Improving Multitask MinBERT with Regularized Optimization and Contrastive Learning

Stanford CS224N Default Project

Zhengdan Li, Weian Yin
Department of Statistics
Stanford University
zhengdan@stanford.edu, weiany2@stanford.edu

Abstract

The two-stage transfer learning framework, pre-train models on a large amount of out-of-domain data and then fine-tune them on downstream tasks, has been widely used in NLP. We seek to implement the minBERT model to obtain robust sentence embeddings that can generalize across different natural language task. We experimented with different model architectures and discovered that surprisingly a simple design - using a single BERT - turned out to have the best performance. Further, we applied regularized optimization at fine-tuning stage and used contrastive learning aiming to improve the embeddings for all tasks.

1 Key Information to include

- Mentor: Sauren Khosla
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

The difficulty to obtain large amount of labeled data has been an existing issue for applying natural language processing techniques to specific downstream tasks. An effective way to address this issue is to first train a large language model such as BERT on out-of domain tasks, and fine-tune the model to obtain robust embeddings that generalize well to downstream tasks.

BERT is a transformer-based model that utilized deeply bidirectional word representations to generate powerful contextual word representations. In this project, we incorporated a bidirectional transformer language model, known as BERT, for obtaining lower level sentence embeddings shared between tasks, and constructed task-specific layers on the top. After implementing the minBERT model, we will perform sentence-level tasks including sentence classification, sentiment analysis, paraphrase detection, and semantic textual similarity.

To prevent the fine-tuned model from overfitting the training data, we implemented a regularization technique SMART proposed by Jiang et.al. Further, we built an unsupervised contrastive learning framework aiming to improve the embeddings so that various tasks could benefit.

3 Related Work

One important high-level component of our project is multi-task learning based on BERT. Liu et al. (2019) proposed a Multi-Task Deep Neural Network (MT-DNN) to learn representations across various language tasks. MT-DNN benefits from cross-task data and can learn a more general representations that help adapt to new tasks and domains.

Our main extension is based on smoothness-inducing regularization and Bregman proximal point optimization presented (SMART) in Haoming Jiang (2020). The fine-tuning stage in the transfer learning framework suffers from the limited data from the target domain and the high complexity of the pre-trained model, leading to overfitted model that does not generalize well to unseen data. Using a regularized optimization strategy can alleviate this problem.

4 Approach

Before feeding sentences to the BERT encoder, we designed a preprocessing procedure, which we call "Simple BERT". The details will be discussed in section 4.2. The processed sentences is then fed to the Bidirectional Encoder Representations from Transformers (BERT) with multi-head self-attention in the transformer (Devlin et al., 2018). Each of the 12 transformer layers in addition contains a feed-forward layer with an additive and normalization layer, and a residual connection in between (Vaswani et al., 2017).

On the top of BERT, we used multi-task learning to obtain robust sentence embeddings that can generalize across different natural language tasks. Multi-task learning is an effective approach where the representations can take advantage of shared knowledge of various tasks by learning jointly (Liu et al., 2019). This project focuses on three classification tasks: sentiment classification, paraphrase detection, and semantic similarity prediction. For pairwise comparison tasks that involve two inputs (paraphrase detection and semantic similarity prediction), we concatenate the inputs and add a separator between them. Then we feed contextual embeddings obtained from BERT into different feed-forward layers for each task.

To control the model complexity and prevent aggressive updating during fine-tuning, we employed the Smoothness-inducing Adversarial Regularization technique (SMART) proposed by Haoming Jiang (2020). We also used contrastive learning to improve the overall representations (Gao et al., 2022).

The model architecture can be summarized by the following chart:

4.1 Baseline

Sentiment Classification We obtained our predictions based on sentence embeddings generated by the vannila BERT model.

Paraphrase Detection We employed a simplified version of the pairwise text classification structure proposed by Liu et. al Liu et al. (2019). After obtaining the embeddings for the input sentences, we compute their normalized difference and concatenate the results in the form $[\text{embedding}_1; \text{difference}; \text{embedding}_2]$. Then we feed the output to a trainable linear layer to obtain our predictions.

Semantic Similarity Prediction Same as paraphrase detection.

4.2 Simple BERT

For sentiment classification task, we employed the same structure as the baseline model.

For paraphrase detection and semantic similarity prediction tasks that involve two input sentences, we first concatenated each pair of sentences and added a separator between them. The output for each pair of sentences is in the form $[\text{sentence}_1; [\text{SEP}]; \text{sentence}_2]$. We then generate attention masks from the output and feed them to BERT to obtain contextual embeddings. Finally, we add a trainable linear layer on the top of BERT to generate final outputs.

4.3 SMART

Given the model $f(\cdot; \theta)$ and n observations $\{(x_i, y_i\})_{i=1}^n$ from the target task, this technique solves for the following optimization problem:

$$min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) \tag{1}$$

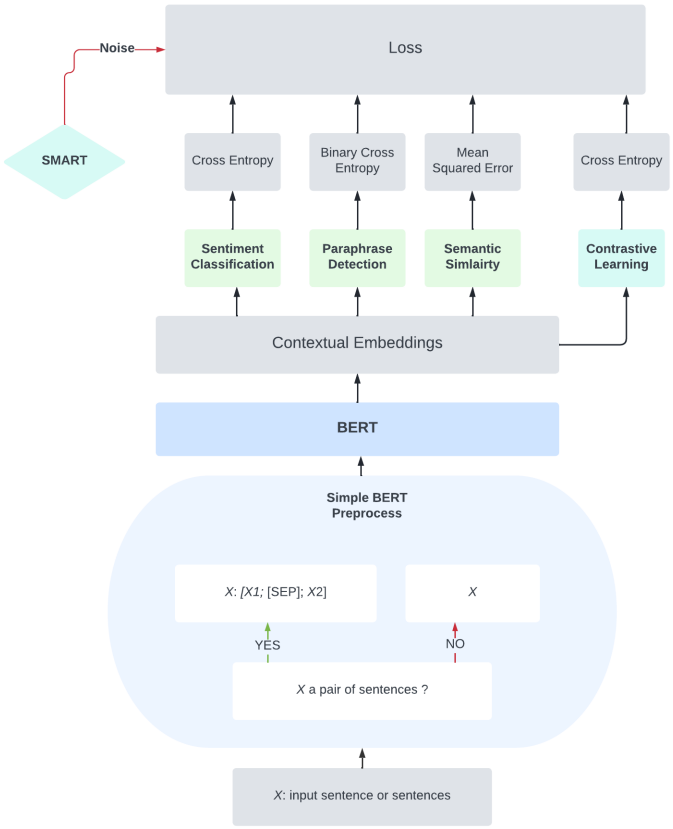

Figure 1: Model Architecture

where $\mathcal{L}(\theta)$ is the loss function:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i;\theta), y_i)$$

l is the loss function depending on the target task, $\lambda_s > 0$ is the tuning parameter, and $\mathcal{R}_s(\theta)$ is the smoothness-inducing adversarial regularizer, defined as:

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

where $\epsilon > 0$ is a tuning parameter.

For classification tasks (sentiment classification and paraphrase detection),

$$l_s(P, Q) = \mathcal{D}_{KL}(P||Q) + \mathcal{D}_{KL}(Q||P)$$

For regression tasks (semantic similarity prediction),

$$l_s(p, q) = (p - q)^2$$

The optimization problem in 1 will be solved using the Proximal point Optimization method, which prevents aggressive updating. $f(\cdot;\theta_0)$ is initialized using the pre-trained model. At the $(t+1)$-th iteration, the update rule is defined as follows:

$$\theta_{t+1} = argmin_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{Breg}(\theta, \theta_t)$$

where $\mu > 0$ is a tuning parameter and $\mathcal{D}_{Breg}(,)$ is the Bregman divergence defined as

$$\mathcal{D}_{Breg}(\theta, \theta_t) = \frac{1}{n} \sum_{i=1}^{n} l_s(f(x_i; \theta), f(x_i; \theta_t))$$

In general, SMART relieves aggressive fine-tuning through introducing a strong penalty on model complexity and injecting noise into the model input.

4.4 Contrastive Learning

Another extension we introduced to the model is a contrastive learning framework using unlabeled data called SimCSE. Contrastive learning can learn useful representations by pulling semantically similar sentences close and pushing part others, and thus produce better sentence embeddings that can benefit other language tasks.
As proposed in Gao et al. (2022), unsupervised SimCSE takes one input sentence and aims to predict itself with standard dropout noise in Transformers used as data augmentation. Taking a collection of sentences $\{x_i\}_{i=1}^{m}$, we denote $h_i^z = f(x_i, z)$ where z is a random mask for dropout. To generate positive pairs, the same input is fed to the encoder twice, returning two embeddings with different dropout masks z, z'. The training objective of unsupervised SimCSE in a size-N mini-batch is as follows:

$$l_i = -log \frac{e^{sim(h_i^{z_i}, h_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{sim(h_i^{z_i}, h_i^{z_j'})/\tau}} \tag{2}$$

where τ is a temperature hyperparameter and $sim(a, b) = \frac{a^T b}{\|a\| \cdot \|b\|}$ is the cosine similarity function. Contrastive learning uses cross entropy as the loss function.

5 Experiments

5.1 Data

- **Stanford Sentiment Treebank (SST) dataset** (Socher et al., 2013)
  This dataset contains 11,855 single sentences from movie reviews, parsed with the Stanford parser and includes 215,154 unique phrases from the parse trees. Each phrase is annotated by 3 human judges, labelled as negative, somewhat negative, neutral, somewhat positive, or positive. We aim to predict the sentiment classification labels of the sentences using BERT embeddings.

- **Quora Dataset**
  This dataset contians 400,000 question pairs with binary labels **No** and **Yes**, indicating whether the questions are paraphrases of each other. We will use this dataset to detect if a pair of sentences are paraphrases.

- **SemEval STS Benchmark Dataset** (Agirre et al., 2013)
  This dataset contains 8,628 different sentence pairs labelled with varying similarity from 0 (unrelated) to 5 (equivalent meaning). We aim to measure the semantic textual similarity of sentence pairs using BERT embeddings.

- **English Wikipedia Dataset** (Gao et al., 2022)
  This dataset contains a million sentences randomly sampled from English Wikipedia and does not have any label. However, we only used 20,000 sentences in the dataset to reduce the training time.
  To setup the labels for the unsupervised learning, one input sentence is encoded twice with different dropout masks as a positive pair. On the other hand, the input sentence and other sentences in the same batch form negative pairs.
  The dataset is used for contrastive learning to produce better general embeddings for other tasks. Thus, we do not evaluate the unsupervised SimCSE task itself.

5.2 Evaluation method

- **Sentiment Classification**
  We used the cross entropy loss function to update model weights and accuracy to evaluate our results.
- **Paraphrase Detection**
  We used the binary cross entropy loss function to update model weights and accuracy to evaluate our results.
- **Semantic Textual Similarity**
  We used mean squared error to update model weights and correlation between actual labels and predictions to evaluate our results.

5.3 Experimental details

Our experiment was carried out in two phases, first with high-level model architecture and extended approaches, and then with hyperparameters. For all experiments after the milestone, we used fine-tuned embeddings. Each run was trained for 8 epochs.
The following table describes the model configurations we used for different approaches. After adding SMART to the model, we changed to use 16 as the batch size as a larger number would lead to memory error.

Approach	Batch Size	Dropout Prob	Learning Rate
Baseline	32	0.3	1e-3
SimpleBert	32	0.3	1e-5
+SMART	16	0.3	1e-5
+Contrastive	16	0.3	1e-5

Table 1: Experiments for Different Approaches

After switching from the baseline model to SimpleBert, the extensions SMART and contrastive learning were introduced to the model in an additive manner, which means the experiment with contrastive learning also includes regularized optimization.

After adding the contrastive learning framework to our final model, we furthered tuned the learning rate and the dropout probability sequentially. We ran with learning rates 10^{-3} and 10^{-5} with a fixed dropout probability of 0.3. After find the learning rate 10^{-5} yielded a better result, we tested various levels of dropout probability, 0.1, 0.3, and 0.5.

5.4 Results

The results of three tasks are visualized on Figure 2 for both the development and test datasets. Although the baseline model predictions were not submitted to the test leaderboard, we can see that it had a poor performance predicting the semantic textual similarity of sentence pairs with a Pearson correlation value lower than 0.4. Adjusting the high-level architecture to SimpleBert largely improved the correlation to 0.8. Besides, the accuracy of the task of paraphrase detection increased from 0.67 to 0.88. However, the accuracy of sentiment classification has been consistent across different approaches with a value around 0.5.

Table 4 contains the overall score and individual task results from on the test leaderboard for the final model with extensions. Introducing SMART, the regularized optimization, boosted the overall test score by 2%. On the other hand, adding a new task of unsupervised contrastive learning did not have a significant effect on the model performance. With both extensions included, the final model reached a score of 0.756.

It is observable from Figure 3 that the training curves followed a similar pattern for three approaches. Although all experiments were trained for 8 epochs, they all converged relatively quickly after 3 epochs. As the training score continued to increase, the development score flattened out around 0.75 at Epoch 3, which signals the possibility of overfitting.

The model using SMART started with the lowest training and development performance

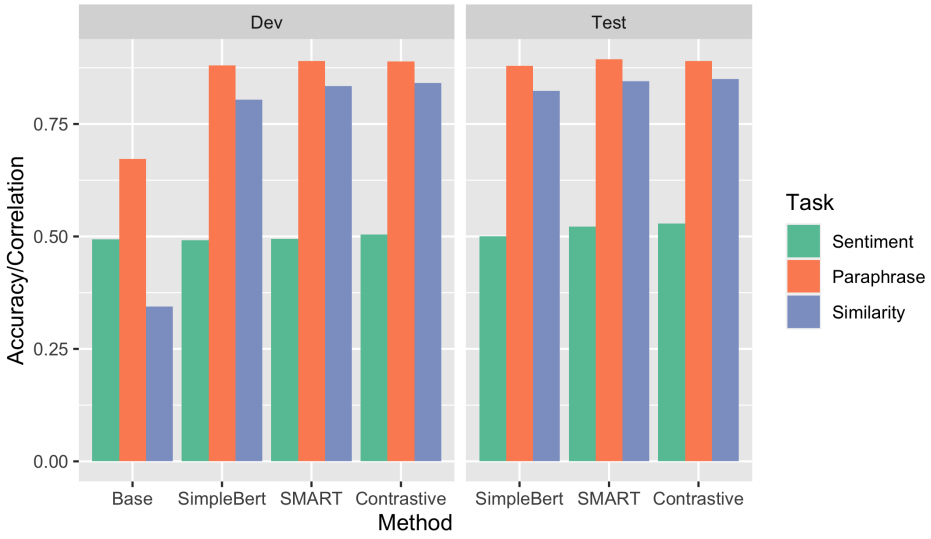

Figure 2: Task Results on Dev and Test Datasets

	Overall Score	Sentiment Acc	Paraphrase Acc	Similarity Corr
SimpleBert	0.734	0.500	0.879	0.824
+SMART	0.753	0.522	**0.894**	0.845
+Contrastive	**0.756**	**0.529**	0.890	**0.850**

Table 2: Test Leaderboard Results for Final Model with Extensions

score at Epoch 1 but ended with a similar score as other models. The regularized optimization was only reflected at early epochs but did not have a larger impact on the final performance.

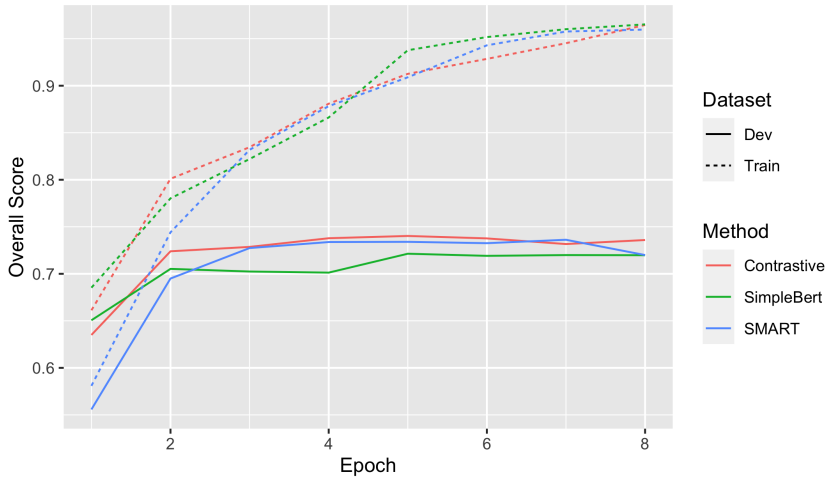

Figure 3: Training Curve of Results on Train and Dev Datasets

The results of hyperparameter tuning is summarized in Table 3. We can see that using a larger learning rate yielded a poor result. In addition, adjusting the dropout probability had little effect on the overall model performance. We failed to reach a better score through sequential hyperparameter tuning.

Tuned Parameter	Value	Overall Score
Learning Rate	10^{-5}	**0.740**
	10^{-3}	0.30
Dropout Prob	0.1	0.737
	0.3	0.740
	0.5	0.735

Table 3: Scores of Hyperparameter Tuning on the Dev Dataset

6 Analysis

In comparison with the original paper, we used the vanilla Bregman proximal point (VBPP) method instead of the momentum Bregman proximal point (MBPP) method for optimization due to the length of this project. MBPP adds an additional momentum to the update which can accelerate the Bregman proximal point method. This may explain the regularized model's failure to obtain more significant improvements on the dev set. For furture work, we will implement the MBPP method to gain full strength of SMART and re-run our model.

As seen in the results above, adding unsupervised contrastive learning did not have a significant impact on the overall model performance. A potential reason is the small training dataset size. We only used 20000 out of the one million sentences in the English Wikipedia dataset for the sake of training time. Further, we was not able to tune the hyperparameters such as temperature for the SimCSE task. The sentence embeddings could not be improved also possibly because we did not implement a supervised SimCSE, which is expected to be more effective than the unsupervised learning.

We also compare the time and space complexity for different model architectures. Each model is ran on a 24 GB GPU and we record the training and evaluation time for one epoch. For Simple BERT, a batch size of 32 can fit the GPU while for the added extensions, a batch size of 16 already fits the GPU.

	training + evaluation / epoch
SimpleBert	18 min 21 sec
+SMART	29 min 25 sec
+Contrastive	38 min 27 sec

Table 4: Training and evaluation for Final Model with Extensions

The results in table 4 indicate that compared to Simple BERT, model regularized by SMART takes about 10 minutes longer to run and model extended by contrastive learning doubles the running time. This may be explained by the fact that SMART and constrastive learning are trained on smaller batch sizes. If larger GPUs are accessible, we will be able to dive deeper in the comparison of time complexities between different models.

7 Conclusion

In the project, we implemented key aspects of BERT including multiheaded self-attention and transformer layers with adam optimizer. Multi-task learning was applied to obtain robust sentence embeddings that can generalize across different natural language tasks. We experimented with different architectures for the sentiment classification, paraphrase detection and semantic textual similarity tasks. Surprisingly, a relatively simple structure where two sentences are concatenated before passing into the encoder together outperformed other attempts. Further, we employed a regularized optimization strategy to prevent aggressive updating during fine-tuning. In addition to the three main tasks, we added a new task using unsupervised contrastive learning aiming to get better overall embeddings. However, the improvement was not as good as we expected.

This project has several limitations. First of all, we did sequential hyperparameter tuning due to time constraints while grid search is preferred for a thorough examination. Regarding to contrastive learning, we only used a very small fraction of the dataset and could not improve the performance

significantly. A next step is to use a supervised contrastive learning framework and implement the momentum Bregman proximal point for optimization to gain the full strength of SMART.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

Weizhu Chen Xiaodong Liu Jianfeng Gao Tuo Zhao Haoming Jiang, Pengcheng He. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Association for Computational Linguistics (ACL)*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

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.