MOPS: Memory Occupancy and Performance Surveying when Hard Parameter Sharing in a Multi-Task minBERT Context

Stanford CS224N Default Project

Anthony Weng
Department of Computer Science
Stanford University
`ad2weng@stanford.edu`

Mark Bechthold
Department of Computer Science
Stanford University
`markpb2@stanford.edu`

Callum Burgess
Department of Computer Science
Stanford University
`callumb@stanford.edu`

Abstract

This project firstly implements the BERT (Bidirectional Encoder Representations from Transformers) architecture, which has cemented itself as a state-of-the-art model tackling a wide variety of NLP tasks since its introduction in 2018. We then aim to create hard parameter sharing regimes in a multitask BERT architecture, analyzing the efficacy of various hard parameter sharing strategies. To do so, we will directly compare the performance of multitask BERT architectures with varying amounts of “frozen", “shared", and “unshared" layers across different sets of tasks; we also strive to comment on performance difference given varying time and memory constraints during training.

1 Key Information

No external collaborators or external mentor. We are not sharing this project with other classes.

2 Approach

We complete the initial baseline of the default final project by implementing key parts of the minBERT architecture [1] and training our model on sentiment analysis as covered in the Default Project Handout [2]. In this milestone report, we mainly discuss the minBERT architecture, and give a brief overview of the general approach of our future model that will use parameter sharing.

To implement minBERT, we first extract word, positional, and task embeddings and feed these into a multi-headed attention model. Keyly, the attention of each head is given by the following formula, where Q represents tokens in a query sequence of a given length, K the key vectors, d_k the dimension of the key vectors, and V the value vectors for the sequences [3]:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V.$$

After computing $S = \frac{QK^T}{\sqrt{d_k}}$, we apply an attention mask setting all values corresponding to a padding token in S to a large negative value. This means that when computing the softmax these values, we contribute minimally to the attention. The final multi-headed attention result is then:

$$Multihead(Q, K, V) = Concat(head_1, ..., head_h)W^O,$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ and W_i^Q, W_i^K, W_i^V, W^O are all learned parameter projection matrices.

After passing our embeddings through the multi-headed attention, we project the output using a dense linear layer. We then add the input embeddings to the output of the dense layer with dropout and pass the sum through a layer-norm layer, computing: $y = \frac{x-\mathbb{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}} * \gamma + \beta$, where γ and β are learned parameters. Next, we pass the outputs of the layer-norm layer through a feed forward network, applying a GeLU activation function: $\text{GELU}(x) = x * \Phi(x)$ where $\Phi(x)$ is the CDF of the Gaussian distribution.

Finally, we pass the outputs through another additive and normalization layer with a residual connection with dropout to get the outputs of our minBERT. The complete architecture of our implementation includes passing the inputs through the minBERT, and passing its outputs through a linear layer with dropout to output the correct number of logits depending on the prediction task. We minimize the cross entropy loss between the predicted logits and the ground truth values:

$$J(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{n} y * \log(softmax(\hat{y})),$$

where $\hat{y}$ are the outputted logits from our model. To perform this minimization we use our own implementation of the AdamW gradient descent algorithm [4] where we use the first and second moment and weight decay to improve convergence time. We used the computationally more efficient method [5] given by the following update rule:

$$\theta'_t = \theta_{t-1} - \alpha_t \frac{m_t}{\sqrt{v_t + \epsilon}}; \theta_t = \theta'_t - \alpha w \theta'_t$$

, where $\alpha_t = \alpha\sqrt{1-\beta_2^2}/(1-\beta_1^2)$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $g_t = \nabla f_t(\theta_{t-1})$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$ and α, β_1, β_2 are model hyper-parameters.

In our future survey of different minBERT multitask parameter sharing architectures, we aim to analyze the effectiveness of hard and soft parameter sharing by constructing models with varying amounts of “frozen" (F), “shared"(S), and “individual"(I) layers across different set tasks. Frozen layers have constant and unchangeable weights, shared layers can be updated by any task through standard update rules, and individual layers are independent to each task. As a general baseline, we will compare the performance of multitask models with varying amounts of parameter sharing (different amounts of F-S-I ratios) to the implemented minBERT model with no parameter sharing.

We further discuss the baselines for our project in the experiments section below.

3 Experiments

Two sets of experiments have been conducted to date, with one set evaluating our minBERT model's performance on two sentiment classification tasks, and the other set evaluating a preliminary baseline of the minBERT architecture extended for multitask learning (with the tasks being sentiment classification, paraphrase detection, and semantic textual similarity evaluation).

3.1 Data

For the sentiment classification experiments, the Stanford Sentiment Treebank (SST) dataset [6] and CFIMDB dataset were used. For the multitask learning experiments, the SST dataset was used once more in addition to the Quora dataset [7] (used for the paraphrase detection learning task) and the Semantic Textual Similarity (STS) dataset [8] (used for the semantic textual similarity learning task). All datasets attributes are described at length in the default project handout [2].

3.2 Evaluation method

Accuracy of predicted labels is used to evaluate model performance on the sentiment classification and paraphrase detection tasks. For the semantic textual similarity task, we use the Pearson correlation of the true similarity values and the predicted similarity values.

3.3 Experimental details

Expt 1: Sentiment Classification Tasks

We perform sentiment classification on each dataset using pretrained BERT embeddings [9] and then fine-tune the embeddings on the respective datasets prior to prediction. Our BERT-based classifier applies a dropout layer and then a linear layer (both trainable) to a BERT model sentence embedding before using the resulting vector (unnormalized) as the input to the cross-entropy loss computation.

When using the pretrained embeddings, a dropout probability of $p = 0.3$ and a learning rate of $\alpha = 1e^{-3}$ were used. Training for 10 epochs and generating predictions took approximately 30 minutes. When using fine-tuned embeddings, a dropout probability of $p = 0.3$ and a learning rate of $\alpha = 1e^{-5}$ were used. Training for 10 epochs and making predictions took approximately 1.5 hours.

Expt. 2: Multitask Learning Baseline Model

The same basic architecture of retrieving a BERT sentence embedding and passing it through a dropout and linear layer from Expr. 1 is maintained. Important differences include: task-specific dropout and linear layers are used; and for the paraphrase detection and semantic textual similarity tasks, the embeddings for each question/sentence pair member was retrieved and summed, with the sum being used as the input for downstream layers.

This multitask learning baseline model uses pretrained BERT embeddings and only trains on the sentiment classification task. A dropout probability of $p = 0.3$ for all dropout layers and a learning rate of $\alpha = 1e^{-3}$ were used. Training for 10 epochs and making predictions took approximately 2 hours.

3.4 Results

Table 1: Performance metrics and scores achieved on development sets for:

Table 1A: Sentiment classification

	SST	CFIMDB
Pretrained	39.9%	75.1%
Finetuned	51.9%	95.5%

Table 1B: Multitask learning baseline model

Task	Dataset	Performance	Score
Sentiment class.	SST	39.3%	.393
Paraphrase detect.	Quora	55.2%	.552
Semantic textual sim.	STS	-.0067	-.0067

For the sentiment classification tasks, for each combination of dataset and pretrained versus finetuned embeddings, we achieve accuracies similar to the stated benchmark figures [2].

For the multitask learning baseline model, given that we train under conditions similar to those of ‘SST-Pretrained’ from Table 1A, we achieve expected, better-than-expected, and expected-but-poor performance on the sentiment classification, paraphrase detection, and semantic textual similarity tasks, respectively.

This pattern may have arisen as the degree to which two sentences paraphrase each other should correlate with how strongly they express the same sentiment, whereas the sentiment of two sentences could unrelated to their semantic similarity. Together, these results suggest that a more robust multitask learning baseline model should incorporate training on all three tasks for fair assessment of nontrivial improvements made via hard parameter sharing. Regimes with parameter sharing for some tasks and not others (e.g., a shared layer for only sentiment classification and paraphrase detection) may be a promising direction for maximizing performance.

4 Future work

In our future survey of different minBERT multitask parameter sharing architectures, we aim to analyze the effectiveness of hard and soft parameter sharing by constructing models with varying amounts of “frozen" (F), “shared"(S), and “individual"(I) layers. An interesting question that has not yet been explored by our reference papers is intelligently choosing to share these resources between tasks that are more intimately related to each other. In approaching this question, we will strive to evaluate the difference in parameter sharing architectures between different pairs and sets of tasks and creating a general baseline for hard parameter sharing model.

Furthermore, we hope to extend our analysis of the multitask minBERT model to look at its performance given certain time and memory constraints. To analyze the model’s improvement over time, we will store and test our model’s parameters at varying time steps. To analyze the impact of memory, we will graph the performance dependency on the dimensions of our learned parameters after undergoing multiple model trainings with this varied memory. We then hope to compare resource-limited performance to the baseline of standard BERT architecture performance as previously discussed.

References

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

[2] Cs 224n: Default final project: minbert and downstream tasks.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017.

[6] https://nlp.stanford.edu/sentiment/treebank.html.

[7] https://www.quora.com/q/quoradata/first-quora-dataset-release-question-pairs.

[8] https://aclanthology.org/s131004.pdf.

[9] https://huggingface.co/bert-base-uncased.