STS) tasks. This result can be attributed to the fact that the datasets for these tasks have different distributions, and optimizing for one task might cause the loss function to converge to a suboptimal solution for the other task. Thus, it highlights the challenges of efficiently performing multi-task learning. If we do not carefully select and combine the tasks, the performance might even deteriorate instead of improve. Nonetheless, despite these challenges, multi-task learning remains an important approach in machine learning as it allows for leveraging shared representations and improving the generalization capability of the models.

7 Conclusion

Our most important finding is that a single BERT model performs well on all three tasks when finetuned on these tasks one after the other. There are challenges unique to the problem of multi-task training. For example, we observed significant overfitting on one task while having little overfitting on the other. This challenge was mitigated by choosing a more suitable loss function for the specific task and a higher dropout rate.

We often observe a performance drop for one task when the model is being trained on the dataset of another task. We think that simple round robin training and our model architecture were not the best way to merge gradient updates. More work can be done on studying/resolving this performance conflict in the future.

References

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

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

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

Samuel Fernando and Mark Stevenson. 2008. 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.

Matthew 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. *arXiv preprint arXiv:1705.00652*.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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*.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning.

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

Nikhil Dandekar Shankar Iyer and Kornél Csernai. 2017. First quora dataset release: Question pairs.

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.

Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. 2019. Bert for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601. IEEE.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Jan Philip Wahle, Terry Ruas, Norman Meuschke, and Bela Gipp. 2021. Are neural language models good plagiarists? a benchmark for neural paraphrase detection. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 226–229. IEEE.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.