BERT Fine-Tuning with Contrastive Loss and Smoothness-Inducing Regularization

Stanford CS224N Default Project

Laura Wu
Department of Computer Science
Stanford University
laurawjr@stanford.edu

Frank Zhao
Department of Computer Science
Stanford University
frankz24@stanford.edu

Abstract

Semantically rich sentence embeddings can prove extremely useful for NLP tasks. Thus, The goal of this project is to develop a baseline BERT model and fine-tune it for three downstream tasks simultaneously. We implement three techniques – cosine similarity finetuning, contrastive loss for additional pretraining, and smoothness-inducing regularization – to improve upon the pre-trained BERT embeddings. Results have shown that all three techniques boost overall model performance.

1 Introduction

The classification of sentences is a critical research area in Natural Language Processing (NLP), and tasks such as sentiment analysis and paraphrase detection are the bedrocks to many practical applications including social media analysis, question answering, and recommendation systems. The development of effective approaches for these tasks has the potential to not only contribute to engines that make our lives more convenient but also enhance our understanding of human language comprehension and variation.

To convert sentences into machine-recognizable inputs, we typically use vector embeddings in a high-dimensional space to capture their semantic and syntactic features. The overarching goal of this project is therefore to develop the architecture to produce robust and generalizable embeddings for English sentences that can subsequently be used to perform multiple sentence-level NLP tasks. Specifically, we will first implement the most important components of the original Bidirectional Encoder Representations from Transformers(BERT) model by Vaswani et al. (2017) and then add in fine-tuning techniques to better adapt the pre-trained BERT's contextualized embeddings to 3 sentence-level downstream tasks. These include sentiment analysis, which labels phrases of opinions as negative, neutral, or positive, paraphrase detection, which checks whether two sentences are rewordings of each other, and semantic textual similarity, which measures the degree of semantic equivalence between two sentences. The datasets we will be using come from the Stanford Sentiment Treebank Dataset by Socher et al. (2013), the Quora Dataset by Sharma et al. (2019), and the SemEval STS Benchmark Dataset by Agirre et al. (2013).

At a high level, the framework of fine-tuning existing language models for downstream tasks allows many NLP-related problems to be tackled more efficiently with little extra training and re-designing. There has been numerous proposals of novel techniques and extensions related to this goal since the publication of BERT. Some important areas include the incorporation of domain-specific text corpora into pre-trained language models for fine-tuning and strides in multi-task learning, which involves fine-tuning a language model on multiple related tasks simultaneously. Our project explores three such techniques, namely Cosine-Similarity Finetuning, Contrastive Loss, and Smoothness-Inducing Regularization. More specifically, in fine-tuning our BERT base model, we apply cosine-similarity to pairs of BERT embeddings before adding them to the output of a linear layer for the final logit. Also, contrastive loss is implemented for the Quora dataset to achieve additional pre-training. Further, we

implement a smoothness-inducing step in computing loss that introduces slight perturbation into the inputs with the goal of maintaining consistency in the output of the model.

The performance is evaluated for each of the three techniques as well as for the three approaches combined. Results show promising advantage of all three techniques on multitask finetuning.

2 Related Work

As mentioned in the introduction, the bulk of our baseline model BERT is derived from its original paper by Vaswani et al. (2017). Before then, developments in NLP have shown that rich, unsupervised pre-training is an integral part of many language understanding systems, and BERT goes a step further to generalize these findings to deep bidirectional architectures, broadening the range and effectiveness of NLP tasks that pre-trained models can achieve.

While BERT already achieves state-of-the-art performance, there are still limitations to its capabilities. For one, as it is pre-trained only on masked language modeling and next sentence prediction, fine-tuning on other datasets is often-needed for other tasks and domain-specific knowledge. Pan et al. (2019), for example, fine-tuned BERT on the task of open-domain question answering using external knowledge sources such as Wikipedia, realizing significant gains over previous methods. Ostendorff et al. (2019) fine-tuned BERT on the task of document classification by incorporating knowledge graph embeddings, which also achieved state-of-the-art results on several datasets.

Regarding specific techniques involved in fine-tuning, different additional learning objectives have demonstrated notable improvement in model performance on the corresponding tasks. Multiple Negatives Ranking Loss Learning is one that aims to minimize the distance between similar pairs of sentences while simultaneously maximizing the distance between different sentences (Henderson et al., 2017), which has shown promising results. To ensure that fine-tuning does not compromise the model's ability to generalize to unseen data, Jiang et al. (2020) propose a framework to reduce aggressive fine-tuning involving two components, smoothness-inducing regularization and Bregman proximal point optimization. When combined and tested on multiple tasks, this framework does achieve state-of-the-art accomplishments. Additionally, Gao et al. (2021) developed a contrastive learning framework that works on both supervised and unsupervised tasks, which in effect regularizes pre-trained embeddings' anisotropic space to be more uniform and has shown to achieve significant improvements on pre-trained BERT base.

3 Approach

3.1 Baseline Model

We will start by describing the main architecture of our model below, the backbone of which has already been provided in the starter code. At a high level, our BERT model takes in sentence inputs and tokenizes them before feeding them into an embedding layer and then a stack of 12 encoder transformer layers.

Starting with transforming the input sentences into a desired format, our BERT model utilizes WordPieceTokenizer that first breaks sentences into individual words and then into word pieces within its 30,000 token vocabulary, which are lastly converted into input token ids to feed into our model. Each sequence always starts with a special classification token ([CLS]) and padded to max_length (512) with the [PAD] token. Unknown word pieces will be automatically set to the [UNK] token.

Then, given a list of input_ids $w_1, ..., w_k \in \mathbb{N}$ that corresponds to the rows (or columns) of the embedding matrix (self.word_embedding), the embedding layer converts the indices into token embeddings $v_1, ..., v_k \in \mathbb{R}^D$. We also obtain position indices and position embeddings from self.position_ids and self.pos_embedding. In addition, we obtain embeddings for sentence segmentations for differentiating amongst different sentences inputs, but for the purpose of this project we are just using them as placeholders. As any given input is of length 512, these embeddings are learned for each of the 512 positions. So, we generate input embeddings (E) at each index i that is used later in the model as $E_i = E_i^{token} + E_i^{segmentation} + E_i^{position}$. This input embedding is then fed into the encoder transformer layer. Each of the 12 transformer layers consists of multi-headed attention, an additive and normalization layer with a residual connection, a feed-forward layer, and a final

additive and normalization layer with a residual connection. The multi-headed attention is computed by concatenating h single-headed attentions performed on matrix of queries(dimension d_k) Q, matrix of keys(dimension d_k) K, matrix of values(dimension d_v) V projected onto different representation subspaces at different positions. Mathematically, the single-headed attention score and multi-headed are respectfully calculated as

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$
$$MultiHead(Q, K, V) = Concat(h_1, ..., h_h)W^O,$$

where $h_i = Attention(QW_i^Q, KW_i^K, VW_i^V), W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. Next up, we have an add_norm layer that takes in the input and output of the previous layer and produces the following output: $Layer\_Normalization(input + Dropout(Dense\_Layer(output)))$. Afterwards, we apply a feed forward layer, which first transforms the input with a linear dense layer, followed by a GELU activation function and another add_norm layer. The final output consist of last_hidden_state, the contextualized embedding for each word piece of the sentence and a pooler_output for the embedding of the [CLS] token.

Now that we have a working BERT model, we can load the pretrained model weights and attach a simple feedforward head to its sentence embedding (pooled output) for a range of downstream tasks. To train our model, we also develop the code for a more efficient version of Adams Optimizer, which performs bias correction on the learning rate α and uses two momentum estimates m, v at each time step t to update the parameter θ according to the following equations: $\alpha_t = \alpha \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}, \theta_t = \theta_{t-1} - \alpha_t \frac{m_t}{v_t + \epsilon}$, where $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ with g_t being the gradient. Our implementation also uses decoupling L2 regularization at the end.

3.2 Fine-Tuning

To further improve our BERT embeddings for multitask classification, we implement three specific techniques to finetune our BERT model in the training step – Cosine-Similarity Finetuning, Contrastive Loss, and Smoothness-Inducing Regularization.

3.2.1 Cosine-Similarity Finetuning

To utilize the BERT sentence embeddings for downstream tasks, we need to design a feedforward projection head that transforms the BERT output into the desired prediction output. In the context of sentiment analysis, we can simply attach a dropout layer followed by a linear projection layer to the BERT architecture. This would allow us to simply output logits in the dimension of the number of classes / sentiments. In the context of paraphrase detection and semantic textual similarity, the input consists of two sentences and hence we are dealing with two BERT embeddings. One naive and straightforward approach is simply to concatenate the two embeddings and add a dropout and linear projection at the end. However, this linear function applied to the two sentence embeddings may not be able to extract all the semantic similarities and differences. To improve upon this simple method, we use cosine similarity finetuning.

Cosine similarity of two vectors X, Y is simply defined as:

$$CosineSimilarity(X, Y) = \frac{X \cdot Y}{\|X\| \|Y\|}.$$

It measures the distance between two vectors in an inner-product space. If two vectors are pointing roughly at the same direction, their cosine similarity would be close to 1. In the context of paraphrase detection and semantic textural similarity, we are interested in how similar two sentences are. Thus, if BERT embeddings are semantically-rich, similar sentences should have similar embeddings. Then, cosine similarity would have high predictive power for detecting whether two sentences are paraphrases of each other and whether two sentences have similar meaning. Previous researches like Reimers and Gurevych (2019) have also confirmed this intuition.

To use cosine similarity of the two embeddings in our framework, there are a few options. We can replace the linear layer with cosine similarity; or we can concatenate the two embeddings with cosine

similarity and feed that into the linear layer; or we can add cosine similarity to the linear layer output as the final output. Simply testing for all of these options show that adding cosine similarity with the linear layer output performs best. Thus, we present our final model design as the one below:

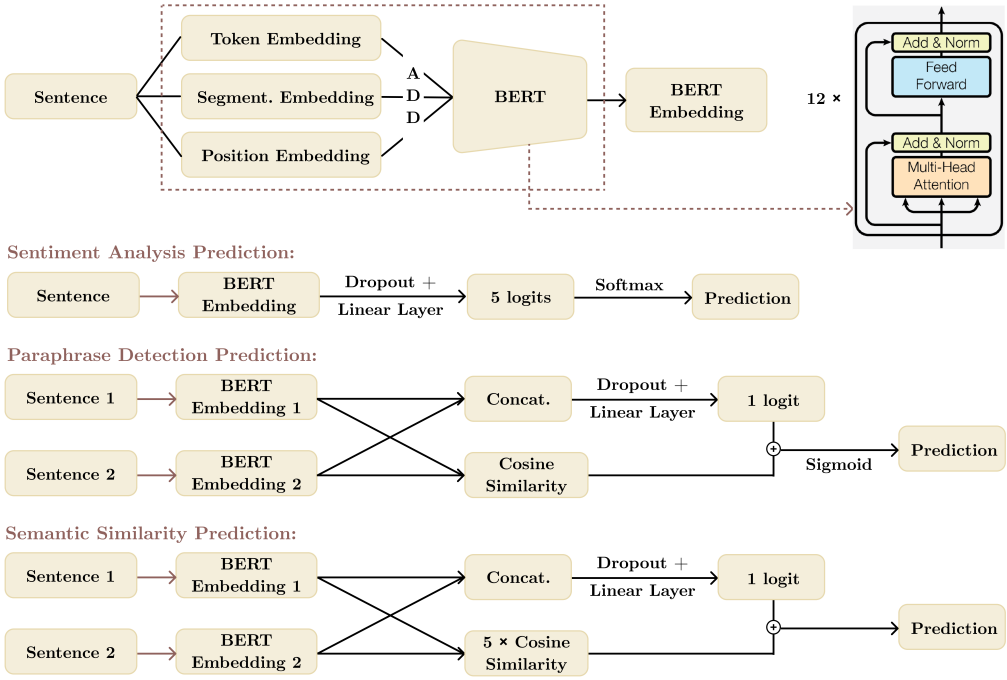

Figure 1: *Model design for the three downstream tasks.*

Note that we add a multiplication factor of 5 to the cosine similarity for semantic similarity prediction because the specific dataset we use have targets in the range of $[0, 5]$.

3.2.2 Contrastive Loss

To take full advantage of using cosine similarity for our downstream tasks, we want to find a way to minimize the distance of similar sentences' embeddings and maximize the distance of different sentences' embeddings. We can achieve this with contrastive loss, proposed by Hadsell et al. (2006).

Formally, denote a batch of sentence pair inputs and labels as $\{(x_{1i}, x_{2i}, y_i)\}_{i=1}^{n}$, where x_{1i}, x_{2i} are a pair of sentences, y_i is 1 if the two sentences have similar meaning and 0 if the two sentences have different meaning, and n is the batch size. Let e_{1i} and e_{2i} be the BERT embeddings of x_{1i} and x_{2i}. Let distance $d_i = 1 - CosineSimilarity(e_{1i}, e_{2i})$. And the contrastive loss is defined as:

$$l_i(x_{1i}, x_{2i}, y_i) = 0.5 \cdot (y_i \cdot d_i^2 + (1 - y_i) \cdot (\max(0, m - d_i))^2),$$

$$L = \frac{\sum_{i=1}^{n} l_i(x_{1i}, x_{2i}, y_i)}{n}.$$

The margin m is a hyperparameter that is usually set as 0.5. The contrastive loss function encourages similar sentences' embeddings to be as close as possible and forces different sentences' embeddings to be at least $m = 0.5$ apart.

We adopt the code from `https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers/losses/ContrastiveLoss.py` to implement this loss function. Using the contrastive loss function, we can provide additional pre-training for BERT using the Quora dataset and further optimize its embedding space.

3.2.3 Smoothness-Inducing Regularization

Often when adopting the pre-trained model for additional downstream tasks, the limitation in data for these tasks might result in overfitting in the fine-tuned models. We therefore introduce a regularization technique to effectively manage the complexity of our model, derived from Jiang et al. (2020). Following the original paper, we refer to this technique as SMART as well. At a high level, the technique relies on Smoothness-inducing Adversarial Regularization, which injects small perturbations to the input at training while maintaining minimal changes to the output of the model, thereby ensuring the overall model smoothness. Specifically, given the model f with parameter θ and n sample data points of the target task $\{(x_i, y_i)\}_{i=1}^n$, the objective function to minimize is as follows

$$min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda \mathcal{R}(\theta),$$

where $\mathcal{L}(\theta) = \frac{1}{n}\Sigma_{i=1}^n \mathcal{L}_1(f(x_i, \theta), y_i)$, and $\mathcal{L}_1$ is the loss function depending on the target task. λ is the tuning parameter for the smoothness-inducing regularizer, and $\mathcal{R}(\theta) = \frac{1}{n}\Sigma_{i=1}^n \mathcal{L}_r(f(\tilde{x_i}, \theta), y_i), f(x_i, \theta), y_i))$, where $\mathcal{L}_r$ is the smoothness-inducing adversarial regularizer, whose functional form again depends on the finetuning task.

The framework of the code is adopted from `https://github.com/archinetai/smart-pytorch`, with significant changes and customized implementations to adapt it for the goals of this project. The outline of our implementation can be seen in Algorithm 1 below, which walks through the training step for fine-tuning on the downstream task. Depending on the particular task, each iteration takes in either one or two input embeddings instead of always one in the provided code, and we simultaneously update model parameters from the whole batch's loss using the appropriate loss functions.

Algorithm 1 SMART Loss Calculation

Notation: $\mathcal{P}$: Prediction function for the similarity between 2 sentence embeddings, $\mathcal{L}_1$: loss function depending on the target task, $\mathcal{L}_2$: loss function chosen for the smoothness-inducing step, L: the total loss at each iteration, $AdamUpdate$: the ADAM update rule for each iteration
Input: T: Number of epochs, I: Number of iterations, χ: training dataset, b_labels: the true label for input embeddings, σ^2: The variance of the random initialization for the perturbations, η: step size for noise gradient, ϵ: normalization parameter, λ: smoothness-inducing loss step size

for $t = 1, .., T$ **do**
  for $x = 1, ..., I$ **do**
    Sample mini-batch $\mathcal{B}$ or mini-batch of input pairs $\mathcal{B}_1, \mathcal{B}_2$ from χ (Assuming 2 in the following parts)
    Obtain embeddings e_1, e_2 for the batch of two inputs
    $L \leftarrow \frac{\mathcal{L}_1(\mathcal{P}(e_1, e_2), b\_labels)}{|\mathcal{B}|}$
    $\epsilon_1, \epsilon_2 \sim N(0, \sigma^2 I)$
    for $i = 1, 2, \ldots, num\_steps$ **do**
      $\tilde{e_1} \leftarrow e_1 + \epsilon_1$
      $\tilde{e_2} \leftarrow e_2 + \epsilon_2$
      $\tilde{l} \leftarrow \mathcal{P}(\tilde{e_1}, \tilde{e_2})$
      $g_1, g_2 \leftarrow \bigtriangledown \mathcal{L}_r(\tilde{l}, b\_labels, \theta_i)$
      $\epsilon_1 \leftarrow \frac{\epsilon_1 + \eta * g_1}{|\epsilon_1 + \eta * g_1| + \epsilon}$
      $\epsilon_2 \leftarrow \frac{\epsilon_2 + \eta * g_2}{|\epsilon_2 + \eta * g_2| + \epsilon}$
      if i = num_steps **then**
        $L \leftarrow L + \lambda * \mathcal{L}_r(\tilde{l}, b\_labels)$
      end if
    end for
    $\theta_{s+1} \leftarrow AdamUpdate_{\mathcal{B}_1, \mathcal{B}_2}(\theta_s)$
  end for
end for

For our purposes, we have chosen $\lambda = 5, \eta = 10^{-3}, \epsilon = 10^{-5}, and \sigma = 10^{-5}$, following the original paper by Jiang et al. (2020). For classification tasks, $\mathcal{L}_1$ is chosen as cross-entropy loss and $\mathcal{L}_r$ is chosen as the symmetric KL-divergence, and for regression tasks, $\mathcal{L}_1$ is chosen as mean-squared error loss and $\mathcal{L}_r$ is chosen as the squared loss.

4 Experiments

4.1 Data

To apply our approach to the overarching task of finetuning BERT embeddings for multiple downstream tasks, we mainly utilize three datasets. These data sets correspond to the three tasks described in the introduction. The first dataset is the Stanford Sentiment Treebank (SST) Dataset by Socher et al. (2013). It consists of 11,855 single sentences from movie reviews, which were parsed into 215,154 phrases. Each phrase is annotated into 5 categories from most negative to most positive. The task is to predict which sentiment category each phrase falls into (sentiment analysis). The second dataset is the Quora Dataset by Sharma et al. (2019). It consists of 400,000 question pairs from Quora with labels indicating whether sentences are paraphrases of one another. We use roughly 50% of all instances. The associated task is to binary-classify two questions as re-wording of each other or not (paraphrase detection). The last dataset is the SemEval STS Benchmark Dataset by Agirre et al. (2013). It consists of 8,628 sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). The task on this dataset is to predict the similarity score between two sentences (semantic textual similarity). We use all three data sets for downstream training and evaluation.

4.2 Evaluation method

The SST dataset and quora dataset are both classification problems. Thus, it is most natural to use accuracy as our evaluation metric. Due to SemEval's original paper by Agirre et al. (2013), we treat the third STS dataset as a regression problem and use Pearson correlation of true similarity values against the predicted similarity values to evaluate model performance.

4.3 Experimental details

We experiment with our three finetuning methods one by one, and each time we evaluate the tweaked model's performance on SST, Quora, and STS. We also include the average of the evaluated results to assess the model's overall performance and robustness. In addition, for the SST dataset, we use cross entropy as our loss function because it is a multi-class classification problem. For the Quora dataset, we use binary cross entropy (BCE) because it is a binary classification problem. And for the STS dataset, we use mean-squared error (MSE) as it is a regression problem.

For all of our experiments below, we use the ADAM optimizer to train the model. The additional feedforward head attached to BERT for each dataset is exactly the same as in Figure 1. Whenever we freeze the BERT parameters and only train the linear layer weights, we use a learning rate of 1e-3. And whenever we are updating BERT, we use a learning rate of 1e-5 . We use a batch size of 32 due to GPU memory constraints and run each experiment for 10 epochs, keeping the model that has the best performance on the dev set.

As a baseline method, we use the pre-trained weights of BERT and freeze BERT parameters during further training. These pre-trained weights are provided to us in the starter code.

To improve upon the baseline method, we first fine-tune BERT's parameters on each dataset alone. For example, we update BERT's model weights when training on the SST dataset and then freeze them when subsequently training on the Quora and STS datasets. This finetunes the BERT contextual embeddings with respect to sentiment analysis. We do the same for the other two datasets. This process also takes advantage of the cosine similarity component for Quora and STS datasets. Further, we can also finetune on all three datasets sequentially, without freezing at each stage.

Secondly, we can apply contrastive loss on the Quora dataset, as it consists of both positive and negative sentence pairs. Specifically, given sentence pair inputs in the Quora dataset, we first calculate their BERT embeddings, and with the paraphrase-or-not label, we calculate the contrastive loss and use that to update BERT parameters. This provides additional pre-training for BERT. Following pretraining with contrastive loss, we can again finetune on all 3 datasets sequentially.

Lastly, we employ the smoothness-inducing regularization technique (SMART) in section 3.2.3 to make our model more robust and less prone to over-fitting. The hyperparameters for SMART are already specified in section 3.2.3. We first apply this technique to finetuning each of the three datasets alone and then sequentially. Note that the SMART framework is only used when updating BERT weights through BERT sentence embeddings but not used when training the linear layer weights.

4.4 Results

The quantitative result from the experiments outlined above is shown in the table below:

Table 1: *Experiment results*

Experiments	SST Acc.	Quora Acc.	STS Corr.	Average
Pretrain (baseline)	0.401	0.679	0.272	0.451
Finetune on SST	0.510	0.681	0.376	0.522
Finetune on Quora	0.398	0.831	0.471	0.567
Finetune on STS	0.427	0.703	0.594	0.575
Finetune on all three	0.471	0.820	0.710	0.667
Pretrain with contrastive loss	0.393	**0.851**	0.630	0.625
Pretrain with contrastive loss then finetune on all three	0.463	0.838	0.775	0.692
Finetune on SST + SMART	**0.521**	0.691	0.456	0.556
Finetune on Quora + SMART	0.415	0.843	0.502	0.587
Finetune on STS + SMART	0.458	0.696	0.681	0.612
Pretrain with contrastive loss then finetune on all 3 + SMART	0.499	0.830	**0.827**	**0.719**
Final result on test set	0.490	0.828	0.814	0.711

By comparing the result from finetuning on one of the datasets with the baseline method, we can see that finetuning not only increases the performance on the finetuned dataset dramatically but also boosts the performance on the other two datasets. The large increase in accuracy and Pearson correlation for the Quora and STS dataset are testimony of the success of using cosine similarity.

By comparing additional pretraining with contrastive loss with naive finetuning on Quora, we can see that contrastive loss increases the model's performance on paraphrase detection even more. Further, it has additional spillover benefits on the STS dataset, as the Pearson correlation also increases by 0.159. Since Quora and STS are both about sentence similarity, this confirms the usefulness of contrastive loss in grouping similar sentences' embeddings and separating different sentences' embeddings.

Finally, by comparing finetuning with and without smoothness-inducing regularization, we can also confirm the effectiveness of SMART. Using this regularization technique not only improves the performance on the finetuned dataset but also generally improves the performance on the other two datasets. This shows that SMART makes our BERT embeddings more robust and more generalizable.

As expected, the best overall performance comes from using all the techniques together. We first provide additional pretraining using contrastive loss on the Quora dataset. We then finetune on SST, Quora, and STS sequentially with smoothness-inducing regularization, and finally train the weights for SST and Quora's linear layers. This combined approach achieves 0.499 accuracy on the SST dev set, 0.830 accuracy on the Quora dev set, and 0.827 Pearson correlation on the STS dev set. The average score is 0.719. This result is reasonably high and demonstrates the validity of our approaches. The performance on the test set is also similar, with the average being only 0.008 less than the dev set average. This shows that our model is not overfitting to any of the datasets significantly. Again, this may show that SMART is providing robust regularization to our BERT embeddings.

It may be surprising that the accuracy for SST is always somewhat low. But this is due to the multi-class fine-grained nature of SST dataset. In fact, the state of art performance is still below 0.6.

Overall, the experimental results align with our intuition about the three approaches. They demonstrate that cosine similarity finetuning, additional pretraing with contrastive loss, and adding smoothness-inducing regularization are all useful and powerful techniques for improving BERT's embeddings.

5 Analysis

One concern over simply concatenating the two sentence embeddings together and feeding that into a linear layer in the case of Quora and STS is that concatenation encodes an order into the

sentence inputs, but swapping sentence 1 with sentence 2 should have very little effect on the model performance. Ideally, the linear layer weights would have learned about this symmetry by itself. Indeed, if we swap the two sentences, the reduction in model performance is minimal. The accuracy on Quora only drops by 0.005 and the Pearson correlation on STS stays the same. This shows that our model has learned the symmetric nature of the two input sentences by itself.

Another concern over the model design is whether the linear layer output is useful for Quora and STS. It is possible that cosine similarity itself is powerful enough. To test this, if we simply drop the linear layer output (but keeping the constant term), the accuracy on Quora drops by 0.455 but the Pearson correlation on STS only drops by 0.007. Thus, the linear layer is almost useless for STS but still extremely useful for Quora.

Next, we can look at some success cases and failure cases of our model.

Table 2: *Success cases and failure cases of final model*

	Input sentences		Label	Pred.
SST Success	A deep and meaningful film.		4	4
SST Failure	If Soderberah's Solaris is a failure it is a glorious failure.		4	0
Quora Success	What are the differences between metal, ceramic, semiconductor, polymers, and composites?	Is it possible to make flexible armor plating from Carbon composites and lightweight metal polymer?	0	0.0002
Quora Failure	What is the difference in pronunciation between "role", "roll", "row"?	What is the difference in pronunciation between "role", "roll", and "row"?	1	0.002
STS Success	A small black dog is in the grass.	A black dog is running in the grass.	3.6	3.601
STS Failure	Work into it slowly.	It seems to work.	0	2.769

The success cases are all examples of very straightforward instances, so it is not surprising that the model predicts correctly. The failure cases are more interesting. The sentence "If Soderberah's Solaris is a failure it is a glorious failure" is very subtle because it uses "failure" to juxtapose the glory of the film. The "if" indicates a hypothetical situation that is perhaps very difficult for language models to understand. The model probably simply relied on the strong negative connotation of "failure" and hence predicted a negative review.

The failure case for Quora is more peculiar. It is not clear why adding an "and" made the model predict the two sentences are not paraphrases of each other. One possible explanation is that in most training instances, seeing "and" means that the question is asking about some additional information that makes the pair not paraphrases. The failure case for STS is more understandable. The two phrases are relatively short and don't have much context around them. There is very little information for the model. And perhaps because of the similar structure and diction, the model concluded that both sentences have some similarity in meaning. Note that the difference between "work into it slowly" and "it seems to work" are even subtle for a human to understand at first glance. It is therefore not surprising the model completely fails.

6 Conclusion

In this project, we implemented cosine similarity finetuning, contrastive loss for additional pretraining, and smoothness-inducing regularization for improving BERT's contextualized embeddings. Results have shown that all three techniques improve model performance on three downstream tasks including sentiment analysis, paraphrase detection, and semantic textual similarity. Through using all three approaches together, we were able to increase the overall performance by 60% from our baseline method. Overall, we learned in depth about the structure of BERT, many techniques for improving this powerful model, and the wider research context around the pretrain-then-finetune paradigm. One obvious limitation, however, is that our approach finetunes on the three tasks sequentially. The performance can depend on the order of finetuning, which is a large issue. Thus, one future research area is to perform multitask learning on all three tasks simultaneously by adding the losses and utilizing gradient surgery.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

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. 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.

Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm, and Bela Gipp. 2019. Enriching bert with knowledge graph embeddings for document classification.

Xiaoman Pan, Kai Sun, Dian Yu, Jianshu Chen, Heng Ji, Claire Cardie, and Dong Yu. 2019. Improving question answering with external knowledge. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 27–37, Hong Kong, China. 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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural language understanding with the quora question pairs dataset. *CoRR*, abs/1907.01041.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.