minBERT and Downstream Tasks

Stanford CS224N Default Project

Bay Foley-Cox
Department of Computer Science
Stanford University
bayfc@stanford.edu

David Wendt
Department of Computer Science
Stanford University
dwendt@stanford.edu

Abstract

We develop effective general-purpose sentence embeddings for a variety of downstream tasks. We augment a minBERT model, pretrained on masked-word prediction over a large corpus of text, with task specific heads to perform sentiment analysis, paraphrase detection, and similarity classification. In order to achieve good results on these datasets, we employ rigorous hyperparameter tuning, explore regularization techniques to avoid forgetting key knowledge in the base model, and augment our datasets with multiple negatives and with synthetic data to extend limited finetuning datasets. We find evidence supporting the effectiveness of multitask fine-tuning, with additional multiple negatives pretraining on paraphrase yielding improved performance on semantic textual similarity, and general-purpose sentence embeddings requiring only shallow task-specific classification layers for good performance on all three downstream tasks.

1 Key Information to include

- Mentor: Sonia Chu
- Team contributions: Bay primarily worked on completing the multitask learning implementation, the hyperparameter sweep, and the synthetic data extension, while David primarily worked on the base BERT implementation completion, the regularization methods, and the multiple negatives extension.
- External Collaborators: None
- Sharing project: No

2 Introduction

The main scientific question of our project is how to create general representations of sentences, which can then be used for a variety of downstream tasks, with limited further specialization. This is a very important subject because training the best LLMs is extremely expensive, and so it is important that we maximize the value of the depth of knowledge and linguistic understanding that these models encode. We know that these models must have a deep understanding of language in order to generate it; so it stands to reason, and has been shown experimentally, that we can use this understanding of language via sentence embeddings for other NLP tasks.

Our task is to do this as effectively as possible, by making the best modifications we can under a limited budget. In particular, we are interested in focusing on whether augmenting the optimizer to help maintain mathematical properties like smoothness can help generate better embeddings. We are also interested in determining how these improvements can assist in transferring pretrained knowledge to a variety of downstream tasks. We also explore whether augmenting data with multiple negatives or GPT-labelled synthetic data can assist in learning across multiple tasks.

3 Related Work

The Bidirectional Encoder Representations from Transformers model, or BERT, was introduced by Devlin et al. (2018). This transformer model was state-of-the-art at the time on several tasks including GLUE, MultiNLI and SQuAD. The sentence encodings generated by this transformer can be used for several downstream tasks including text classification, as discussed in Sun et al. (2019). They examine the prior multi-task learning of Liu et al. (2019), which provides benefits of learning across tasks as well as the regularization effect of needing to learn generalized representations helpful for several downstream tasks, but improve on the method by replacing mult-task learning of the full model with mult-task BERT fine-tuning. Sun et al. (2019) find that this helps with efficiency by not needing to train tasks from scratch at each training run and avoids the necessity of carefully weighing task-specific objectives, and rather makes better use of the shared, pretrained model.

Additional finetuning techniques in the literature include *Smoothness-inducing Adversarial Regularization* (SAIR) to enforce smoothness of the fine-tuned model and encourage robustness, as well as *Bregman Proximal Point Optimization* (BPPO) to "effectively prevent aggressive updating and stabilize the fine-tuning process," both introduced by Jiang et al. (2019); and *Multiple Negatives (MN) Ranking Loss Learning*, discussed by Henderson et al. (2017). SAIR contributes to robustness of the trained model by adding an adversarial regularization term which punishes the model during training for changing too abruptly around datapoints seen. BPPO encourages stability during fine-tuning by regularizing using distance in the output space (i.e. model predictions on a downstream task like text classification) between a model and its previous iteration, rather than only regularizing in parameter space, since the map from the latter to the former may not be very smooth everywhere.

MN helps with efficiency of training and quality of embeddings by adding additional loss terms corresponding to negative sentence pairs; in a given batch of sentence pairs which are each a positive example, each sentence can be grouped into a pair with all other sentences besides its true pair, allowing for many more negative examples to be generated. This also allows each sentence in the dataset to be used more than once per epoch, contributing to data efficiency and effectively augmenting the dataset.

4 Approach

Our approach consists of implementing a pretrained minBERT model with three classification heads, which operate on the pooled output of the base model on an input sentence. We train the model to perform sentiment analysis, paraphrase detection, and similarity classification simultaneously, as we hypothesize these tasks share common structure, and can benefit from exposure to more data. Our minBERT model is then augmented with a set of extensions described below.

4.1 Baseline

Our baseline model is the default minBERT implementation with a simple prediction head architecture, as described in the Default Final Project Handout and scaffolded in the starter code, and with a single linear layer in each task-specific prediction head.

4.2 Regularization methods

Our two advanced regularization methods are BPPO and SAIR, as developed in Jiang et al. (2019), which we have implemented ourselves based on the following mathematical descriptions from the cited paper:

SAIR adds a regularization term $\lambda_s \mathcal{R}_s(\theta)$ to the objective, where λ_s is a tuned regularization parameter and $\mathcal{R}_s(\theta)$ is the "smoothness-inducing adversarial regularizer," defined as

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\tilde{x}:\|\tilde{x}-x_i\|_p<\epsilon} \ell_s(f(\tilde{x}|\theta), f(x_i|\theta))$$

where $\epsilon > 0$ is another tuned parameter, p is a norm order to be chosen, and ℓ_s is a loss measure on predictor outputs, either a symmetrized KL-divergence $\ell_s(P, Q) = \mathcal{D}_{KL}(P\|Q) + \mathcal{D}_{KL}(Q\|P)$ for classification or a squared L_2 distance for regression. Intuitively, $\mathcal{R}_s$ measures the (lack of) local

smoothness of the fine-tuned predictor $f(\cdot|\theta)$, penalizing rapid change of f within a p-norm ϵ-ball around each x_i. Thus, including this term helps keep the model smooth and improve robustness.

The second contribution, BPPO, modifies the objective by adding a second regularization term:

$$\min_{\theta}. \mathcal{F}(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \theta_t) \qquad \text{where} \qquad \mathcal{D}_{\text{Breg}}(\theta, \theta_t) = \frac{1}{n} \sum_{i=1}^{n} \ell_s(f(x_i|\theta), f(x_i|\theta_t))$$

Here, μ is a tuned regularization strength, and θ_t is the value of θ in the previous iteration of the optimzation algorithm. This regularization reduces how much the predictor changes iteration-to-iteration during optimization, similarly to PPO or trust-region methods in RL.

In both of these cases, i indexes the datapoint x_i in the dataset with n datapoints. In practice, since we use minibatch stochastic gradient descent rather than gradient descent with a single averaged gradient step per epoch, the natural modification of these algorithms is to average over a minibatch rather than over the full dataset. This is how we have implemented each of these algorithms.

For SAIR, we made an additional approximation: we sample randomly rather than exactly computing the above maximization. As written, the maximization is over all possible $\tilde{x}$ within a p-norm ϵ-ball around each x_i in the dataset (or in the minibatch). Solving a separate high-dimensional (given sentence embeddings of $d = 768$ dimensions) optimization problem for each datapoint could become very inefficient. Instead, we sample a fixed number of points on the p-norm d-dimensional ϵ-ball, and then use these same randomly chosen steps for each x_i in the minibatch, and take the maximum over these sampled points, as an estimate of the true maximum.

Moreover, we only regularize the prediction heads with SAIR, not the entire transformer model. Since our sampling approach requires one forward pass through the model per ϵ-step per datapoint, and we used between 100 and 1000 random ϵ-steps per datapoint to estimate the maximum, it would have been infeasible both in terms of time and memory to pass each step through the full BERT model in addition to the prediction head. As-is, we ran into out-of-memory errors when trying to increase the number of samples to 5000.

Moreover, note that Bregman PPO requires keeping around the previous iteration of the model, which can be quite memory-expensive for a full transformer model like BERT. Moreover, since we update the model parameters each minibatch rather than once per dataset pass, we must copy all of these model parameters many times per epoch, which is also time-intensive. An alternative approach is to update the "previous model" only once per epoch, but this would correspond to a regularization term involving a model from many gradient updates ago toward the end of each epoch, which might provide too strong a regularization, especially since the model parameters may have changed significantly since then. While in theory Bregman PPO is helpful for stabilization of learning, these drawbacks in the minibatch SGD case render it a less than optimal approach in terms of time and memory efficiency.

4.3 Multiple Negatives

Another extension we implemented is augmenting our data with multiple negatives. We first filtered the Quora paraphrase dataset (see §5.1) to obtain only the positive examples (i.e. sentences with label 1 for being paraphrases of each other) so that all sentence pairs remaining are a true pair and should have similar embeddings. Then, for each minibatch $\{(a_i, b_i)\}_{i=1}^{m}$ (where a_i and b_i are labelled as paraphrases of each other for each i), we consider all alternative pairs (a_i, b_j) for $i \neq j$ as negative examples, i.e. sentences which are not paraphrases of each other. (This assumption could technically be broken if there are multiple datapoints in the dataset where the paraphrase pairs are also paraphrases of each other, but this is extraordinarily unlikely in any given minibatch.) Then, we implemented a loss function (similar to that of Henderson et al. (2017)) which encourages positive pairs to have similar embeddings but also encourages negative pairs to have different embeddings: if $S(a, b)$ is the similarity score predicted by our model between sentence a and sentence b, for a given minibatch of size m we add a loss term

$$\mathcal{L}_{\text{MN}} = -\frac{1}{m} \sum_{i=1}^{m} \left[S(a_i, b_i) - \log \sum_{j=1}^{m} \exp S(a_i, b_j) \right]$$

Of course, in our dataset, there are already negative examples; thus, this extension would not be quite as effective as if our dataset were only positive sentence pairs and did not include any pairs of false paraphrases. However, our artifical negative examples are different from the preexisting negative examples in an important way: the Quora dataset is a dataset of questions asked on Quora, where each sentence pair is a potentially similar pair of questions. In particular, even when the questions are not actual paraphrases, they tend to involve common words, or be nearly identical except for a few words which make the meaning different. They do not tend to include entirely different sentences. Our multiple negatives augmentation adds negative examples of unrelated sentences, encouraging sentence embeddings to spread out more overall.

Regardless, we implemented this as an additional pretraining step before the additional downstream multitask fine-tuning, which includes fine-tuning on the Quora dataset with its negative examples included.

4.4 Synthetic Data

In order to explore the potential of synthetic data to address tasks with limited data we focused on seeing if GPT4o could be used to boost performance on the sentiment classification task. Both sentiment classification and textual similarity had much less data than the paraphrase task, so those were our two candidate tasks for generating synthetic data, but we chose sentiment classification due to its simplicity and single sentence focus. We used 10 examples from the SST dataset to prompt GPT-4o to generate numeric ratings for 10000 sentences extracted from the Good Reads dataset Malcolm (2024).

5 Experiments

5.1 Data

We use the Stanford Sentiment Treebank (SST) and CFIMDB for sentiment classification, the Quora dataset for paraphrase detection, and the SemEval STS Benchmark for Semantic Textual Similarity. These datasets all focus on sentence-level classification tasks, although STS and Quora classify sentence pairs, while SST and CFIMDB focus on single sentence classification.

During our training recipe, we shuffle all datasets together, randomly selecting batches from each dataset. This resulted in a strong bias towards training on paraphrase, which is by far the largest dataset, so we modified the algorithm so that we would equally see data from all three tasks.

5.2 Evaluation method

We are interested in understanding how well our contextualized embeddings will perform on downstream tasks. We focus on three particular downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. For sentiment analysis, our dataset has integer labels from 0 to 4, so we currently treat this as a classification problem and use cross-entropy loss. For paraphrase detection, we have binary labels and can frame this as a classification problem, using binary cross-entropy loss. For these two datasets, we use validation set accuracy as our evaluation metrics. For semantic textual similarity, which has a similarity score between 0 and 5 and which we treat as a regression problem, we will use the Pearson correlation of true values against predicted values for similarity as our evaluation metric.

5.3 Experimental Details and Results

5.3.1 Learning Rate

Once we had a stable working model, and were achieving reasonably strong initials scores on all three tasks, we began the process of rigorously sweeping hyperparameters to improve performance. We started by sweeping the learning rate of the model, as this is perhaps the most important hyperparameter. Because viable hyperparemters can cover a large range we swept logarithmically spaced values of lr between 1e-7 and 1e-2.

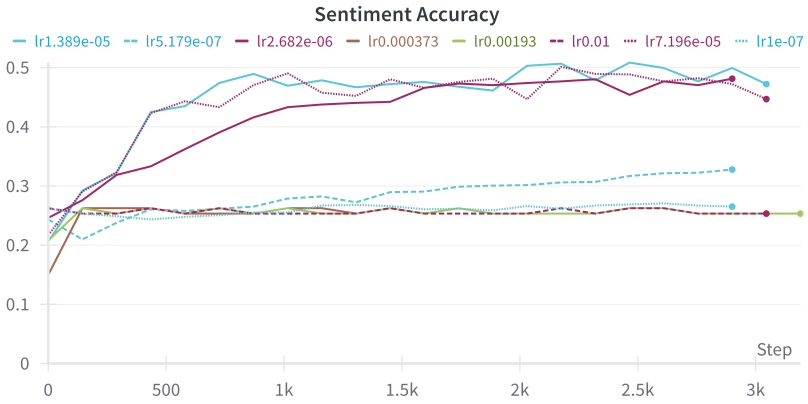


Figure 1: Sentiment Accuracy Learning Rate Sweep. A legend label "lrX" refers to a learning rate of X.

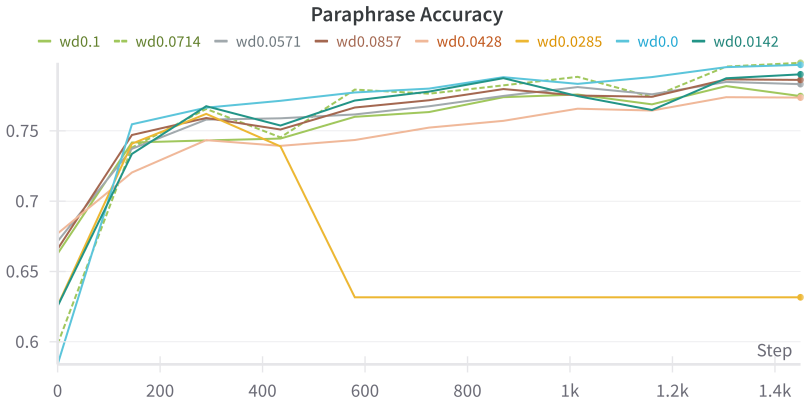


Figure 2: Paraphrase Accuracy Weight Decay Sweep. A legend label "wdX" refers to a weight decay strength of X.

We found similar results for overall quality and convergence across all three tasks, and so report one plot (Fig. 1) for brevity. The learning rate sweep showed that rates around 1e-5 converged quickly and yielded good performance. We conducted a second linear sweep in this range, and found no significant variation, and chose 7e-5 as a learning rate.

5.3.2 Weight Decay

Early experiments clearly showed significant overfitting, in which we achieved far higher test set accuracy than we were able to achieve on the dev set. We thus swept L_2 regularization weights to see if this technique could help with this issue. We similarly found that the results of applying weight decay were quite uniform across our three tasks, adding to our confidence in a multitask approach. We found that zero weight decay performed best (Fig. 2), and so moved on to other approaches to improve performance.

5.3.3 Dropout

Continuing with our efforts to improve performance through regularization, we swept the dropout parameter to understand how this affected performance. At convergence, we found limited differences between dropout parameters within a reasonable range across all tasks (Fig. 3), and chose dropout of 0.3 for further experiments.

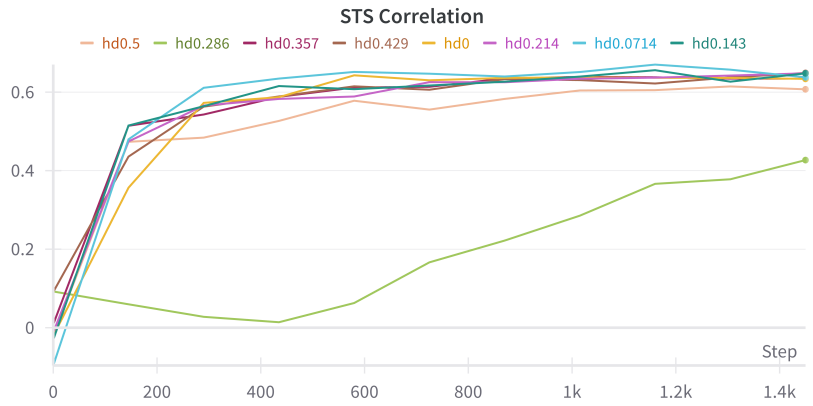


Figure 3: STS Correlation Dropout Sweep. A legend label "hdX" refers to a hidden dropout probability of X.

5.3.4 Paraphrase Multiple Negative Pretraining

Because the BERT model was not directly pretrained to create general semantic embeddings, we hypothesized that performing an embedding specific pretraining step before multitask finetuning could result in performance gains. Because the paraphrase task essentially amounts to recognizing similar and dissimilar sentences, we thought it had high potential to benefit other tasks. We compounded this by applying multiple negatives, a technique which allows us to further push apart the embeddings for unrelated sentences.

We found strong results for this approach. Pretraining on one or more epochs of multiple negative paraphrase data improved downstream paraphrase performance. More interestingly, we saw that this paraphrase MN pretraining led to large boosts in STS Correlation performance (Fig. 5.3.5). The best performance was achieved after two pretraining epochs. We found that sentiment accuracy was relatively unaffected by this pretraining process, compared to paraphrase and STS.

5.3.5 Synthetic Data

We trained on synthetic sentiment data in two ways. First, we tried training entirely on these data points for the first 5 epochs. Then we tried training on the combined synthetic and real data points for the first five epochs. In both cases, we continued training with real data for the remainder of the run.

We found that training with synthetic data alone could lead to relatively strong performance (0.417 sentiment accuracy, compared with around 0.5 max achieved) on the task (Fig. 4). This was somewhat surprising, as we had assumed that training on automatically labeled book review sentiments would lead to the model learning a significantly different distribution, which might help as a pretraining step, but which would not directly learn the task.

We found that training with synthetic data in our general experimental setup with paraphrase MN pretraining yielded only marginal benefits. However, pretraining with synthetic labels without this paraphrase pretraining step led to our highest evaled sentiment score of 0.527 on dev, at the cost of dramatically harming other tasks.

5.3.6 Regularization

Due to the requirement to copy the entire transformer model at each gradient update in our default implementation of BPPO, the training ran approximately 20 times slower and thus a full training run with BPPO was infeasible. We experimented with updating the previous model only once per batch instead of once per epoch, but this did not result in any improvement. We conclude that this is because of the changes to BPPO we made as a result of using minibatch SGD rather than full-dataset gradient descent, as discussed in §4.2

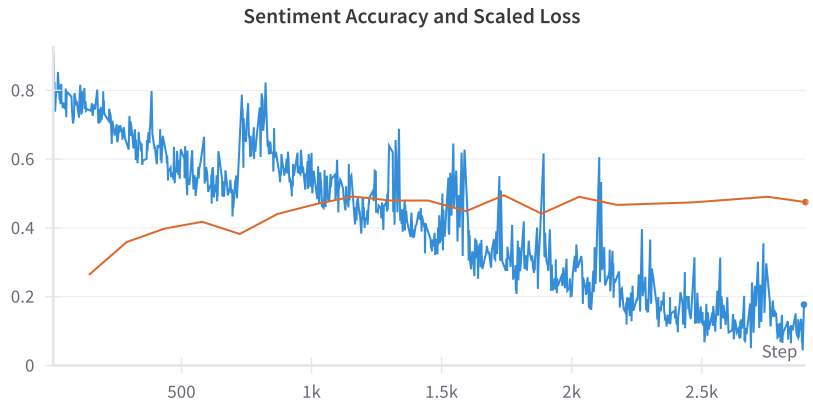
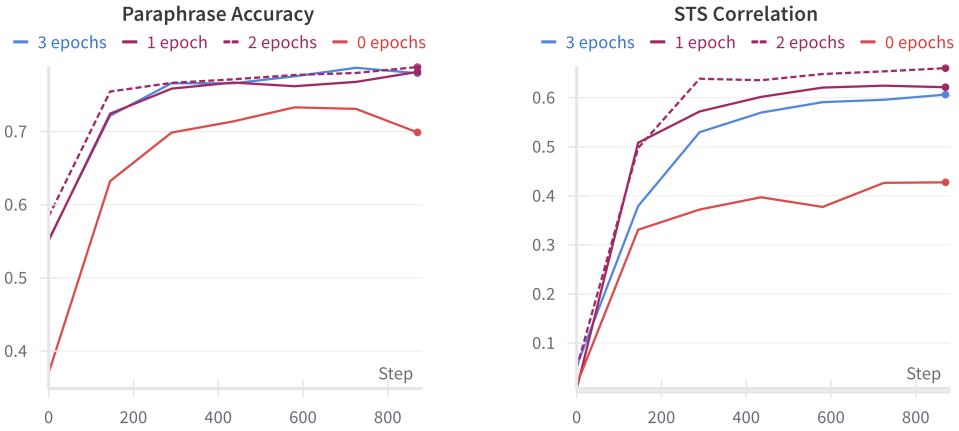


Figure 4: Sentiment Accuracy (Red) and Sentiment Loss (Blue) during training with synthetic data. The loss spike around 700 steps corresponds to the switch from synthetic to real sentiment data.



Our SAIR experiments involved a sweep across regularization strength (0.1, 1.0, 10.0), number of samples per datapoint in the maximization estimation (100, 1000), and size, ϵ, of the sphere on which to sample (1e-5, 1e-4). Regardless of our choices of these hyperparameters, we found no significant difference from non-SAIR experiments. We conclude that SAIR is not very effective in experiments with such shallow prediction heads, since we only regularize the heads rather than the entire BERT model.

6 Analysis

We perform qualitative examinations below of our model's dev set predictions in order to see the likely failure modes for our system.

For the sentiment predictions, we find that in the significant majority of incorrect cases, we were only off by a single bucket. This is to be expected, given that dividing sentiments into 5 buckets is inherently ambiguous. A human could not perform this task at high accuracy, so it is unlikely an ML will be able to achieve significantly accuracy than this. There are also noisy sentences, for instance "Nothing is black and white" for which the sentiment is quite ambiguous. Treating this as a regression problem would probably be a more well-defined problem and the eval metric would be more representative of model performance.

We find that for paraphrase detection, a relatively straightforward binary task, we get most of the answers correct. However, our model does seem to struggle with more complex examples.

For instance, “Why are vapor absorption refrigeration systems not used for domestic/small scale purposes?" and “Why is Vapor absorption cycle not used to run a domestic refrigerator?" should be marked as paraphrases, but are not by our model. This could be because these questions use relatively obscure technical terms, that are not as well know by our model.

We find that for STS, upon manual inspection of the results that the correlation we measure appears to hold up in general, but there are some cases where our results are quite off, that seem to be somewhat straightforward. For instance, we score the similarity between “The couple danced in the church” and “A couple slow dances” as just 0.78, even though they have high similarity (true label 3.2). This could perhaps be because “Church" is understood to be a relatively important word, that has a generally strong influence on meaning, but overall seems like a failure of the model. Our model also mis-scores a sentence with a typo, indicating that unknown words also harm model performance.

7 Conclusion

This project demonstrated that multitask learning is an effective paradigm for creating general purpose sentence embeddings. Our multitask classifier needs only shallow (1-2 layer) prediction heads on top of the BERT model used for generating the sentence embeddings in order to yield good performance on each of the three downstream tasks. If these embeddings were not sufficiently general-purpose, we would have needed deeper prediction heads to perform more post-processing on the embeddings. However, we found that the regularization effect gained from shallower prediction heads outperformed our architecture experiments with deeper prediction heads, demonstrating that the embeddings were already sufficient for good performance across tasks. Our final test set performance had an SST accuracy of 0.499, a paraphrase accuracy of 0.789, and an STS correlation of 0.606. For comparison, our baseline pretrained minBERT model, with 20 epochs of fine-tuning only on the last linear layers and without any extensions or hyperparameter tuning, yields metrics of 0.308 (SST accuracy), 0.661 (paraphrase accuracy), and 0.118 (STS correlation).

One important result supporting this conclusion is the evidence for a strong link between paraphrase and textual similarity performance. Across experiments involving architecture, hyperparameter, and training recipe choices, we observe that paraphrase and textual similarity performance seem to either go up or down together. Conversely, paraphrase and semantic tasks sometimes see an inverse relationship, with our optimizations for the latter decreasing the performance of the former or vice versa, though this did not happen all the time. For instance, MN pretraining on paraphrase data significantly increased STS correlation.

7.1 Future work

For future work, we would like to study in detail the interplays between various tasks. For instance, we would be interested in trying to train various combinations of tasks together and separately in order to precisely understand when we can benefit from including multiple tasks, and when this impedes learning. We would also be interested in exploring whether task specific learning rates, or any other methods, could help us handle training on tasks with widely varying amounts of data.

With a more significant compute budget, we would try SAIR on the full transformer and with an optimizer to solve each maximization problem, to see whether the exact SAIR algorithm (as opposed to our efficiency-based approximations) would be an effective regularizer for our minBERT model.

8 Ethics Statement

Our project has a broad goal of generating contextual embeddings which are useful for a wide range of tasks, including tasks that might not be immediately forseen. Thus, we are working to advance general capabilities that could potentially be misused or cause harm. One concern is that a classification system could be deployed in the real world without proper care to consider the bias it might possess. Our embeddings are likely to contain societal biases present in the training data, and so should not be used to make sensitive decisions in the real world. We could reduce the likelihood of this happening by not releasing weights except to trusted partners. We could also perform additional fairness training by augmenting our data with random replacements of races, gender identities, etc. with other races or gender identites so that the training data is as close as possible to invariant under

any swap of such identites, and therefore the resulting embeddings would have to be much more blind to potential biases.

Another concern is that the pretrained sentence embedding model could be used for classification or regression tasks similar to, but distinct from, sentiment classification, such as estimation of political bias or effectiveness in spreading misinformation, and thus be used in a pipeline for intentional spreading of political misinformation. While model development cannot fully predict how others will fine-tune a pretrained model to their specific tasks, our use case is for fairly simple sentence analysis, and it is unlikely that more complex and nuanced characteristics such as political leaning could be determined from our embeddings without significant further training. To counteract this, we could examine if there is any correlation between, for instance, political bias and the sentiment classifications from our model. We can then examine whether this correlation is present in the labelled data itself, and if so, adjust labels up (for one group) and down (for the other group) until the average sentiment is the same across groups.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Malcolm. 2024. goodbooks-10k-extended: Extended version of the 2017 dataset, with additional fields. `https://github.com/malcolmosh/goodbooks-10k-extended`. Accessed: 2024-06-07.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.