minBERT and Multi-task Training with Gradient Surgery

Stanford CS224N Default Project

Yunqi Li
yunqili1@stanford.edu
Department of Computer Science
Stanford University

Yihe Tang
yihetang@stanford.edu
Department of Computer Science
Stanford University

Abstract

In this project, we implement some key components of the BERT model, and extend it to perform three different tasks: sentiment analysis, paraphrase detection, and semantic textual similarity simultaneously. Our aim is to optimize the overall performance of the model on the three tasks. The study examines several strategies including gradient surgery, using extra datasets, data sampling and changing dropout rates. We also explore different methods to train the model on three tasks at the same time. Our results indicate that mixing the datasets enhances performance compared to training each task sequentially, and applying the gradient surgery technique on the former helps to stabilize the training process, although it does not provide a noticeable boost in accuracy. Additionally, our experiments suggest that directly augmenting the fine-tuning process with auxiliary tasks does not yield any improvement on the model's performance. Our model reaches 0.513 on sentiment analysis task, 0.877 on paraphrase detection task, and 0.870 on semantic textual similarity task, with an overall score of 0.753 on dev set leaderboard. It also achieves 0.531 on sentiment analysis task, 0.876 on paraphrase detection task, and 0.869 on semantic textual similarity task, with an overall score of 0.759 on test set leaderboard.

1 Key Information

- External collaborators, External mentor, Sharing project: None
- Yihe Tang uses 3 late days of Yunqi Li

2 Introduction

The Bidirectional Encoder Representations from Transformers(BERT) [1] is the state-of-the-art model for generating embedding representations for natural language, which is semantically rich and can be utilized for a variety of downstream tasks. In the first part of our project, we finish the implementation of the original BERT model, train and evaluate it on the Stanford Sentiment Treebank dataset [2].

The next step of our investigation is multitask learning on BERT. While we observe impressive results of BERT finetuned on the SST task, we recognize the fact that task-specific finetuning would lead to task-specific model weights for BERT [3], which could bring inefficiency or inconvenience under circumstances where the same embeddings being used for multiple tasks is preferred. Therefore, we are interested in exploring methods to enhance the robustness of BERT embeddings, enabling them to perform well across diverse language tasks.

To accomplish this, we first examine the effect of training order on the prediction accuracy. We compare sequential training, where we train the model on one task in a round-robin manner, with mixed training, where the model is jointly trained on data from all datasets combined. In addition, since the tasks have different final goals, it is reasonable to infer that the gradient of each task's loss function might be different in direction, which could potentially lead to inefficiency in optimization step. Thus, we adapt the gradient surgery technique which is originally proposed for reconciling this conflict in multi-task reinforcement learning [4], and recently been adapted for natural language processing tasks [5]. However, the original NLP adaptation is trained with manually defined auxiliary task on the same dataset, so we are interested in checking the effectiveness of this technique when applied to multiple tasks on distinct datasets and prediction goals. Finally, we attempt to enhance the fine-tuning procedure by incorporating an additional task with its unique dataset to experiment if the model can generate more robust representations when encountering more diverse data.

3 Related Work

The BERT [1] model is pretrained on large unlabelled text datasets with a "masked-language-model(MLM)" objective, which trains the model to predict randomly masked tokens based on their context. The key use of self-attention mechanism and bi-directional training design allows BERT to selectively focus on sentence components and generate semantic-rich representations. BERT demonstrated its potential by reaching state-of-the-art performance on multiple natural language processing tasks, and has been viewed as a powerful structure widely used in NLP research for generating text embeddings.

The MT-DNN [6] paper proposed a model for natural language multi-task learning which introduced a framework combined with shared BERT layers and task-specific prediction heads. The choice shared BERT allows the model to leverage cross-task data and produce more generalize representations. We adapted the framework design as well as the training losses used for relevant tasks in our model. In MT-DNN, the authors mentioned that the presence of multiple losses acts as a regularization effect for model training, while another work MTRec[5], which attempted to perform multi-task learning for new recommendation, showed the destructive effect for model optimization and performance brought by conflicting gradients between tasks.

Instead, MTRec[5] utilized the Gradient Surgery [4] technique to alleviate the gradient interference issue. This simple and effective approach is originally proposed for multi-task reinforcement learning, which projects conflicting gradient from tasks to a normal plane to align their direction and thus improving update efficiency. As reported in MTRec, this technique indeed brought boost in model performance. However, considering the fact that the tasks in MTRec are all on the same dataset, which differs from our multi-dataset setting, we are excited to examine the effect of gradient surgery for our models.

4 Approach

4.1 Model

We implemented the BERT model introduced in [1] for text representation generation, with our training procedure and detailed model structure shown below in fig.1. In our experiments, we will load weights for the BERT model MLM-pretrained on large language corpus and perform further finetuning. In addition, we created an efficient version of AdamW algorithm proposed in [7] for model optimization. In following subsections we discuss several important technical aspects of our model and optimizer.

4.1.1 Multi-head self-attention

BERT utilizes the multi-head self-attention algorithm to capture contextual semantic information for input text. An original "Scaled Dot-Product Attention" as shown in fig.2(left) computes the attention function on a set of queries, keys and values packed together into matrices Q, K and V. Specifically, the outputs is computed as:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head attention 2(right) is an extension that allows the model to jointly attend to information from different representation subspaces at different positions, computed as:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

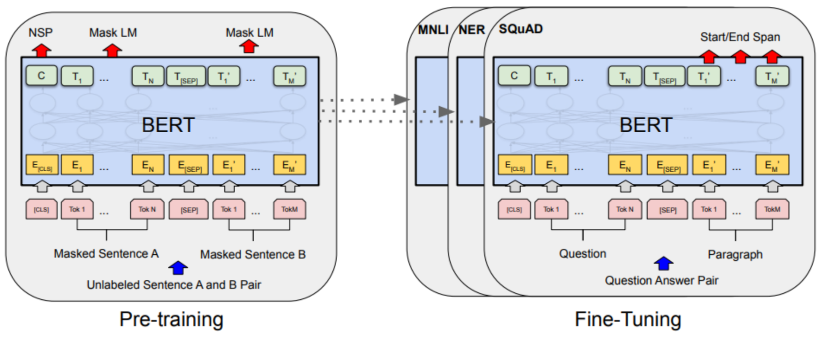

Figure 1: Overall pre-training and fine-tuning procedures for BERT.

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

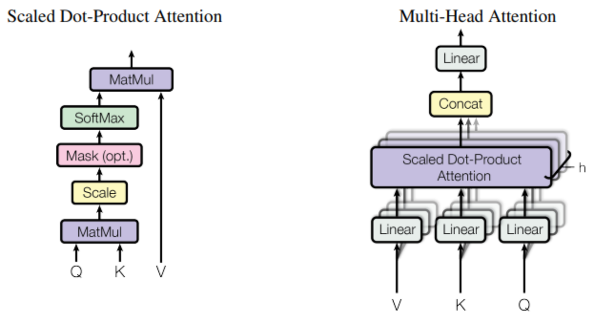

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

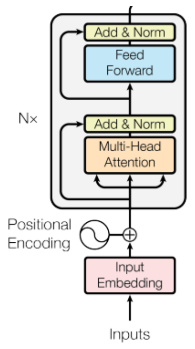

Figure 3: Encoding part of the Transformer.

4.1.2 Transformer layer

The transformer layer used in our BERT model is similar to the encoding part in the original paper, as in fig3, while we discard the decoding part. The Transformer layer, as shown in 3, consists of multi-head attention, followed by an additive and normalization layer with a residual connection, a feed-forward layer, and a final additive and normalization layer with a residual connection.

4.2 Downstream Tasks

On top of the shared BERT model that generates representations for input text, we followed the design in [6] and designed a prediction head for each of our downstream tasks.

For classification problems like Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset, we use a linear layer that projects output [CLS] embeddings to sentiment labels and select the label with the highest probability as the final output, which is supervised by cross entropy loss.

For tasks that require pairs of sentences, we started with the attempt of measuring the cosine similarity between mean-pooled BERT embedding for both sentences, inspired by models presented in [8]. However, preliminary experiment of pretraining the paraphrase detection task yields unsatisfactory accuracy of 0.51 on the training set. Thus, we decide to use the approach mentioned in [6] that first concatenates the tokenized results of both sentences with a special [SEP] token, passing the new sentence through BERT, and then apply a linear layer on the returned [CLS] embedding to predict paraphrase or similarity. For both the Quora Dataset and SemEval STS Benchmark Dataset, we use linear layers that turns [CLS] embeddings into a logit that outputs scores. For paraphrase detection, we apply sigmoid to the output score before using binary cross entropy for the loss function. For the semantic textual similarity task, we use mean squared error as the loss function to make the output score close to the label.

4.3 Gradient Surgery

To mitigate the conflict of gradients from different tasks, we apply the algorithm from [4] shown in alg.1. For conflicting (negative inner product) gradients, the algorithm will map both gradients to each other's normal plane, with the detailed pseudocode shown in 1. We adapt the code from [9] to our training pipeline to get the final result.

Algorithm 1: PCGrad

```
Input: Model parameters $\theta$, task minibatch $\mathcal{B} = \{T_k\}$
1  $\boldsymbol{g}_k \leftarrow \nabla_\theta \mathcal{L}_k(\theta)\ \forall k$
2  $\boldsymbol{g}_k^{PC} \leftarrow \boldsymbol{g}_k\ \forall k$
3  for $T_i \in B$ do
4      for $T_j \overset{uniformly}{\sim} \mathcal{B} \backslash \mathcal{T}_i$ in random order do
5          if $\boldsymbol{g}_i^{PC} \cdot \boldsymbol{g}_j < 0$ then
6              //Substract the projection of $\boldsymbol{g}_i^{PC}$ onto $\boldsymbol{g}_j$
7              Set $\boldsymbol{g}_i^{PC} = \boldsymbol{g}_i^{PC} - \frac{\boldsymbol{g}_i^{PC} \cdot \boldsymbol{g}_j}{||\boldsymbol{g}_j||^2} \boldsymbol{g}_j$
8          end
9      end
10 end
11 return update $\Delta\theta = \boldsymbol{g}^{PC} = \Sigma_i \boldsymbol{g}_i^{PC}$
```

4.4 Auxiliary Task

We learn from our experiment that the model's performance on SST dataset is much worse than the other two datasets, which share similarities in many aspects. Therefore, we consider adding an auxiliary task that is similar to SST, hoping that it will improve SST's performance by changing the BERT model's parameters to a direction that favors SST dataset.

We sample 2,500 data points from Yelp dataset[10], with reviews as inputs and stars as labels. The stars in the dataset are floats representing star ratings, including 1.0, 2.0, 3.0, 4.0 and 5.0. The reviews are the comments made by users along with the stars. We sample 500 data points for each star rating, and transformed the star ratings into integers with the value of the original floating points' minus 1, to make it similar to SST dataset. We add the model's loss on the new task together with those of the other three tasks and perform backpropagation as a whole.

5 Experiments

5.1 Data

For semantic classification task(SST), we utilize the Stanford Sentiment Treebank(SST)[2] consisting 11855 sentences from movie reviews and 5 possible classification labels. For the paraphrase detection task(PD), we use a subset of the Quora Dataset [11] with 141,506 question pairs associated with binary label. For the similarity detection task(STS), we used the SemEval STS Benchmark dataset [12] containing 8,628 sentence pairs with their similarity score on a scale from 0 (unrelated) to 5(equivalent in meaning). We use all datasets with a 70/10/20 train, dev and test split. In some experiments, we also use 2,500 reviews together with corresponding star ratings sampled from Yelp dataset with 500 pairs for each label, as described in 4.4.

5.2 Evaluation method

Evaluation for all three main tasks are performed on the train and dev sets, and we also output predictions for test sets. The sentiment analysis and paraphrase detection tasks are evaluated with multi-class and binary classification accuracy correspondingly, and the similarity detection task is evaluated with Pearson correlation with the true labels. We don't evaluate the auxiliary task introduced in 4.4 and used in some experiments, as it doesn't directly relate to our goal.

5.3 Experimental details

The BERT model setting for all experiments consists of 12 layers and loaded with HuggingFace pretrained weights. We utilized the AdamW optimizer for all gradient descent updates, and the learning rate for pretrain(freezing BERT and only train classification heads) is 0.001 while that of finetune(training BERT and classification heads at the same time) is 0.00001 for all experiments. The dropout rate for pretrain is 0.3 and we try different dropout rates when in finetune mode. For all experiments, the model is trained for 10 epochs.

We try different methods of training three tasks at the same time when unfreezing the BERT model. The first method is to treat each task independently when training, where we train the three tasks sequentially (i.e. iterate over the full training set for one task before moving on to the next) in an order of paraphrase detection, sentiment analysis and similarity detection. Specifically, we use a separate optimizer per task, so each task maintains its own learning rate decay schedule. The second method is to sample one batch of a task for all tasks within an iteration, pass the batch to the model and get the loss for the specific task, and then add all losses up and use the total loss to perform backpropagation. The sampling of data is realized via PyTorch Lightening's combined loader [13], under the mode of 'max_size_cycle'. Upon that, we can pass a list of losses for the tasks instead of calculating a sum, then, we calculate the gradient for each loss independently, and use gradient surgery to mitigate their conflicts, before adding the gradients from all tasks up and use them to update the model.

In all experiments regarding calculating the losses of different batches for each task independently and then sum up the losses as a whole for backpropagation in each iteration, we use batch size of 2 for SST dataset and STS dataset, as well as the extra YELP-5 dataset, and batch size of 24 for Quora dataset. The values are selected for two reasons: we want the proportion of data from each dataset in a batch as close to that of the overall datasets as possible; we need to satisfy the GPU memory limit. For other settings, the batch size is set as 32.

5.4 Results

The tasks in all figures' legends are denoted as described in 5.1. We select the model with the best overall performance for each setting.

An overall result on dev set can be found in table 1.

Model	dropout prob	SST acc.	PD acc.	STS corr.	overall
pretrain	0.3	0.390	0.702	0.457	0.516
finetune-seq	0.3	0.510	**0.877**	0.830	0.739
finetune-mix	0.3	**0.524**	0.872	**0.855**	**0.750**
	0.5	0.505	0.872	0.849	0.742
finetune-mix	0.3	0.515	0.871	0.849	0.745
+gradient surgery	0.5	0.513	**0.877**	**0.855**	0.748
finetune-mix	0.3	0.510	0.875	0.851	0.745
+auxiliary task	0.5	0.501	**0.877**	0.846	0.741

Table 1: Model performance on dev set

5.4.1 Multi-task learning methods

In this experiment, we compare three different settings to select the best setting for further exploration. The first setting is denoted as 'pretrain', where we freeze BERT and only train the classification heads. The second setting is denoted as 'finetune-seq', where we iterate over the full training set for one task before moving on to the next task within an epoch. The third setting is denoted as 'finetune-mix', where in each iteration, we train one batch from each task for all three tasks and add up the losses as the loss used for backpropagtion. We use dropout rate 0.3 for all three settings. The result is shown in fig.4.

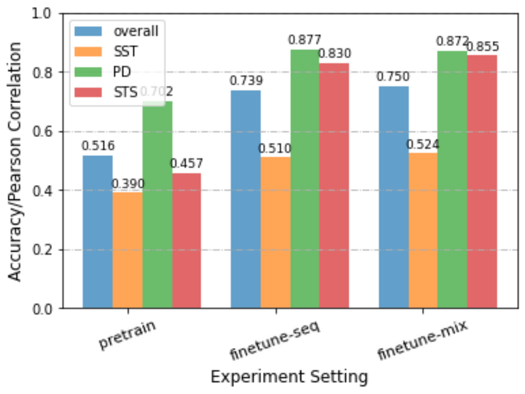

Figure 4: Different setting of multi-task learning

5.4.2 Gradient Surgery

In this experiment, we use the setting of 'finetune-mix' presented in 5.4.1 as the baseline. Upon that, we introduce gradient surgery as described in 4.3 and implementation as in 5.3. We try two dropout rate, 0.3 and 0.5, in this experiment. The format of the x_ticks in the result figure5 is 'baseline/gradient surgery-dropout rate'.

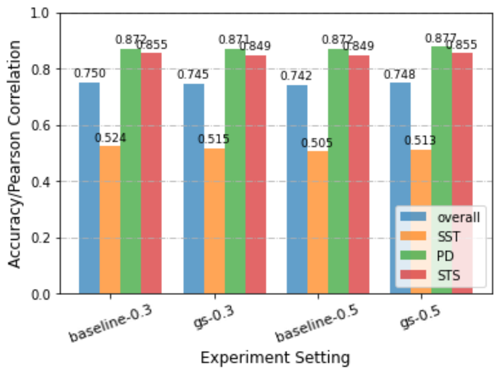

Figure 5: Impact of gradient surgery

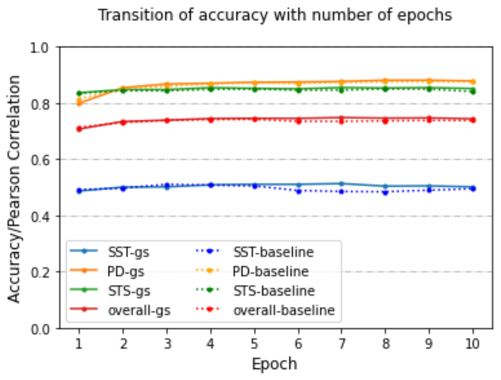

Figure 6: Transition of accuracy when applying gradient surgery

5.4.3 Auxiliary Task

We introduce an auxiliary task on Yelp dataset as described in 4.4. The baseline used in this experiment is also 'finetune-mix' and 'extra' means the model is trained with 4 tasks instead of the 3 main ones, and the number after the '-' means dropout rate.

5.5 Other Exploration

Besides the result shown above, we have also tried adding both gradient surgery and the auxiliary task together to train the model. We try two different methods, where the first one is using the list of four losses from four tasks to perform gradient surgery, while the second one is using the sum of losses of the three main tasks as a whole into the loss list, and then adding the loss of the auxiliary task to the loss list and performing gradient surgery with the two losses. However, neither of the method surpasses baseline.

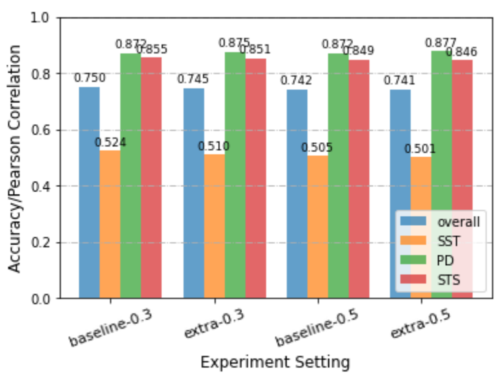

Figure 7: Impact of adding auxiliary task

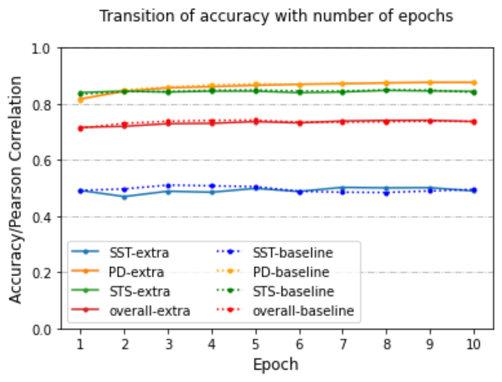

Figure 8: Transition of accuracy when adding auxiliary task

6 Analysis

From our results in 1, it is obvious that unfreezing BERT model weights during fine-tuning leads to a significant performance boost over the model that only updates the prediction heads. Therefore, we can conclude that updating the BERT representation is a crucial factor in enhancing the model's performance. In the following sections, we will analyze the efficacy of each of our experimented modifications. Since the overall accuracy for all trials is very similar (within 1%), we will concentrate on comparing the accuracy of each task relative to the others.

6.1 Training Order

Comparing the finetune-seq and finetune-mix results, we observe that mixing the datasets gives better performance for the SST and STS tasks, as well as general performance. It is worth noting that the PD dataset is more than 10x larger than the other two datasets, which means that it receives more update iterations in the sequential training approach. We suggest that this is the main reason for the performance we observed. In contrast, in the mixed training pipeline, the smaller datasets get iterated more than one time to match the length of the largest dataset, thus leading to a more balanced performance over the three tasks. In addition, as we observe in our experiments, the task order in the sequential pipeline has a large impact on the final accuracy and makes the result less stable.

6.2 Gradient Surgery

During our experiment, we observe that from a certain point on, if further training can improve the model's performance on one task, its performance on another task will become worse. Therefore, we conduct gradient surgery in an attempt to mitigate gradient conflicts of BERT parameters during backpropagation, hoping that it will help increase accuracy on all of the three distinct tasks, thus improve overall performance.

However, in contrast to the results reported in MTRec[5], our experiment results for trials including the gradient surgery technique are similar to the ones without. We observe that, when comparing the best baseline model and the best model that applies gradient surgery, gradient surgery leads to an improvement in PD accuracy and STS correlation, coupled with decrement in SST accuracy. We can also observe that the variation in dropout probability has a more significant impact on the performance difference between baseline models in the SST task than it does on the models with gradient surgery. However, in the PD or STS tasks, the performance difference between the models with and without gradient surgery when changing dropout probability is less or comparable. This phenomenon can be explained by the inherited similarity between the PD and STS tasks – they are both attempting to detect similarity between a pair of sentences. The SST task, on the other hand, aims to detect sentiment information in the input text. Therefore, the PD and STS task require similar text embeddings, generating similar gradients for update, which conflict with the SST gradients. According to how gradient surgery is applied, the overall gradient will be dominated by PD/STS gradients, and the conflicting component in SST-specific gradients might be largely discarded.

In addition, we observed that although the gradient surgery technique applied does not improve best-hyperparameter-performance of the model, it does outperform the baseline when dropout probability is 0.5. This proves that the technique can stabilize training and lead to a more robust solution that is less prone to regularization strategies.

6.3 Auxiliary Task

From our previous experiments, we observe that our model achieves reasonably good results in similarity or paraphrase detection tasks but doesn't perform as well in sentiment classification task. Given this perspective, we augment our training set with an auxiliary rating classification task. However, our results show that the performance of all original tasks are impaired, indicating that including this extra dataset may lead to more severe gradient conflicts that could not be fully resolved.

7 Conclusion and Future Work

In summary, this project aims to find approaches for training robust and semantically-rich BERT representations that can perform well simultaneously on multiple natural language processing tasks. We investigate effects of strategies including changing multi-task training data sampling order, applying the gradient surgery technique, and incorporating an auxiliary task.

Through our experiments, we have demonstrated the effectiveness of mixing all train data in improving overall model performance by mitigating the destructive effect brought by uneven dataset size across tasks. Additionally, our results indicate that the gradient surgery trick enhances model's robustness against regularization while, including an auxiliary task directly, impairs the overall performance, which implicitly suggests the challenging nature of our goal to make the BERT embedding mutually effective for a wide range of downstream tasks.

Overall, our investigations have provided valuable insights on the advantages and drawbacks for each our attempted strategies, and suggest challenges including exploration on more effective use of the presence of auxiliary tasks are still open for future research.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[3] Asa Cooper Stickland and Iain Murray. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671, 2019.

[4] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[5] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over BERT for news recommendation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2663–2669. Association for Computational Linguistics, 2022.

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

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[8] 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[9] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[10] Inc Yelp. Yelp dataset, 2022.

[11] First quora dataset release: Question pairs. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Accessed: 2023-02-12.

[12] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *sem 2013 shared task: Semantic textual similarity. *Proceedings of *sEM*, pages 32–43, 01 2013.

[13] PyTorch Lightening. Combined loader.