Cooking A Multitude of Optimizations for BERT Multi-Task Mastery

Stanford CS224N Default Project

Michael Cho
Department of Mathematics
Stanford University
mycho21@stanford.edu

Michael Peter Hong
Department of MS&E
Stanford University
hongm@stanford.edu

Michael Marcotte
Department of Mathematics
Stanford University
marcottm@stanford.edu

Abstract

In this paper, we are interested in researching and applying advanced NLP techniques to enhance the performance of BERT (Bidirectional Encoder Representations from Transformers) on downstream tasks such as sentiment analysis, paraphrase detection, and semantic textual analysis. We create a multi-task learning model, focusing on three main optimization techniques: random batch scheduling, data augmentation, and gradient surgery. We start by implementing a version of BERT with AdamW optimizer to obtain baseline performance metrics on sentiment analysis. We then build a more robust BERT with a multitask classifier, capable of performing the three downstream tasks simultaneously, yielding strong results which are also reflected on the project leaderboard. The authors describe a multitude of experiments for model improvement, such as changes to the loss function (such as cosine similarity), data augmentation, and hyperparameter optimization. Though many of the experiments provide slight improvements to performance, we find that the most effective changes were experimenting with methods of simultaneously training on the three tasks. In particular, the most effective models utilize interleaving batches from different data sets and random batch scheduling to achieve the authors' best performance on the project leaderboards.

Mentor: Archit Sharma

1 Introduction

Large language models (LLMs) have made significant advancements in the field of Natural Language Processing (NLP). The ongoing ChatGPT revolution has its origins in part in BERT. BERT showcased the novel ability to pretrain on large data with bidirectional representations, able to condition on left- and right- contexts. This has been shown to have performed extraordinarily well on a multitude of NLP benchmarks in the seminal BERT paper [1]. However, in this paper we aim to tackle the interesting and challenging problem of optimizing BERT for multi-task learning; specifically, on the downstream tasks of sentiment analysis, paraphrase detection, and semantic textual similarity. This problem is interesting due to the efficiency and performance improvements possible through multi-task learning. Training a single model to perform multiple tasks can make use of shared representations, improving generalizability and reducing computational costs. However, some challenges that are present include balancing learning across tasks and managing conflicting gradients. Additionally, we want strong performance across all three downstream tasks, not just one specific task; fine-tuning BERT for multiple tasks requires careful tackling of task-specific nuances and clever optimization strategies.

We start with implementing a baseline BERT model, which is pretrained and fine-tuned for sentiment analysis. Then, we explore three main optimization strategies to address the challenges in multi-task learning for BERT: gradient surgery to mitigate conflicting gradients, data augmentation to make our training data more robust and model more generalizable, and random batch scheduling to balance the training process and not be too reliant on any one data. Through our experimentation, we find that these techniques significantly improve the performance of our multitask classifier, as indicated by our performance on the CS 224N leaderboard. Specifically, we were able to achieve minor improvements to our BERT model's performance via changes to the loss function (e.g., cosine similarity) as well as hyperparameter tuning, and major performance improvements via interleaving batches from the different data sets and random batch scheduling.

2 Related Work

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [1]

This groundbreaking paper on BERT (Bidirectional Encoder Representations from Transformers) serves as the foundation and motivation for our project. The authors introduce a novel transformer model that conditions on both leftward and rightward contexts. During pre-training, BERT uses a masked language model and fine-tuning, leading to great improvements on downstream tasks. In the paper, BERT performed extremely well on NLP benchmarks such as GLUE, SquAD, and MultiNLI. A limitation of this paper is that, while BERT excels on various benchmarks, its performance in multi-task learning scenarios involves much more complexities that need to be implemented, which serves as the motivation for our project.

BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning [2]

In this paper, the authors explore a multi-task learning approach using a single BERT model with additional task-specific parameters. They focus on improving performance on natural language understanding (NLU) tasks, by introducing Projected Attention Layers (PALs) to effectively adapt BERT for multiple downstream tasks. They achieve strong performance on the Recognizing Textual Entailment (RTE) dataset and the GLUE benchmark, while using much fewer parameters. We can utilize the techniques explored in this paper to cleverly sample batches in a probabilistically-proportional manner.

Gradient Surgery for Multi-Task Learning [3]

The authors present a novel approach, which they call "Gradient Surgery," to address the problem of conflicting gradients in the context of multi-task learning. The aim is to improve the learning process by aligning gradient directions cross-task. In doing this, our model can achieve improved overall performance in multi-task learning settings. The authors apply this technique to several multi-task learning benchmarks and achieve strong performance, demonstrating the improvements offered by gradient surgery. We are motivated to apply the greedy heuristic outlined in this paper, which although they present greater complexities in our model, show great promise in performance on multi-task learning.

Is Cosine-Similarity of Embeddings Really About Similarity? [4]

In this paper, Steck et al. discuss problems with regularization and methodology that explain the dampening in effectiveness of Cosine-Similarity as an analytical approach to measure similarity between two objects. In our experiments deploying Cosine-Similarity in inference and in training through `CosineEmbeddingLoss()`, we observe similar results pointing to needed future exploration in regularization to render our implementation of Cosine-Similarity useful to the paraphrase detection task.

3 Approach

To serve as our baseline models, we have implemented `bert.py`, `classifier.py`, `optimizer.py`, and `multitask_classifier.py` and submitted our results to the leaderboard. We will refer to the original BERT paper, which describes the transformer model we are using for our baselines [1]. We use multi-headed attention with dropout for our BERT implementation [5]. For our `optimizer.py`, we implemented the AdamW algorithm, as described in the CS224N default project handout [6].

For the `multitask_classifier.py`, we chose to call the BERT model for the forward function and implement a simple set of neural networks that would appropriately transform the output of

BERT into logits needed for the downstream tasks. For the paraphrase and similarity tasks, we decided to concatenate the two `input_ids`, separated by a tensor of `[SEP]` tokens, and similarly concatenated the two `attention_mask`'s with a tensor of 1's before passing the two arguments through the forward function and the dropout and last-linear-layers to obtain the logits as described above. This feed-forward method remained unchanged for most of the top results we saw in our experiments.

Additionally, we experimented with cosine similarity for the paraphrase task. Here, the idea was to obtain two embeddings outputted through BERT that were then fed through linear layers. The two outputted vectors for the sentences in each paired sample were then fed through a cosine similarity to quantify the proximity and similarity of each of the embeddings. This score was then intended to be used as an objective function through PyTorch's CosineEmbeddingLoss function, and on inference for evaluation and test as an intermediary step before filtering by an arbitrary `threshold=0.5` to identify two pairs of sentences as having high probability of being paraphrased [7]. The `CosineEmbeddingLoss` creates a criterion that measure the loss given two input tensors x_1, x_2 and a *Tensor* label y with values 1 or -1, with equation given as follows:

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}. \quad (1)$$

An important element in improving the overall multitask performance for our BERT multitask classifier was our implementation of the joint fine-tuning on the three separate tasks. For our project milestone submission, we did not modify the default implementation of the training process, so the BERT model was fine-tuned exclusively on the SST dataset and thus exhibited very poor performance on the paraphrase and STS tasks. We first implemented our batch scheduler to iterate through the all batches in the order of SST, Quora, and SemEval but we found the training process tedious as the Quora dataset has over 35000 batches with a `batch_size` of 8. We decided to truncate the Quora dataset to 10% of its original size to expedite training by a magnitude of 5 times faster, with the full training of each of our experiments taking around 3 hours running on the `full-model` with a the default 10 epochs.

We further improved the quality of our multitask training by incorporating random scheduling for the batches. The naive multitask training process we had implemented as previously discussed sequentially iterated through the dataloaders for the datasets in order. The random scheduling as discussed in Stickland et al.'s paper on "BERT and PALs" samples batches from task i with probability proportional to N_i, where N_i is the number of samples in the training dataset for task i [2]. The goal was to reduce any bias towards any dataset in particular and backpropgate across each task in a normalized fashion as we step the optimizer for each batch loaded from any dataloader. Without the random scheduler, we might see for instance that when we trained our multitask classifer on the Quora dataset with 35000 batches that our AdamW optimizer had already been stepped ≈ 360000 leading to weaker gradient descent on the objective function for the STS task on the SemEval dataset as it was the last dataset to be trained on for each epoch.

The combination of the above design decisions resulted in an overall rank on the dev leaderboard of 58, just above the median score for class submissions. In order to differentiate our model from the other implementations, we decided to halt optimizations on the paraphrase detection and STS task in favor of boosting the overall sentiment classification accuracy. To do so, two approaches we considered were gradient surgery and data augmentation. The former approach was a greedy solution following the methodology outlined in the gradient surgey paper by Yu et al. that would adjust the gradients obtained from the backpropogation in the paraphrase and STS batches by projecting them onto the gradients obtained from the last SST batch's gradients [3]. Because we implemented a random scheduler for batches, this the most recent gradients from the last SST batch were cached and updated as a new batch was processed.

Our final optimization approach included augmenting the existing training data with the goal of ensembling two models in evaluation on the sst dataset for boosted performance. We leveraged the OpenAI API to carry out data augmentation with the prompt:

```
‘‘Rephrase the sentences words with synonyms so the sentences retain the same
meaning.  Do not replace the proper nouns and movie titles wrapped ‘‘.  Do
```

```
not wrap output in quotation marks"
```

Note that the above script was ran **exclusively** on the training dataset. The result was a parallel SST dataset with rephrased sentences that retained an approximately identical semantic rating to their analogous sample in the original dataset. The goal here was to create augmented data with an autonomous script that required no extra labelling and did not possible infringe on samples present in the dev and test dataset. A model trained on the concatenation of the two datasets, or the ensemble of two models trained separately on each dataset would hopefully allow for more robust and accurate inference.

4 Experiments

4.1 Data

We are utilizing the following datasets for our tasks – **Sentiment Analysis:** SST[1] [8] and CFIMDB datasets, **Paraphrase Detection:** Quora [2] dataset, **Semantic Textual Analysis:** SemEval [9] dataset.

Note that CFIMDB was only used for the movie reviews task for the minBERT checkpoint. We found that overall, the trend with our experiments and the statistics from the top leaderboard submissions to indicate that the sentiment classification task for the SST dataset to be bottleneck in maximizing the overall score. To help maximize the performance of the models on this dataset, we considered data augmentation and gradient surgery as the primary methods to address this weakness in our models.

4.2 Evaluation method

For our evaluation method, we are using the accuracy calculated for the Sentiment Classification and Paraphrase Detection and the overall Semantic Textual Similarity Correlation score as they are calculated for project leaderboard submissions. We also use the calculation for the overall score $\frac{\text{SST}+\text{PARA}+\frac{\text{STS}+1}{2}}{3}$ during model evaluation to save the model checkpoint that was the best overall multitask performer according to this metric.

As a baseline, we first ran an experiment using the given baseline with the provided hyperparameters. In particular, this meant that we used the following: learning rate of 1e-5, hidden layer size of 768, on 10 epochs, training on the full model. The last two layers for each task was just a 2-layer deep neural network.

Next, we tried to change the hyperparameters, such as the learning rate or the number of epochs. For example, the second run used hyperparameters: learning rate of 1e-4, 10 epochs. These first two experiments were ran with the BERT model frozen and showed us that either our last-linear-layers were not robust enough to express the dependencies needed to perform the downstream tasks or that the BERT pre-trained model we were using was just not trained good enough for the task. For the next two experiments (3 and 4), we tried unfreezing the BERT model and by training the full-model, keeping the rest of the architecture the same. We repeated the above experiments with different learning rates of 1e-5 and 1e-6 with 10 and 20 epochs respectively.

Quantitative Results of Experiments (Milestone)				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 1 ($lr = 1e-5$, 10 epochs)	0.527	0.382	0.632	0.132
Experiment 2 ($lr = 1e-4$, 10 epochs)	(unsubmitted)	0.395	0.632	0.132
Experiment 3 ($lr = 1e-5$, 10 epochs)	0.545	0.525	0.564	0.09
Experiment 4 ($lr = 1e-6$, 20 epochs)	(unsubmitted)	0.500	0.472	0.080

[1]https://nlp.stanford.edu/sentiment/treebank.html

[2]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

In general, we found that training the full-model yielded better overall dev accuracy during training however we lost representation as evidenced by the STS correlation scores. These results for the project milestone indicated that pretraining was a dire objective to address the mediocre Paraphrase Detection and STS scores as these were not yet included in the training loop at the time. Following our implementation of the random scheduler, we saw better results for both Paraphrase Detection and STS. For all experiments after the milestone (i.e. for all experiments that we henceforth discuss), we use our implementation of joint fine-tuning on the three separate tasks, as discussed in Section 3 above.

4.2.1 Experiment 5: Last-linear-layer 1e-5 lr

4.2.2 Experiment 6: Full model 1e-5 lr

4.2.3 Experiment 7: Last-linear-layer 1e-5 lr Modified STS loss functions (MSE)

4.2.4 Experiment 8: Full-Model 1e-5 lr Modified STS loss functions (MSE)

Quantitative Results of Experiments				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 5 ($lr = 1e-5$, 10 epochs)	0.580	0.387	0.633	0.442
Experiment 6 ($lr = 1e-5$, 10 epochs)	0.635	0.449	0.624	0.665
Experiment 7 ($lr = 1e-5$, 10 epochs)	0.507	0.308	0.553	0.317
Experiment 8 ($lr = 1e-5$, 10 epochs)	0.753	0.500	0.836	0.843

Here, we noted that as a proof of concept, training the Last-linear-layer was effective in demonstrating that artifacts we implemented were properly training. Next, we experimented with transforming the output layer of the STS dataset to make the output of the last linear layers more interpretable and relevant to the dataset we were working with that had scores ranging from $[0.0, 5.0]$. We regularized the output by applying a `SigmoidMultiplyBy5()` layer that translated the logit output of the last linear layer with dimension 1 into the desired interpretable score. These changes we felt were appropriate to keep for the remaining tests moving forward.

4.2.5 Experiment 9: Sigmoid$() \cdot 5$

4.2.6 Experiment 10: Sigmoid$() \cdot 5$, batch_size $= 16$

Quantitative Results of Experiments				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 9 ($lr = 1e-5$, 10 epochs)	0.762	0.509	0.851	0.852
Experiment 10 ($lr = 1e-5$, 10 epochs)	0.766	0.535	0.857	0.813

The results of these experiments suggested that our current implementation had the potential to crack the top 20 of the leaderboard with an overall high score of 0.77, given enough training epochs and appropriate hyperparameters. Note that we also were still using only 10% of the Quora dataset possibly limiting the potential of the paraphrase detection accuracy we were observing. While we were curious about hyperparameter sweep with modifying parameters such as `hidden_size`, `dropout_prob`, `batch_size`, we felt that these experiments would not be fruitful towards overall improvements in the design and architecture of our model in the long run. We decided it would be more interesting to explore significant changes going forward, namely cosine similarity, gradient surgery, and eventually data augmentation.

4.2.7 Experiment 11: Cosine Similarity

The next modification we implemented was the utilization of cosine similarity. To do so, we modified our last linear layers to output logits for each pair of sentences, which would then be fed through the `CosineEmbeddingLoss` with the targets taken from the labels for each pair. On inference, this meant that the embedding pair would once again be extracted then the cosine operation would be applied to the logits giving us similarity score from which we could determine if the sentences were similar enough to have been paraphrased. We felt this change overall was not very significant to the overall training process so we did not pursue it further as a permanent change to our model's objective and inference functions moving forward.

4.2.8 Experiment 12: Gradient Surgery

The experiments after reverting the cosine similarity implementation focused specifically on optimizing the sentiment classification accuracy. For this task, we needed to output logits for each class that would induce a probability distribution P whose $\arg\max_{c} P(x = c)$ would indicate the class on inference. Gradient surgery's intent was to remove the interference other gradients from other tasks have on updating the model's parameters that would stray them further away from the optimum of those of the conflicting task. We considered both the paraphrase and STS tasks as conflicting with the sentiment classification task, and performed gradient surgery on the paraphrase and STS gradients to ease the conflicts with those from the SST batches. We felt that this was especially appropriate because the paraphrase and STS passed the concatenation of pairs of sentences separated by a `[SEP]` token which had loose relations if any to the SST task which passed a single sentence through BERT.

Quantitative Results of Experiments				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 11 ($lr = 1e-5$, 10 epochs)	0.703	0.505	0.684	0.838
Experiment 12 ($lr = 1e-5$, 10 epochs)	0.745	0.469	0.852	0.832

For experiments 11 and 12, we early stopped when we determined by the accuracy measurements on the dev set that it was clear that the intent of the implementations of cosine similarity or gradient surgery were not improving the accuracy on the SST dataset.

4.2.9 Experiment 13: Data augmentation

For our final experiment, we reran our top model configured using the architecture from experiments 9 and 10 that produced the highest overall score on the new SST dataset. This dataset is a concatenation of the original and augmented dataset we generated using synonyms and rephrasings of the original sentences in the dataset to retain the same semantic meanings. The result is 2 times the training examples for the SST task and a greater representation of SST batches sampled by the random scheduler that brings the proportion of batches from SST from 0.20 to 0.33.

Quantitative Results of Experiments: Milestone				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 13 ($lr = 1e-5$, 10 epochs)	0.752	0.500	0.853	0.808

4.3 Final Leaderboard Submission Experimental details

With our `multitask_classifier.py`, we implemented batch scheduling in our training algorithm in order to improve training time. Specifically, while working with the Quora dataset for our paraphrase task, we concatenate all the batches for each of `sst_train_dataloader, sts_train_dataloader, para_train_dataloader`. Next, we reduce the Quora dataset to 10% of its original size, and apply a random shuffling on our batches. We then run our paraphrase prediction on the resultant batches. This reduces training time approximately by a factor of 7 to around 17 minutes per epoch for a total approximate training time of 3 hours for each run with a default batch size of 8.

For our sentiment classification task, predicted labels were chosen as the class corresponding to the argmax of the probability distribution induced by the 5 logits outputted by the last linear layer of the sentiment model, and the loss was evaluated against the true class using the cross entropy loss function.

For the paraphrase task, predicted paraphrasing truth was calculated using the raw logits, one per sample, outputted from the last linear layer. The logits were then evaluated with the ground truths using the cross entropy loss function.

For the semantic textual similarity task, the concatenation of the two sentences with a separator token was passed through BERT, whose output was then passed through a sigmoid function and scaled to the $[0.0, 5.0]$ scale. The predicted scores were evaluated against the true scores using the MSE loss function.

The learning rate was configured to $1e-5$ with a `batch_size` of 16.

Gradient surgery and cosine similarity and data augmentation were not included in our top submission.

4.4 Final Results

Final Results: Test Leaderboard				
Experiment Number	Overall Score	Sentiment Classification Accuracy	Paraphrase Detection Accuracy	Semantic Textual Similarity Correlation
Experiment 10 ($lr = 1e-5$, 10 epochs)	0.765	0.525	0.849	0.838

The results on the test set match almost exactly what we observed with this model's performance on the dev set. They match our expectations of that model in the context of the other experiments we ran, and the consistency is explained by our selection metric for saving model checkpoints matching the scoring formula. The test results are an indication that our current implementation of the multitask classifier is able to adequately generalize to unseen data.

5 Analysis

Our implementation of the multitask classifier excels at the paraphrase and semantic textual similarity tasks specifically, with mediocre performance in sentiment classification accuracy. In general, for a discrete multiclass classification task like SST, it's sometimes difficult for models to pinpoint the exact class and a better metric to measure the performance of the model can be top-k accuracy, or the accuracy of whether the true class was in the top k probabilities induced by the softmax function applied to the outputted logits of the model. However here, the goal of optimizing for SST is to have a perfect top-1 model which can be difficult depending on the quality of the dataset as we have realized.

In this project, we explored many different ways of tackling the multitask paradigm including scheduling, data augmentation, and gradient surgery. For the purposes of maximizing leaderboard score, it appeared that using none of these approaches was the best option which is counter-intuitive to what we expected when we developed our hypotheses on how these methodologies would impact the final results. While we were able to confirm from an academic standpoint that the algorithms and formulas we implemented were working correctly, more finesse to see better results. For instance, while cosine similarity was proving useful for the paraphrase detection task, the gradients obtained from backpropogation in our implementation were significantly lower in magnitude compared to those obtained in the other tasks which led to a very slow convergence of the paraphrase task compared to the others making it unfeasible to deploy in future versions of models we were developing. This could have been corrected by adding regularization terms to the gradients or adjusting the step function of the optimizer to more heavily weight the gradients from the paraphrase task in its update so that we could see an even convergence of all three tasks as the multitask classifier was trained. However, this design decision would introduce another parameter in our hyperparameter search making it more difficult than it already was to search for the optimal hyperparameters.

6 Conclusion

The main findings of our work is that multitask fine-tuning and random batch scheduling were by far the most influential improvements to the multitask classifier. This aligns with our initial hypothesis early on in the project that data, actually fine-tuning for each task, and sampling would be the most important aspects to successfully developing a robust multitask model. Regarding data, we really wanted to use a Large Language Model as a generator that would allow us to explore the General Adversarial Network paradigm and potentially deploy DPO on the LLM to fine-tune its output for the data we would ultimately feed to our multi-task classifier for fine-tuning. However, there was an unavoidable gray-area with possibly infringing on the dev and test data that we could not guarantee because we don't know how the model's we were considering using were trained. This means that it was very possible that if we prompted an LLM to generate more samples for the SST dataset that some of these samples would be heavily correlated with the samples that would appear in the test dataset that we could not necessarily control without sophisticated pruning that ultimately wouldn't be worthwhile pursuing as an experiment in this project. We settled on exploring data augmentation as another avenue to obtain a more robust model through this new training set we generated that we could verify was within the guidelines of the Honor Code.

One clear downside of the experiments we performed in this project was the lack of runs on each configuration. While we covered a lot of design decisions and techniques to bolster our model's performance, due to the average training time being significantly long we weren't able to adequately evaluate the effect each decision had on the overall model. Ideally, we would have multiple runs for each experiment where we could report the average and top statistics for each task, and furthermore ideally the number of epochs would be in the 20-30 range to cover more ground in the search for the optimal weights. However, performing 5 runs for an experiment at 30 epochs would take over 45 hours and this was not feasible with the 13 reported experiments we included in this report. There was also not a naive or obvious way to early-stop because our best runs did not follow desired approximately monotonic improvements in reported dev accuracy.

Regarding future work, there is plenty of potential to run sophisticated hyperparameter sweep on the model configurations we have implemented and without a doubt there is a configuration of hyperparameters we didn't select that would've yielded a better raw overall score. Regarding implementation, we left the BERT and optimizer implementations untouched. There is definitely potential for a smarter optimizer, especially since we shared it across tasks, that better steps in a multitask setting.

While not all the ideas we implemented were necessarily successful permanent additions to the overall multitask classifier, their exploration helped us isolate the unavoidable systematic problems we were facing when it comes to designing and training a deep learning model in data and generalization.

7 Ethics Statement

One ethical concern surrounding this project is that NLP models (such as sentiment analysis algorithms) can reflect human biases from the datasets they are trained on. This challenge, discussed very often throughout the past quarter, can manifest through phenomena such as gender bias or racial bias in NLP models. For example, a model may evaluate a difference in sentiment between words like “man” and “woman”. Leaving such biases unchecked in our research can perpetuate a trend of discrimination or prejudice (in a wide variety of contexts), and is thus an issue that deserves to be properly addressed. One possible method of mitigating such bias is by tagging/labelling/augmenting the training datasets with information about similar words (i.e. words that we want to be treated similarly, without bias separating them). This could be done for every form of bias that we are concerned about. This is an especially concerning issue for our project because we are using BERT, which has become an incredibly widespread model for any tasks in the neighborhood of this project's focus; even besides semantic classification, paraphrase detection, and STS, there are many other use cases within the realm of token-level or sentence-level NLP tasks. One prominent example could be to augment browser search to better match the user. Since we are using such a popular architecture, it is dangerous to leave bias (and other ethical issues) unchecked. In particular, we run the risk of contributing to a precedent where the research community ignores bias and other ethical issues, putting it off for another day. It is imperative that, as researchers, we remain conscious of such trends and concerns as we conduct our research.

Other similar concerns (that are commonly found in many NLP models) include the lack of explainability for NLP models; the lack of clarity on how our system works on the inside may pose a serious challenge. This is especially true in sensitive applications like hate speech (one could imagine wanting to do a semantic analysis on hate speech, for example) or other text with discriminatory microaggressions (for example, sentiment analysis on phrases like "you're attractive for an Asian person" can become ambiguous). The solution to this is to implement models that are designed to be more explainable; though we are no experts on this topic, one method to make a model "explain" itself is to use a gradient method. For example, we could implement a method to analyze the change in sentiment or decision class based on the tokens in the input example, which would then highlight which words had the largest impact. If we were to implement this mitigation strategy, we could also utilize the attention mechanism in BERT to explain itself (it would be helpful to show the connections between heads and attention layers). As we get closer to achieving our goal of creating a perfect model for the three tasks described in this project, it becomes increasingly important to keep these general concerns in mind. The better a model is, the more incentive users have to blindly trust it; but this is all the more reason why we continue to consider how to justify a model's decision to call two sentences similar, or to analyze text a certain way, or anything of that sort. In this case, we are the only ones who can keep our models in check, and it will only become more important as our models become more sophisticated.

Ethical concerns could also be raised by our usage of Google Cloud and OpenAI's API. Regarding the former, carbon emissions are almost unavoidable with the expenditure of energy locally and virtually over the cloud. There are actions we could have taken to be more conscious of carbon emissions which include testing fully remotely with tmux so locally our laptops would be conserving energy, and also our choice to use a GPU that was not necessarily configured to output low carbon emissions. Because Google's Cloud platform informs users of the carbon emissions they may generate, ethically we arguably overstepped this boundary by overlooking this statistic in our virtual machine setup. Additionally, for performing data augmentation we utilized OpenAI's API to execute a script that would paraphrase and perform synonym replacement on the data included in the dataset. Here, we could've possibly overstepped boundaries related to proper usage of this dataset by not specifically checking the terms and conditions of the data we were using. Furthermore, we also utilized a large language model to perform this augmentation which has its own set of concerns with ethical usage. Ways of mitigating this potential ethical concern would be to explicitly confirm whether what we are doing lies within terms and conditions both outlined by the dataset providers and OpenAI, and if not, finding other open-source ways to perform this step.

References

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

[2] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning, 2019.

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

[4] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM on Web Conference 2024*, WWW '24. ACM, May 2024.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] PyTorch. Cosineembeddingloss — pytorch 2.0 documentation. `https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html`, 2023.

[8] 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 David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[9] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *SEM 2013 shared task: Semantic textual similarity. In Mona Diab, Tim Baldwin, and Marco Baroni, editors, *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, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.