UnBERTlievable: How Extensions to BERT Perform on Downstream NLP Tasks

Stanford CS224N Default Project

Sophie Andrews
Department of Computer Science
Stanford University
sophie1@stanford.edu

Naomi Boneh
Department of Computer Science
Stanford University
naomicyb@stanford.edu

Abstract

In the modern world, it is impossible to accurately digest all information. We rely on summary tactics like sentiment analysis to understand the overall opinion towards a service or paraphrase detection to eliminate redundant information. In this paper, we develop a minBERT implementation, and then apply multiple extensions to fine-tune the model for the three tasks of sentiment analysis, paraphrase detection and determining semantic textual similarity. We apply a combination of techniques including cross-encoding our inputs, gradient surgery, and careful selection of fine-tuning data to achieve our best model that has significant improvements over the base minBERT model.

1 Introduction

The development of the pre-trained language model BERT set state of the art performance on many sentence classification and sentence-pair regression tasks. The abilities of the BERT model can aid significantly in data aggregation and providing statistical analysis on large amounts of data. However, in order to understand different ways of conveying meaning and achieve strong performance, BERT requires fine-tuning bidirectional pre-trained representations on individual downstream tasks, which can be computationally intensive (Devlin et al., 2019).

In this paper, we present our multi-task learning strategy to advance the performance of the base minBERT model on sentiment analysis, paraphrase detection, and semantic textual similarity tasks. This strategy primarily leverages cross-encoding for sentence pair tasks and fine-tuning on datasets through running pre-training epochs. In addition, we evaluated the efficacy of several multi-task fine-tuning techniques including using different loss functions, SMART regularization, gradient surgery, and making slight modifications to data labels. In doing so, we hope to implement and present a model with the capabilities of pre-trained BERT that also addresses the existing difficulties with multi-task learning.

2 Related Work

Here, we introduce BERT and discuss various adaptations to improve the performance of this model. BERT is a pre-trained transformer network that can accomplish downstream NLP tasks. The original BERT paper employs cross-encoding, in which input sentences are first concatenated with a `[SEP]` token, then passed into BERT to produce a single embedding (Devlin et al., 2019). This technique is largely used for tasks where the input is a pair of sentences, including paraphrase detection and semantic textual similarity. Hence, we were motivated to explore this technique in our enhancements to the BERT model.

Another paper presents a new tool, Sentence-BERT or SBERT, which improves the performance of finding semantically similar sentences (Reimers and Gurevych, 2019). One limitation of the

original BERT model is that it does not produce independent sentence embeddings. SBERT addresses this limitation and achieves a speedup by passing sentences independently through the same BERT network, using a siamese network architecture to ensure that each network has the same weights, and the result is a fixed size embedding.

With fixed-sized sentence embeddings, cosine similarity, a fast calculation, can be computed to find similar sentences. This method falls under a technique called bi-encoding, which differs from cross-encoding in that the input sentences are considered separately. We were inspired by this research and its use of cosine similarity to experiment with different methods of combining embeddings.

When leveraging multi-task learning, it is possible for the gradient directions of different tasks to interfere with each other, affecting performance. In Yu et al. (2020), researchers propose the use of gradient surgery to mitigate this issue by projecting one task's gradient onto the normal plane of a competing gradient from another task. They concluded that this method improves performance on multi-task supervised problems.

Lastly, when overly fine-tuning large pre-trained models on smaller downstream tasks, overfitting can negatively affect performance. Jiang et al. (2020) describes the regularization technique SMART, which employs both smoothness-inducing adversarial regularization and Bregman proximal point optimization, to balance the model's learning of task-specific configurations and its ability to adapt to new data.

3 Approach

3.1 Network Architecture

Following the architecture detailed in Devlin et al. (2019), we implemented multi-head self-attention and the transformer layer features of a base minBERT model. The embedding layer contains a word embedder, position embedder, and 12 BERT encoder transformer layers. Every transformer layer consists of a multi-head attention layer, an additive and normative operation with a residual connection, a feed-forward layer, and another additive and normalization layer with a residual connection. We use the AdamW optimizer and implemented decoupled weight decay regularization.

In our final approach, each task has its own dropout layer and linear layer. We use different loss functions for each downstream task because each task has a different objective. For both sentiment analysis and paraphrase detection we use cross-entropy loss because there are multiple, discrete output classes. For semantic textual similarity, we use mean-squared error (MSE) loss because the outputs are continuous.

Our baseline is our base minBERT model, which achieves an accuracy of 0.489 on the Stanford Sentiment Treebank (SST-5) dataset, an accuracy of 0.676 on the Quora Question Pairs (QQP) dataset, and a Pearson correlation coefficient of 0.29 on the SemEval STS dataset.

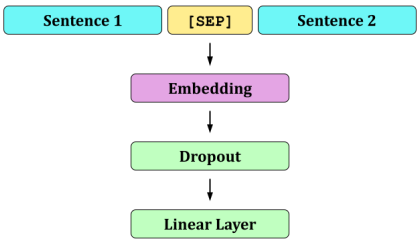
3.2 Downstream Tasks

After obtaining the encoded pooled representations of each sentence of our data, we fine-tuned the base minBERT model to our downstream tasks. For each task, we created a dropout and linear layer. For sentiment analysis, we obtained logits for the five different sentiment values. For paraphrase detection and semantic textual similarity, our input consisted of two sentences. We experimented with bi-encoding and cross-encoding techniques before outputting a prediction. In both cases, we predicted a single logit that either represented whether two sentences were paraphrases of each other or a logit that represented how similar the two sentences were.

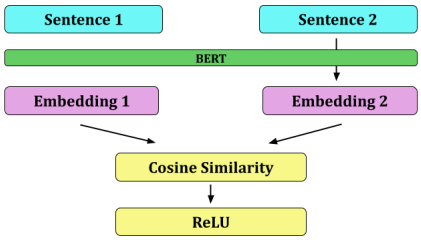
3.3 Encoding Methods: Cross-encoding and Bi-encoding

The paraphrase detection and semantic textual similarity tasks take in two sentences as input to determine similarity between the two sentences. We evaluated three strategies related to embeddings for sentence-pair tasks: cross-encoding, bi-encoding, and bi-encoding with cosine similarity.

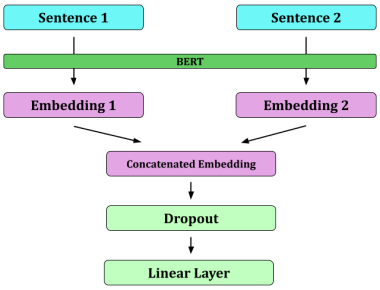
Cross-encoding is explained in (Devlin et al., 2019), and involves concatenating the two input sentences with a `[SEP]` token before feeding the input into the BERT model. Dropout and a linear layer are then applied to the resulting embedding to output a prediction. The cross-encoding technique



(a) Cross-encoding with a linear layer



(b) Bi-encoding with cosine similarity



(c) Bi-encoding with a linear layer

Figure 1: Encoding Methods for Sentence-pair Tasks

is more effective for learning the relationship between the two input sentences, as the embedding itself captures this relationship.

With bi-encoding, each input sentence is independently passed into BERT, then concatenated into a single embedding and passed through a linear layer. For the semantic textual similarity task only, we also tested bi-encoding with cosine similarity to determine the degree of closeness. Once we had the two sentence embeddings from BERT, we applied cosine similarity as a measure of closeness in semantic meaning. Then we applied ReLU to ensure the result was between 0 and 1, and then multiplied the result by 5 since the final scores are between 0 and 5.

3.4 Training Techniques

Given that each dataset was a different size, we used different sampling techniques to take full advantage of all data that was provided. We started with using a sampled round-robin approach, which fully iterated through the STS dataset since it was the smallest, but did not make full use of the SST or QQP datasets. To address this, we assessed different staggered fine-tuning strategies, where we trained only on the full SST dataset for a certain number of epochs, on the full QQP dataset for a number of epochs, and then continued the original round-robin approach.

We discovered the most optimal epoch configuration was to train on the full SST dataset for 2 epochs, train on the full QQP dataset for one epoch, and then use round-robin sampling for the remaining 7 epochs. Pre-training of the datasets for sentiment analysis and paraphrase detection enabled the model to fully take advantage of these larger datasets and perform better on the corresponding tasks without causing overfitting through continuous cycling of the smaller STS dataset.

3.5 Additional Methods

We utilized the `Pytorch-PCGrad` module (Tseng, 2020) to implement gradient surgery in our model in order to reduce potential destructive interference between task gradients during multi-task learning (Yu et al., 2020). This method makes direct changes to an i-th task's gradient $\mathbf{g_i}$ as follows, where $\mathbf{g_j}$

is the gradient of a competing task:

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

Additionally, we used the `smart-pytorch` module (Schneider, 2022) to integrate the SMART technique, specifically the smoothness-inducing adversarial regularization method detailed in (Jiang et al., 2020). This approach is meant to mitigate overfitting, which is a particular risk with the similarity score task given that the STS data is the smallest dataset, and hence is cycled through more than other datasets during training. It works by reconfiguring and solving the following optimization to use a smoothness-inducing adversarial regularizer with the standard loss function for fine-tuning:

$$min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta)$$

In the equation above, $\mathcal{L}(\theta)$ is the loss function, λ_s is a tuning hyperparameter. $\mathcal{R}_s(\theta)$ is the moothness-inducing adversarial regularizer. $\mathcal{R}_s(\theta)$ is defined with the following equation:

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

When leveraging SMART, we apply Kullback-Liebler (KL) loss in addition to cross-entropy loss to sentiment analysis and paraphrase detection, as they are both classification tasks. Specifically, the cross-entropy loss is perturbed by a small amount according to the KL loss and a defined weight, and we set this weight to be 0.02. For semantic textual similarity, we use MSE loss for both the actual loss and the degree of perturbation to this loss. SMART injects perturbation to encourage $\mathcal{F}(\theta)$ to be more smooth, which is meant to prevent overfitting and improve generalization.

4 Experiments

4.1 Data

For sentiment analysis, we used the Stanford Sentiment Treebank (SST) dataset made up of sentences from movie reviews. The dataset has 8,544 `train` sentences, 1,101 `dev` sentences, and 2,210 `test` sentences. Each sentence is rated negative, somewhat negative, neutral, somewhat positive, or positive. We also used the CFIMDB dataset of polar movie reviews. There were 1,701 `train` sentences and 245 `dev` sentences that we used. We did not use the `test` sentences. Each sentence is labeled positive or negative.

We used the CFIMDB dataset in two different ways. In the first experiment, we enhanced the SST `train` set by adding in the CFIMDB `train` and `dev` examples. Given that the CFIMDB reviews are highly polar, we thought it was reasonable to assign CFIMDB's positive reviews the same positive label as used in the SST dataset and assign CFIMDB's negative reviews the same negative label as used in the SST dataset. Given the polarity, we did not feel like we had to indicate that any sentences should be labeled "somewhat positive" or "somewhat negative".

In our second usage of this dataset, we trained an extra head in our model to perform classification on the CFIMDB dataset to update the overall weights of our model. In this case, we did not have to change any labels in the dataset.

For the paraphrase detection task, we used the Quora Question Pairs (QQP) dataset. Each item in the dataset is a pair of questions with a label indicating whether one question is a paraphrase of another. There are 283,010 `train` examples, 40,429 `dev` examples, and 80,859 `test` examples.

To determine the semantic textual similarity between a pair of sentences, we use the SemEval dataset. Unlike the paraphrase detection task, the semantic similarity of two sentences is not indicated by a binary label, but instead by a score on a continuous scale of 0-5, where 0 indicates no similarity and 5 indicates same meaning. There are 6,040 `train` examples, 863 `dev` examples, and 1,725 `text` examples.

4.2 Evaluation method

Since the labels in the sentiment analysis and paraphrase detection tasks are discrete, we evaluate our model using accuracy percentage. We measure number of correctly labeled examples as a fraction

of the total number of examples. To evaluate the similarity task, which has continuous labels, we calculate the Pearson correlation coefficient between the true similarity scores and our predicted scores as described in (Agirre et al., 2013). We measure our model's total performance on the leaderboard using this formula: $\frac{\text{SentAcc + ParaAcc + 0.5 + 0.5(SimCoeff)}}{3}$.

4.3 Experimental details

We ran our experiments on an Nvidia GPU through Google Compute Platform. For the model itself, we tried two learning rates of 1^{-5} and 2^{-5}. For our AdamW optimizer, we used a learning rate of 1^{-3} and weights of 0.9 and 0.999, as outlined in the original Devlin et al. (2019) paper. For each of our dropout layers, we used a probability of 0.1. We tested batch sizes of 8 and 16.

Each experiment took approximately two hours, where one hour was spent in the three pre-training epochs using the full SST and QQP datasets, and another hour on the seven training epochs that use a round robin strategy to train on all three tasks.

4.4 Results

We report the results of all experiments we ran, indicating whether the results were from the `dev` or `test` set, and analyze them below. For all experiments, we used the full model (instead of just the last linear layer) so the BERT embeddings could reflect the latest state of the fine-tuned model. At the time of this writing, we were 29 on the TEST leaderboard, and 22 on the DEV leaderboard.

We refer to our best model as BestBERT. BestBERT uses cross-encodings for the paraphrase and similarity tasks, and conducted two pre-training epochs on the full SST train set, followed by one pre-training epoch on the full QQP train set, followed by seven epochs using a round robin strategy to sample from all three train sets and update the weights of the model.

Model	**Total Score**	**SentAcc**	**ParaAcc**	**SimCoeff**
Baseline: minBERT (`dev`)	0.603	0.489	0.676	0.290
BestBERT (`test`)	0.779	0.513	0.881	0.888
BestBERT + gradient surgery (`test`)	0.779	0.511	0.882	0.889

Table 1: Our two best models show significant improvement over the baseline, with the biggest improvement occurring in the semantic similarity task.

We also incorporated gradient surgery into our BestBERT model. Interestingly, the overall score remained the same because the paraphrase and similarity scores increased while the sentiment analysis score decreased.

Gradient surgery improved the results for the two tasks that involve pairs of sentences, since it is able to better leverage the shared information between the sentences to make a prediction. However, when using gradient surgery, the sentiment classification accuracy suffered, likely because the fine-tuning that learned to compare two sentences was not able to use that same approach when there was only one sentence to label in the sentiment classification task.

To compare our different models and how we determined which to submit to the TEST leaderboard, we looked at results on the `dev` set. All results below are reported on the `dev` set.

We first tried to uncover how embedding strategies would affect our results. In the "Combo" model, we used cross-encoding for the paraphrase detection task (Fig. 1a) and bi-encoding with cosine similarity for the similarity score task (Fig. 1b). Using cosine similarity was specifically used to target the semantic similarity task, yet that ended up performing worse than cross-encoding. This shows that the weights of the linear layer learned more complex information than just taking a cosine similarity between two embeddings. As expected, using bi-encoding (Fig. 1c) performed the worst since these embeddings only considered each sentence in the pair in isolation, not together. Since the paraphrase detection task and the similarity score task heavily rely on the relation between the sentences, the embedding methods considered both sentences together performed much better. Based on the results in Table 2, we used cross-encoding in all further experiments.

Our datasets were different sizes. When training on all three tasks simultaneously, we had to ensure that we were using the same number of examples. However, that left many of the examples in the SST

Model	Total Score	SentAcc	ParaAcc	SimCoeff
Cross-encoding (BestBERT, `dev`)	0.784	0.529	0.880	0.884
Combo (`dev`)	0.778	0.525	0.881	0.843
Bi-encoding (`dev`)	0.659	0.526	0.766	0.368

Table 2: Effects of different encoding strategies on model performance. We chose cross-encoding as it had best overall performance.

and QQP datasets unused. Therefore, we decided to fine-tune on these datasets by running pre-training epochs at the beginning of our experiment. These pre-training epochs involved training on the full dataset for the respective task. We experimented with the number of pre-training epochs to use for each of the sentiment analysis and paraphrase detection tasks. We were pleasantly surprised that two fine-tuning epochs on SST and one fine-tuning epoch on QQP provided such a large improvement, highlighting the importance of a large and diverse training set. While the similarity scores generally remained the same across experiments, as expected since our main focus was how best to use the STS and QQP datasets, these experiments revealed a tradeoff between the sentiment analysis accuracy and paraphrase detection accuracy.

When adding gradient surgery to the model, we saw that paraphrase accuracy and similarity scores increased, while sentiment accuracy decreased. We then ran this experiment again with an additional pre-training epoch on the SST dataset in order to increase the sentiment classification accuracy. However, this led to overfitting as both sentiment classification accuracy and paraphrase detection accuracy decreased. Using only one fine-tuning epoch on SST clearly revealed the tradeoff between the sentiment analysis and paraphrase detection tasks.

Model	Total Score	SentAcc	ParaAcc	SimCoeff
2 epochs SST, 1 epoch QQP, 7 epochs RR (BestBERT, `dev`)	0.784	0.529	0.880	0.884
3 epochs SST, 2 epochs QQP, 7 epochs RR (`dev`)	0.711	0.520	0.673	0.882
2 epochs SST, 1 epoch QQP, 7 epochs RR + gradient surgery (`dev`)	0.784	0.524	0.883	0.885
3 epochs SST, 1 epoch QQP, 7 epochs RR + gradient surgery (`dev`)	0.778	0.516	0.876	0.885
1 epoch SST, 1 epoch QQP, 7 epochs RR + gradient surgery (`dev`)	0.779	0.509	0.887	0.883

Table 3: Effects of pre-training on model performance. Given the wide variation in the sizes of the datasets, we pre-trained on the larger datasets in order to ensure that we took advantage of all data that was available to us. RR means round-robin, where we sampled from all three datasets while training on all tasks in a single epoch.

Since our model consistently performed worse on the sentiment analysis task and given we had another dataset to leverage, we decided to incorporate the CFIMDB `train` and `dev` examples. We either directly added the examples to the `train` set or we trained a separate head on the CFIMDB dataset. Unfortunately, neither of these experiments improved our results, likely indicating that the CFIMDB examples did not align with the existing SST examples.

Model	Total Score	SentAcc	ParaAcc	SimCoeff
Just SST (BestBERT, `dev`)	0.784	0.529	0.880	0.884
Combine SST and CFIMDB data for `train` (`dev`)	0.779	0.520	0.877	0.882
Train separate head with CFIMDB (`dev`)	0.778	0.519	0.882	0.867

Table 4: Effects of using the CFIMDB data for training the model. While we were hopeful that this new dataset would improve our sentiment analysis predictions, it did not.

Next, we turned to the SMART algorithm to improve overall performance given that it smooths the model by penalizing sharp variation in predictions. SMART actually reduced the performance of our

model, suggesting that there was not any sharp variation to begin with and the SMART algorithm overcompensated.

Model	Total Score	SentAcc	ParaAcc	SimCoeff
No SMART (BestBERT, `dev`)	0.784	0.529	0.880	0.884
With SMART (`dev`)	0.759	0.511	0.870	0.879

Table 5: Effects of using the SMART algorithm for regularization. Our model performed better without it.

Our final experiments involved small changes to the architecture of the model. The default learning rate was 1^{-5}, but we also tried 2^{-5} used by Devlin et al. (2019). This led to a significantly worse performance, particularly for the sentiment analysis task. We also tried a batch size of 16 to increase the stability in each update to the model. However, this also decreased the performance, indicating that a batch size of 8 already captured enough detail about the sentences without high variance in the gradients.

Model	Total Score	SentAcc	ParaAcc	SimCoeff
LR 1^{-5}, batch size 8 (BestBERT, `dev`)	0.784	0.529	0.880	0.884
LR 1^{-5}, batch size 16 (`dev`)	0.776	0.511	0.878	0.880
LR 2^{-5}, batch size 8 (`dev`)	0.770	0.493	0.876	0.882

Table 6: Effects of small changes in model setup

5 Analysis

We noticed a general trend in our experiments that there was a tradeoff between performance in sentiment analysis and paraphrase detection tasks. We believe this tradeoff exists because both tasks require discrete labels but the actual prediction relies on different features of the sentence. Paraphrase detection takes in two sentences while sentiment analysis only takes one. Furthermore in sentiment analysis, we care more about tone and emotion of the words, whereas paraphrase detection is based on meaning. Since the sentiment similarity scores are continuous and not discrete buckets, we believe this task is more resistant to the tradeoff.

Overall, sentiment analysis was the hardest task. This was not surprising, as a 5-bucket task is harder than a binary classification. It can also be hard for a even a person to detect the line between a "somewhat positive" and a "positive" sentiment. To further analyze the results, we created a confusion matrix (Fig. 2) for this task.

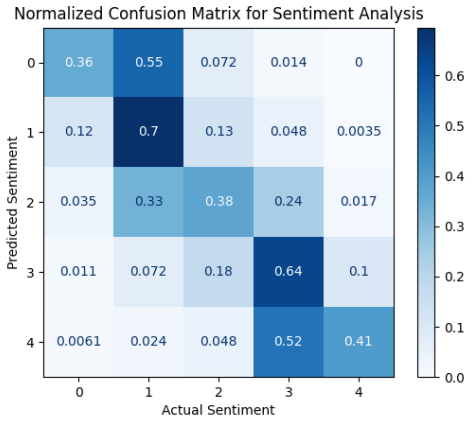


Figure 2: Normalized Confusion Matrix on the SST dataset

We see that the correctly classified examples are concentrated in the "somewhat positive" and "somewhat negative" categories, but our model had more trouble determining what made an example extreme enough to be labeled "positive" and "negative", or have no strong sentiment to be labeled "neutral". In these three categories, there were many classifications to neighboring labels, showing that the distinction is difficult. We believe that this is a reflection of the difficulty of the task, even for a person. For example, the true label for the sentence "It's a lovely film with lovely performances by Buy and Accorsi." is "somewhat positive" while our model rated it as "positive". We prefer our model's rating here, due to the presence of the two positive adjectives "lovely".

This unclear distinction in labels is perhaps why using the CFIMDB dataset for sentiment analysis actually made our predictions worse. Even though the CFIMDB reviews were described as "highly polar", they may have been reviewed on a different scale than the SST datasets. Since our model already struggled with the nuances between categories, having sentences that were labeled "positive" from two different sources may have further confused the model.

Our model consistently achieved a high Pearson correlation coefficient on the semantic similarity task. We believe this is due to the continuous nature of the scores. Even if our model did not perfectly predict the similarity score, it was still able to predict a score in a reasonable range. Furthermore, given the STS dataset was the smallest, we were always using the full dataset when training on all three tasks, which benefitted this task. For sentiment analysis and paraphrase detection, we only used the full dataset in the pre-training epochs and sampled from them when training on all three tasks.

6 Conclusion

After evaluating multiple different techniques, we discover that our model achieves its best performance through using cross-encoding and pretraining on the SST and QQP datasets in their entirety. Specifically, we train on the entire SST dataset for two epochs, train on the entire QQP dataset for one epoch, then use round-robin sampling of all three datasets for the remaining seven epochs. We learned that the implementation of the SMART technique did not improve our model performance in any tasks. Gradient surgery improved performance on the paraphrase detection and semantic textual similarity tasks, but worsened performance on sentiment analysis, making the overall score unchanged.

Our model attains an overall score of 0.779. It achieves a final accuracy of 0.513 for sentiment analysis, an accuracy of 0.881 for paraphrase detection, and a Pearson Correlation of 0.888 for semantic textual similarity. Compared to our baseline, it yields an improvement of 0.024 (4.9% increase) for sentiment analysis, 0.205 (30.3% increase) for paraphrase detection, and 0.598 (206.2% increase) for semantic textual similarity. Relative to other submissions on the TEST leaderboard, our model performs best on the semantic textual similarity task.

Some limitations of our model include that currently, it can only accomplish NLP tasks in English. Additionally, the SST dataset used for sentiment analysis is solely targeted towards movie reviews, resulting in the model not learning sentiment nuances in other forms of text. Further, the QQP dataset is quite large for the use of paraphrase detection, but if the model were to conduct this task for a high-risk scenario, e.g. healthcare data, it should be trained on data related to this specific application.

7 Key Information to include

Mentor: Tim Dai External Collaborators: None Sharing project: No

For code, Naomi integrated the gradient surgery and SMART algorithms into the model. Sophie worked on the different embedding strategies and the different sampling techniques. We both equally contributed to the final paper.

8 Ethics Statement

Language is constantly evolving, with the development of both new vocabulary and novel usage of existing words. Hence, we expect that our model will require continuous maintenance and appropriate modifications to its training data in order to avoid producing outdated and inaccurate results.

We have made deep ethical considerations specific to each of the three tasks that our model is trained to execute. To begin with, we trained our model to have the ability to detect whether questions are paraphrases of each other using Quora questions. However, our same model could be deployed to a forum specifically containing healthcare questions. It would be a problem if the answers for all questions that were deemed paraphrases were grouped together, since there could be conflicting healthcare information in this aggregated answer if questions were incorrectly identified as paraphrases. If two questions were incorrectly grouped, they might recommend two different treatments, and one might actually be harmful for the person's intended query.

To mitigate this, we can ensure that questions always appear with their answer, so a user can see specifically what answer a questions is addressing. We can also display a rating of how well we think questions are paraphrases of each other, so a user can pay more attention if two questions have more uncertainty of whether they are actually paraphrases.

Regarding sentiment analysis, we have considered that the expression of sentiment differs depending on cultural contexts and connotations. Some social groups may place stronger emphasis on politeness even while expressing negative sentiment, while members of another demographic may prefer to use strong or jarring language to describe something positively. This issue exemplifies the possibility of our model having higher rates of incorrect sentiment classification for certain demographics, which could lead to misunderstanding and discrimination against these groups.

This model could also be used to analyze feedback for a service to determine people's satisfaction with the service. However, if only a certain demographic or community typically used this service, and if features of their vernacular were underrepresented in the training data, then the model could incorrectly classify clients' sentiment about this service and portray the service and as better or worse than it really is. This then has implications for the success of the service and the business providing it, and may mislead the public. One possible mitigation strategy is to add more training data with as much diverse vernacular as possible, or to make the training data specific to the context that the model is working in.

Lastly, the ability of our model to perform semantic textual similarity prompts open-ended ethical inquiries and debates about the definition of plagiarism, as well as the efficacy of plagiarism detection tools. Should it be considered plagiarism if our model determines that two texts have extremely similar semantics? If an individual were to use our model as a plagiarism detection tool, this user could also hold significant power in that they can decide what the threshold value of semantic similarity for which texts are considered plagiarism. A threshold that is set too low can lead to a high false positive rate and have implications for the acadmeic and social reputation of students, and all writers in general. One method to mitigate this issue is for us, as developers of the model, to publicly advocate for and recommend a standard threshold value for plagiarism classification.

References

Eneko Agirre, Daniel Matthew Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *International Workshop on Semantic Evaluation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Flavio Schneider. 2022. Flavio schneider/smart-pytorch.

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

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