"That was smooth": Exploration of S-BERT with Multiple Negatives Ranking Loss and Smoothness-Inducing Regularization

Stanford CS224N Default Project

Johnny Chang
Department of Computer Science
Stanford University
cjohnny@stanford.edu

Kaushal Alate
Department of Computer Science
Stanford University
kalate@stanford.edu

Kanu Grover
Department of Computer Science
Stanford University
groverk@stanford.edu

Abstract

In this paper, we finetune minBERT for a few key downstream tasks - sentiment classification, paraphrase detection and similarity scoring. We explore and compare the effectiveness of vanilla minBERT when enhanced with various extensions, particularly Sentence-BERT methods, multiple negatives ranking learning loss, and smoothness-inducing regularization. Ablation studies are conducted to experiment with varying model architectures, objective functions, and regularization terms. Our results show that S-BERT outperforms naive benchmarks across the similarity scoring and paraphrase detection tasks. The mean squared error loss function without regularization shows the best empirical results on similarity scoring, whereas optimizing binary cross-entropy loss with regularization yields the highest accuracy for paraphrase detection. L1 loss with regularization yields the highest accuracy for sentiment classification. This exploration both highlights the versatility and effectiveness of minBERT when enhanced with strategic modifications and also offers valuable insights into the optimal model-hyperparameter configurations for addressing various linguistic tasks.

1 Introduction

Advancements in natural language processing in recent years have paved way for the development of sophisticated models capable of deeply comprehending language. At the forefront of this field lies Bidirectional Encoder Representations from Transformers (BERT), a model that leverages Transformers to generate meaningful contextual word embeddings. The development of BERT marked a steep departure from earlier models that required specialized architecture and domain knowledge to achieve similar performance. An advantage of BERT lies in the model's capability to pre-train on vast amounts of unlabeled text data using unsupervised methods like masked language modeling (MLM) and next sentence prediction (NSP). Pre-trained BERT models can then be fine-tuned with additional output layers to create models that perform effectively on a wide range of tasks. In this paper, we explore three such tasks - semantic textual similarity, paraphrase detection, and sentiment classification.

Despite their state-of-the-art results, training a large-scale, end-to-end BERT model requires a tremendous amount of computational overhead. For this reason, we work with minBERT, a minimalist representation of BERT, and aim to fine-tune the model on the aforementioned tasks using various training strategies. In the first part of the project, we completed the implementation of minBERT, which involved designing the transformer component of the model architecture and various optimization objectives. From here, we designed various extensions to the baseline model, drawing inspiration from Sentence-BERT, contrastive learning, and regularized optimization approaches. Sentence-BERT, abbreviated as S-BERT, was employed for the paraphrase and similarity detection tasks, where it demonstrated empirical results that were superior to that of the cosine-similarity benchmark. To achieve such results, we concatenated pair-wise sentence embeddings (produced from minBERT) with their absolute difference, and learned a classification head to learn accurate embeddings. We experimented with multiple-negatives-ranking learning loss, a contrastive loss strategy that aimed to minimize the distance between the embeddings of positively classified sentences while maximizing the distance between negative pairs of sentences. Training this objective function over many epochs led to overfitting, which we combated with a smoothness-inducing adversarial regularization approach that ultimately yielded further performance gains. With these various extensions, we conducted ablation students to determine which combination of model architecture, optimization strategy, and hyper-parameters yielded the best results. In the end, we often observed that simplicity won out.

2 Related Work

Our extensions were motivated from a few key research papers, namely ‘Efficient Natural Language Response for Smart Reply’ (Henderson et al., 2017), ‘Sentence-bert: Sentence embeddings using siamese bert-networks’ (Reimers and Gurevych, 2019), and ‘SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization’ (Jiang et al., 2020). We outline their key findings below and applications to our problem at hand.

2.1 S-BERT

S-BERT was originally proposed by Reimers and Gurevych as a means to efficiently find similar sentences in a corpus of data by leveraging siamese structures. Batches of sentences simultaneously learn meaningful embeddings in a parallel fashion, which are compactly represented by the CLS token of the BERT output. This is followed by a pooling mechanism that concatenates the difference of these embeddings to produce a larger input with length $3n$ (n being the BERT output dimension). This output is finally passed through a classification layer to determine the similarity of the sentences. Notice how the architecture choice here retains the full dimensionality of the model at every step of the process, allowing the model to learn complex dependencies between pairs of sentences. Other, simpler classification methods like cosine similarity condense this information into a single number.

Additionally, the authors propose more complex modifications to the model that yield better performance - namely, the addition of a triplet objective function. This is similar to the contrastive loss we look at in (Henderson et al., 2017) with a slightly different formulation. An anchor node is chosen, and the loss function aims to minimize the distance between the positive example and anchor point, while maximizing the distance between the negative example to the anchor point. Among STS tasks, S-BERT performs 11.7 points better than the state-of-the-art InferSent and 5.5 points better than Universal Sentence Encoder. As the model is fairly generalizable across a range of tasks involving pairs of sentences, we apply the siamese architecture to both similarity and paraphrase detection.

2.2 Multiple Negatives Ranking Learning Loss

Multiple Negatives Ranking Learning Loss (MNRL) is used as an alternative to the contrastive triplet network loss function in S-BERT, both for its interpretability and ease of implementation. MNRL was first introduced in SMART Reply (Henderson et al., 2017) as a general embedding learning technique that maximizes the distance between dissimilar sentences embeddings, and minimizes the distance between similar sentences. It demonstrated stark improvement when compared to models in SMART Reply which were trained using the sigmoid loss alone - the paper cites a 20% reduction in error rate for Precision @ 1. Moreover, the objective function can reasonably be applied to any task involving pairs of sentences, so it’s especially useful for paraphrase detection and similarity scoring.

MNRL takes in two batches of sentences, where, for all indices i, the sentence at the corresponding index i of the first batch is assumed to be similar to index i of the second batch, whereas dissimilarity is assumed for $i \neq j$. MNRL was an especially practical loss function for the task of paraphrase detection, where two sentences were a positive example when they had a label $y = 1$, and vice versa. However, defining a threshold for a "positive pair" becomes more complicated for similarity detection, which is a regression task with continuous labels from [0, 5]. Our experiments validate this claim.

2.3 Smoothness-Inducing Adversarial Regularization

The model encountered overfitting, which prompted us to explore regularization techniques. An interesting choice of paper was "SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization" (Jiang et al., 2020), which describes an approach to reduce overfitting by adding a regularization term to the loss function that encourages the model to become smoother over the input space. Furthermore, the paper discusses a technique known as Bregman proximal point optimization that imposes a penalty at each iteration of training to avoid aggressive updates. The authors mathematically demonstrate how smoothness-inducing regularization helps ensure a small perturbation in the input does not lead to a large change in output.

Integrating this approach into our minBERT implementation is worthwhile to reduce overfitting and improve the performance of our model. This smoothness-inducing regularization can be seamlessly applied to all tasks in the problem statement, particularly when S-BERT and MNRL tend to overfit.

3 Approach

The goal of this project is to implement different complex fine-tuning methods, such as the earlier mentioned S-BERT (Reimers and Gurevych, 2019), Multiple Negatives Ranking Loss Learning (Henderson et al., 2017), and Smoothness-inducing regularization (Jiang et al., 2020), to optimize for tasks of sentiment classification, paraphrase detection, and semantic textual similarity (STS). These methods and algorithms are built on top of the baseline minBERT.

3.1 Baseline minBERT

For the baseline, we implemented the foundational minBERT embedding layer, which combines both token and positional embeddings. To construct the minBERT transformer block, we implemented an encoder block that applies masked self-attention to the inputs, concatenates their output across multiple heads, and finally applies a normalization layer. To assist in training the model, we implemented the AdamW optimizer, which is a first order-gradient-based algorithm based on adaptive estimates of lower-order moments. This optimization approach especially helps different parameters learn at different rates, leading to faster convergence.

Prior to any adding any extensions, in training the baseline sentiment classification model, we extracted the BERT embeddings of the final CLS token and applied a classification head to predict the most likely sentiment category. Additionally, we implemented the baseline architecture for the tasks of similarity and paraphrase detection using the L1 norm of the element-wise vector difference of the two sentence embeddings:

$$L_1(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} |u_i - v_i| \tag{1}$$

where u and v are the sentence embeddings and n is the dimensionality of the sentence embedding.

3.2 S-BERT Extension

As our first extension to minBERT, we use two fine-tuning strategies from S-BERT. Our first strategy was to estimate the true similarity of sentence pairs as the cosine similarity of their embeddings:

$$logit = \frac{u \cdot v}{\|u\| \|v\|} \tag{2}$$

where u and v are the sentence pairs. As this lead to poor performance in practice, we experimented with concatenating the sentence embeddings u and v with their element-wise absolute difference

$|u - v|$. We then passed this new input through a linear layer W_t that produced a logit representing the similarity of the two sentences:

$$logit = W_t(u, v, |u - v|) \tag{3}$$

For each of these architectures – L1 difference, cosine-similarity, and concatenation – we tested various combinations of regularization and loss functions to observed empirical performance.

3.3 Multiple Negative Ranking Loss Extension

In an attempt to replicate the triplet contrastive loss objective function from S-BERT, we implemented the Multiple Negatives Ranking Loss (MNRL) learning as mentioned in Henderson et al. (2017). This loss function aims to minimize the distance between embeddings of similar sentences while maximizing the distance between embeddings of dissimilar sentences. This is done by training the model with the following objective function:

$$J(x, y, \theta) = -\frac{1}{K}\sum_{i=1}^{K} \log P_{\text{approx}}(y_i|x_i) = \frac{1}{K}\sum_{i=1}^{K}\left[S(x_i, y_i) - \log\left(\sum_{j=1}^{K} e^{S(x_i, y_j)}\right)\right] \tag{4}$$

where each (x_i, y_i) is a pair of positive sentences from the training data and each (x_i, y_j) is a negative pair of sentences with the same first sentence x_i. Here, K represents the number of positive pair sentences in one training batch. Because this loss function requires input data to have both binary positive and negative pairs, this optimization approach applied very cleanly to the task of paraphrase detection. However, the same was not true of similarity scoring - this was inherently a regression task, and defining a "similar" pair of sentences required arbitrarily thresholding the similarity scores, which was observed to work poorly in practice.

3.4 Smoothness-Inducing Adversarial Regularization Extension

Training with MNRL often led to overfitting, so smoothness-inducing adversarial regularization (SIAR) was employed to encourage the model to be more robust to small perturbations in the input. A regularization term was added to the loss function to penalize the model if a small change in the input led to a large change in the model's output. The new objective function is:

$$L(\theta) + \lambda_s R_s(\theta) \tag{5}$$

where L is the previous loss function, $R_s(\theta)$ is the regularization term and λ_s is a hyper-parameter.

The regularization term added to the loss function is defined as follows:

$$R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{||\tilde{x_i} - x_i||_p \leq \epsilon} l_s(f(\tilde{x_i}; \theta), f(x_i; \theta)) \tag{6}$$

ϵ here is a tuning parameter. The loss function l_s is mean squared error loss for regression tasks or symmetrized KL-divergence for classification tasks:

$$\text{Symmetrized KL}(P, Q) = \sum_x P(x)\log(\frac{P(x)}{Q(x)}) + \sum_x Q(x)\log(\frac{Q(x)}{P(x)}) \tag{7}$$

The model penalization forces "smoothness" over the input space, reducing its tendency to overfit.

4 Experiments

4.1 Training and Evaluation Datasets

Three datasets were used, along with their respective evaluation methods. They are as follows:

- Stanford Sentiment Treebank (SST) Socher et al. for sentiment analysis: we use a subset of 11,855 phrases labelled as negative / somewhat negative / neutral / somewhat positive / positive. The evaluation metric will be **accuracy** (what percentage of the labels are predicted correctly).
- A Quora Iyer et al. dataset for paraphrase detection: the subset we are using contains 202,152 question pairs with binary labels. The evaluation metric will be **accuracy**.
- SemEval STS Benchmark dataset Agirre et al. (2013) for similarity scoring: 8628 sentence pairs with labels on a scale from 0 (unrelated) through 5 (the same meaning). The evaluation metric will be the **Pearson correlation** of the true similarity against the predicted similarity.

4.2 Experimental details

Our minBERT's embedding layer computes input embeddings as the sum of token, segmentation (placeholders here) and positional embeddings. 12 Encoder-Transformer layers are used. A learning rate of 1e-05 is employed for finetuning, and the model is trained for 10 epochs. For sentiment analysis, the CLS embedding of the phrase is passed through a dropout (p=0.3) layer and then a linear layer.

For smoothness-inducing regularization, ϵ and λ_s were set to 10^{-5} and 3 respectively.

For multiple negatives ranking loss, negative pairs were determined using positive pairs with labels of 1. For each positive pair (p_1, p_2), we created $k-1$ negative pairs using p_1 and all the "second" sentences $p_{2'}$ in every other pair, where k is the size of a training batch.

We ran 14 experiments in total with all combinations of loss functions, architectures, and regularization. Each experiment was ran for one epoch, and the best performing model would then be chosen to train for 5 more epochs for better performance for the final test set. Note that for the loss functions, only the Mean Squared Error Loss (MSEL) could be applied to the baseline L1 architecture for the sentiment analysis task.

4.3 Results

The following table shows the semantic classification accuracy (sst), paraphrase detection accuracy (para), and the Pearson correlation for the semantic similarity score (sts) of our models after training 1 epoch. The models using mean-square loss, binary cross-entropy loss, and multiple negative ranking loss are labeled with MSEL, BCEL, and MNRL, respectively. Models that use smoothness-inducing adversarial regularization are labeled with SIAR and models without regularization are labeled with non-reg.

Loss Function/Regularization	Architectures								
	Baseline with L1			Cosine-Similarity			Concatenation		
	sst	para	sts	sst	para	sts	sst	para	sts
non-reg + MSEL	0.480	0.375	−0.060	0.262	0.375	0.019	0.175	0.609	**0.501**
non-reg + BCEL	/	/	/	0.183	0.557	0.022	0.134	0.822	0.141
non-reg + MNRL	/	/	/	0.137	0.429	0.433	0.201	0.414	0.452
SIAR + MSEL	**0.487**	0.375	−0.217	0.253	0.375	−0.008	0.486	0.623	0.462
SIAR + BCEL	/	/	/	0.174	0.533	0.165	0.170	**0.828**	0.009
SIAR + MNRL	/	/	/	0.144	0.427	0.482	0.208	0.625	0.029

Table 1: Comparison of dev scores from ablation studies after 1 epoch

Test	Sentiment	Paraphrase	Similarity	**Overall**
Models	L1 + SIAR: 0.543	BCEL + SIAR: 0.845	MSEL + non-reg: 0.582	0.727

Table 2: Best test scores for final submission after 5 epochs

From Table 1, we can see that BCEL with concatenation architecture and regularization performed the best for the paraphrase detection task, while MSEL with concatenation architecture without regularization achieved the best for the semantic similarity task. Surprisingly, the baseline model with L1 difference and with regularization implemented performed the best for the semantic classification task.

Further analyzing the L1 results, we can see that the L1 norm baseline showed significantly inferior performance compared to both cosine similarity and concatenation architectures in the context of paraphrasing and semantic similarity. This aligns with what we expected since the L1 distance between two sentence embeddings cannot well capture the closeness in the vector space.

Additionally, we noticed that on average, MNRL outperformed the MSEL, except in the case of semantic similarity detection. We believe that the concatenation approach, which involves learning a weight matrix, retained more information compared to cosine similarity, which merely relies on a dot product. This effectiveness possibly comes from concatenation's ability to preserve the complexity of data over reducing it to a single dimensionality point, which is overly compressive.

We believe that MNRL did not perform as well as expected, partly due to dataset limitations. MNRL required one positive pair and multiple negative pairs, and we created the negative pairs by pairing all positive pairs in a round-robin fashion. However, this means the negative pairs from the training dataset were not used, which could have enhanced model training. Additionally, there was a trend of overfitting since we noticed that as the number of training epochs increased during our experiments, performance deteriorated.

For the regularization, we can see from the table that it was beneficial in the majority of the cases by encouraging smoother decision boundaries when compared to the cases without regularization. This means a minor input variation is less likely to cause significant output changes, reducing the model's tendency to overfit training data and perform worse on the dev set data.

Based on these first epoch results, we further trained the best-performing models of each task for 5 more epochs for the final submission. The results are shown in Table 2. The final performance aligns with our expectation where more epochs, without overtraining leading to overfitting, would lead to better results.

5 Analysis

To understand and improve our best models further, we performed qualitative analysis by looking into the prediction outputs, comparing them with the ground truth dev set, and analyzing the trends of the failure cases. This section breaks down the qualitative analysis based on each task and its failure trends.

5.1 Sentiment Classification

By analyzing the output of the sentiment classification, we noticed that our model seems more prone to false negatives than false positives, possibly underestimating the positivity of sentiments. Processing the output, there's a noticeable difficulty in accurately classifying very positive sentiments (Ex: 4/5), where it only achieved a 50% success rate. For example, "Uses sharp humor and insight into human nature to examine class conflict, adolescent yearning, the roots of friendship and sexual identity." should be receiving a 4/5, but our model predicted a 3/5 instead.

By analyzing the training set for sentiment classification, we got an average score of 2.06/5 where 4,934 entries have a sentiment value less than 3 and 1,288 entries have a sentiment value greater than 3. We believe that our model is failing to predict positive sentiments because the training data is skewed towards negative sentiment. By including more positive sentiment entries, we believe that the model will perform better in all sentiment scenarios.

5.2 Paraphrase Detection

In the paraphrase detection task, our model performed very well, around 0.85, on both the dev and test sets. However, looking into the failure cases from the predicted outputs, one clear common mistake continuously occurs for our model BCEL without regularization. Specifically, the model failed to detect whether two sentences are paraphrases in two scenarios: when the pair of sentences using different Points of View in grammar or different specificity levels.

Examples include: "What are the best ways to improve English?" and "How can I improve my English vocabulary?" use different Point of View and specificity (English vs Vocabulary). "What are the best and profitable ways for saving money?" and "What are your best ways to save money?" use different Points of View.

We believe that one effective way to mitigate this is to provide more positive sentence pairs with varying Points of View and specificity. By providing datasets that represent the nuances in sentences, the model can better learn to distinguish them.

5.3 Semantic Similarity

Looking at the outputs from the semantic textual similarity task, we noticed our model performs well when the sentence pair has a low similarity score between 1 and 3. However, one common pitfall of our model starts to occur when the score is higher than 4, representing a higher similarity of the sentence pair, but our model predicts a score lower than 2. For example, "More than 1,000 inmates escape from Libya's al-Kweifiya prison" and "1000 prisoners escape from Libyan jail" have a high similarity score (4.4/5), but our model predicted a 1.5/5 instead.

Looking at the training set, the average score for similarity is 2.61/5 with 3,063 similarity values under 3 and 2,638 over 3. The training data skew is not the issue, so we believe this could be an issue with the lack of data overall for the semantic similarity training set. The training set of the paraphrase task has 21.5MB of data while the training set of semantic similarity has only 907KB. By providing many more examples, the model might be able to classify the sentence pairs effectively.

6 Conclusion

In this project, we explored the application of Sentence-BERT (S-BERT) architectures, Multiple Negatives Ranking Learning Loss (MNRL), and Smoothness-Inducing Adversarial Regularization (SIAR) on the BERT model for the tasks of sentiment classification, paraphrase detection, and semantic similarity classification.

Key highlights of our implementation include the implementation of the baseline model (Adam optimizer, multi-head attention, and classification head for the prediction tasks), S-BERT model architectures (L1, Cosine-similarity, and Concatenation), Multiple Negative Ranking Loss (MNRL) with pre-processed paraphrase training data and other loss functions such as Binary Cross Entropy Loss (BCEL) and Mean Square Error Loss (MSEL). Last but not least, we implemented Smoothness-Inducing Adversarial Regularization (SIAR) to avoid overfitting on the dev set, further improving the performance in certain model architecture and loss function scenarios.

With these implementations, we ran extensive ablation studies on the different loss functions, model architectures, regularization, and their impact on the tasks. We analyzed the results both qualitatively and quantitatively to understand why certain models perform better for certain tasks and the trends in prediction failure cases. Our results show that the L1 norm with regularization performs better than L1 norm without regularization. We also see that binary cross-entropy loss with regularization showed the best prediction accuracy for paraphrase detection. For semantic similarity classification, MSEL without regularization showed the best performance. In summary, our results show that loss function performance is highly dependent on the task and regularization is helpful in most scenarios to help the model generalize to unseen data, while the concatenation architecture from S-BERT continuously outperforms the cosine-similarity architecture.

However, the primary limitation of our study includes a few assumptions for our experiments. First, we assumed that the prediction scores after 1 epoch would be representative of the models' performance after 5 or 10 more epochs. This assumption was made because of the limitation of computing credits and training time since each of our 14 experiments requires one to two hours of training time per epoch. Future work with enough computing credits and time would be exploring how each of these models performs after more epochs. Second, we assumed that each positive pair from the datasets for paraphrase detection is only a paraphrase of its pair and not a paraphrase of all the other positive pairs. Future exploration would include preparing datasets where each set of training data includes one positive pair sentence and a large number of negative pair sentences.

A Appendix: Contributions

Johnny worked on cosine similarity and multiple negative ranking loss extensions. Kaushal implemented smoothness-inducing adversarial regularization. Kanu wrote code for S-BERT and multiple negative ranking loss. The three of us have also reviewed each other's code and writing as well as training the various models. We thank our mentor, Olivia Lee, for her input and feedback on the proposal and milestone.

References

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

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. Stanford sentiment treebank.