Margin for Error: Exploration of a Dynamic Margin for Cosine-Similarity Embedding Loss and Gradient Surgery to Enhance minBERT on Downstream Tasks

Stanford CS224N Default Project

Jimming He
Department of Computer Science
Stanford University
jimming@stanford.edu

Alex Kwon
Department of Computer Science
Stanford University
alexkwon@stanford.edu

Abstract

We are exploring BERT because it was the first transformer model ever to accommodate bidirectional context understanding. Our problem is two-fold: 1) implement a working miniature version of BERT (minBERT) to perform sentiment analysis, and 2) explore extensions of minBERT on sentiment analysis as well as additional downstream tasks such as paraphrase detection and semantic textual similarity (STS). Our goal for the first part of our problem is to meet or exceed baseline metrics set by the CS 224N teaching team as described in the Default Final Project Handout. Our goal for the second part of our problem is to extend minBERT by implementing cosine-similarity embedding loss for our training objective function with a dynamic margin as well as explore gradient surgery to hone multi-task learning when tasks may conflict with each other. We find that we accomplish our first goal by successfully implementing minBERT as well as obtain our best performance using cosine-embedding loss with a dynamic margin with gradient surgery.

1 Key Information to include

- Mentor: Arvind Mahankali
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Team contributions: Although we were together most of the time when completing these tasks, Jimming took the lead in implementing minBERT and its extensions, while Alex took the lead in conducting background research and drafting the final report.

2 Introduction

In our increasingly digital world full of knowledge dissemnation in the form of language, big data, technological advancements, and enhanced human-computer interaction, the need for AI in NLP has never been greater. Devlin et al. (2018) introduced the world to Bidirectional Encoder Representations from Transformers (BERT), the first transformer-based model to make use of bidirectional word representations that simultaneously analyze both leftward and rightward context of a sentence. BERT's ability to capture subtle meanings of words based on their context made it effective for addressing complex NLP tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity (STS). We will tackle these tasks through a two-step approach: 1) implementing a working miniature version of the BERT model (minBERT) to specifically perform sentiment analysis, and 2) explore two extensions of minBERT on sentiment analysis as well as paraphrase detection and STS.

For our first step, we implement multi-head self-attention and the transformer layer, as well as its corresponding AdamW optimizer.

We then test our two extensions. We first implement cosine-similarity embedding loss for our training objective function, as initially proposed in Reimers and Gurevych (2019), and explore the possibility of incorporating a dynamic margin into this objective function. Cosine-similarity embedding loss has been shown to help fine-tune BERT specifically on STS, but we will implement it for paraphrase detection and well and hope it helps sentiment analysis too. To further examine the effects of fine-tuning, we will fine-tune BERT on multiple tasks (multi-task learning) and implement gradient surgery as initially described in Yu et al. (2020), controlling for the effect of conflicting gradient directions for the different tasks. We found that our best-performing model combined cosine-embedding loss with a dynamic margin and gradient surgery.

3 Related Work

As BERT was first introduced in 2018, there have been several drawbacks discovered in the original model that researchers have since sought to improve. One disadvantage, as described in Reimers and Gurevych (2019), is that the naive BERT implementation simply averaged the BERT output layer embeddings or used the first sentence-level classification [CLS] token's output, producing sentence embeddings unable to capture the *very* fine-grained semantic relationships between sentences. As such, Reimers and Gurevych (2019) was the first to propose introducing cosine-similarity embedding loss into their objective function to fine-tune their novel BERT model, complementing the model's improved semantically-enriched sentence embeddings. They found improved performance in several downstream NLP tasks.

As cosine-similarity embedding loss helps task-specific performance, we also wanted to explore multi-task learning to fine-tune BERT on multiple tasks at the same time. Several papers stimulated our interest and inspiration in this topic. Stickland and Murray (2019) harnessed projected attention layers enabling task-specific adaptions to fine-tune a single BERT model on multiple tasks. Bi et al. (2022) added togehter the losses from different tasks, category classification and named entity recognition, to enhance BERT's news encoding capability. However, we ultimately settled on exploring gradient surgery as originally proposed by Yu et al. (2020). Since our downstreak tasks are somewhat different, we thought gradient surgery would be the best multi-task learning option to explore, as it projects the gradient of one task onto the normal plane of a conflicting or different task.

Overall, our work builds on top of what has already been explored with cosine-similarity embedding loss and replicates gradient surgery to confirm its efficacy in multi-task learning.

4 Approach

4.1 Implementing minBERT

For our first step, implementing minBERT, we follow the Default Final Project Handout. We first implement the multi-head attention layer of the transformer, as depicted in Figure 1, which maps a query and a set of key-value pairs to an output. We compute the query Q's dot products with each head's corresponding keys K and apply softmax to these dot products (raw attention scores) to obtain weights (probabilities) representing attention level on each value. Then, we multiply the weights by values V to yield a weighted sum (attention mechanism) for each head, concatenate these outputs, and apply a final linear projection to transform the concatenated output into the desired output's dimensions.

We then implement the actual transformer layer, as depicted in Figure 2. We first implement the `addnorm` function, which applies the dense layer, dropout, and layer normalization to the input. Next, we implement the `forward` function, where we integrate multi-head attention and feed-forward layers with our implemented `addnorm` function. Lastly, we implement the `embed` function, which obtains input embeddings and applies normalization.

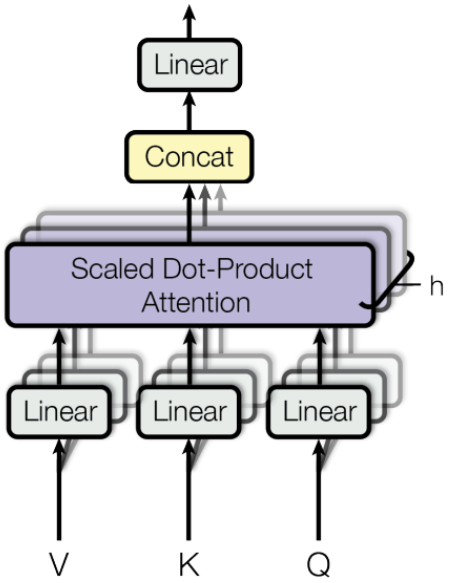

Figure 1: Multi-Head Self-Attention Mechanism from Vaswani et al. (2017)

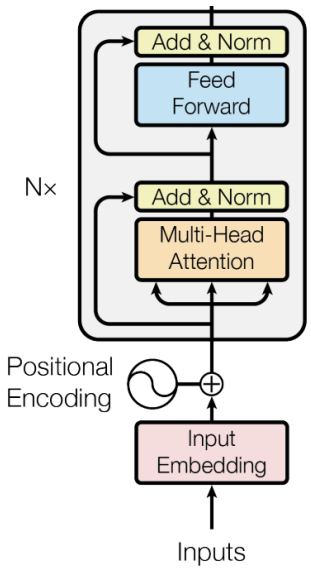

Figure 2: Encoding Layer for BERT Transformer from Vaswani et al. (2017)

4.2 Extending minBERT: Cosine-Similarity Embedding Loss with a Dynamic Margin

After confirming that our minBERT implementation works, to enhance minBERT's performance on both paraphrase detection and STS, we extend it by using cosine-similarity embedding loss instead of standard binary cross-entropy loss for our training objective function as well as implementing a dynamic margin for this embedding loss. Specifically, we define the cosine-similarity embedding loss as that taken from PyTorch's CosineEmbeddingLoss function[1], where $cos(x_1, x_2)$ represents the cosine similarity between two sentence embeddings. For STS, which is the task that cosine-similarity embedding loss has been previously shown to be useful for:

$$\text{CosineEmbeddingLoss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2) & \text{if } y = 3, 4, 5 \\ \max(0, \cos(x_1, x_2) - \text{margin}) & \text{if } y = 0, 1, 2 \end{cases}$$

Note that this function is adjusted to accommodate the labels of our STS task ranging from 0 (not at all related) to 5 (equivalent meaning). For paraphrase detection:

$$\text{CosineEmbeddingLoss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2) & \text{if } y = \text{"Yes"} \\ \max(0, \cos(x_1, x_2) - \text{margin}) & \text{if } y = \text{"No"} \end{cases}$$

Note that this function is adjusted to accommodate the labels of our paraphrase task: "Yes" means that two sentences are paraphrases of each other, and "No" means otherwise.

Described in words, $y = 3, 4, 5$ ($y =$ "Yes" for paraphrase detection) means that the input sentences x_1 and x_2 are similar, so minimizing the loss $1 - \cos(x_1, x_2)$ between them maximizes their cosine similarity $\cos(x_1, x_2)$. Furthermore, $y = 0, 1, 2$ ($y =$ "No" for paraphrase detection) means that the input sentences x_1 and x_2 are dissimilar, so loss will be 0 unless the cosine similarity $\cos(x_1, x_2)$ is greater than the fixed margin (usually 0), which will yield a positive loss to train the model to rank these sentences as dissimilar. Based on the equation, we can see that a lower margin means that the model is more sensitive, as more sentence pairs would be considered to be dissimilar and have a positive loss. The opposite is true for a higher margin. Thus, we can equate the margin to being the "margin of error/discrepancy" between two sentences to be considered dissimilar.

[1] `https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html`

We want to explore the loss function's margin. When analyzing this equation, we thought: *why does the margin have to be fixed? Can it change to adapt to accommodate different types of training pairs?* For example, we thought that sentence pairs with ambiguous relationships should have a lower margin during fine-tuning to encourage more separation. As such, we explore the possibility of implementing a *dynamic* margin to the cosine-similarity embedding loss function. **To our knowledge, we are the first research group to explore a dynamic margin for the cosine-similarity embedding loss to tackle an NLP task**.

Currently, we have developed a prototype DynamicMarginCosineLoss class, defined within the PyTorch framework and extending torch.nn.Module. Our class dynamically adjusts the loss margin of the CosineEmbeddingLoss function (Algorithm 1), encouraging the model to decrease its margin and become more sensitive until performance is hindered, which is when margin is then increased to mitigate overfitting and be more forgiving. To simulate learning rate decay, ***marginStep*** is multiplied by $\frac{1}{epoch\#}$ so that ***marginStep*** has less of an effect over time. We experiment with several different initial margins and margin steps.

Algorithm 1 Dynamic Margin Adjustment for Cosine-Similarity Embedding Loss

Initialize:
$previousMetric, currentMetric = 0$
$currentMargin = [0, 0.25, 0.5]$
$marginStep = [0.05, 0.1, 0.2]$
for each epoch **do**
$currentMetric \leftarrow$ EvaluateModel($devSet$)
$improvement \leftarrow currentMetric - previousMetric$
if $improvement > 0$ **then**
$currentMargin \leftarrow currentMargin - \frac{1}{epoch\#} \cdot marginStep$
else
$currentMargin \leftarrow currentMargin + \frac{1}{epoch\#} \cdot marginStep$
end if
$previousMetric \leftarrow currentMetric$
UpdateLossFunctionMargin($currentMargin$)
end for

4.3 Extending minBERT: Gradient Surgery

We also extend minBERT by implementing gradient surgery with PCGrad, as described in Yu et al. (2020), which accommodates multi-task learning and mitigates negative transfer learning between conflicting tasks. To achieve this, we draw batches of equal size from each of the three tasks and compute task-specific gradients ∇_{SA}, ∇_{PD}, and ∇_{STS} to represent the gradients for sentiment analysis, paraphrase detection, and STS respectively. Then, we loop through these gradients, selecting each one as $\mathbf{g_i}$ and then randomly choosing a $\mathbf{g_j}$ out of the remaining two task-specific gradients. If the similarity $s = \mathbf{g_i} \cdot \mathbf{g_j}$ is negative, then we perform gradient surgery on the original $\mathbf{g_i}$ as such:

$$\mathbf{g_i} = \mathbf{g_i} - \frac{\mathbf{g_i} \cdot \mathbf{g_j}}{\|\mathbf{g_j}\|^2} \cdot \mathbf{g_j}$$

Then, we compare $\mathbf{g_i}$ to the remaining task-specific gradient $\mathbf{g_k}$. If their similarity is negative, gradient surgery is again performed on $\mathbf{g_i}$. In this manner, gradient directions of conflicting tasks are effectively neutralized by projecting the gradient of one onto the normal plane of the other's. Summing the three $\mathbf{g_i}$'s from each task-specific gradient yields the final gradient for input to the AdamW optimizer.

5 Experiments

5.1 Data

Below, we list the three tasks we are performing along with their associated dataset(s):

- **Sentiment Analysis**: We primarily use the Stanford Sentiment Treebank (SST) dataset as described in Socher et al. (2013). The SST dataset has 11,855 single sentences extracted from movie reviews labeled between 0–4 as "negative", "somewhat negative", "neutral", "somewhat positive", or "positive", respectively; it is split into train (8,544 examples), dev (1,101 examples), and test (2,210 examples). For our first part, we will also test our working minBERT implementatio on a second dataset called the CFIMDB dataset provided by the CS 224N teaching team, which consists of 2,434 highly polar movie reviews labeled as "negative" or "positive"; it is split into train (1,701 examples), dev (245 examples), and test (488 examples).

- **Paraphrase Detection**: We use a subset of the Quora dataset as described in `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`, which consists of 400,000 question pairs with a label "Yes" or "No" indicating if the two questions in a pair are paraphrases of each other or not; our subset is split into train (141,506 examples), dev (20,215 examples), and test (40,431 examples).

- **Semantic Textual Similarity (STS)**: We use the SemEval STS Benchmark dataset as described in Agirre et al. (2013), which consists of 8,628 sentence pairs of varying similarity on a scale from 0 (not at all related) to 5 (equivalent meaning); it is split into train (6,041 examples), dev (864 examples), and test (1,726 examples).

5.2 Evaluation method

For our first step, we will compare the metrics on the sentiment analysis task using our baseline minBERT implementation to the baseline metrics provided by the CS 224N teaching team on page 14 of 28 in the Default Final Project Handout.

For our second step, to evaluate our extensions of minBERT, we will compare the metrics on all three tasks using our extended minBERT model to the baseline minBERT model for both the cosine-similarity embedding loss with dynamic margin extension as well as the gradient surgery extension. Describe the evaluation metric(s) you use, plus any other details necessary to understand your evaluation. For reporting sentiment analysis and paraphrase detection metrics, accuracy is used. For reporting STS metrics, Pearson coefficient is used.

5.3 Experimental details

For our first step, testing baseline minBERT, we use the hyperparameters suggested by the default code starter code for all experiments:

- Number of epochs: 10
- Batch size: 8
- Dropout: 0.3
- Pretraining learning rate: 1×10^{-3}
- Finetuning learning rate: 1×10^{-5}

For extending baseline minBERT using cosine-similarity embedding loss with a dynamic margin, we maintained the suggested hyperparameters and changed the following variables: initialMargin and marginStep, and whether the model got more sensitive (reduced margin) or less sensitive (increased margin) when performance decreased after one epoch.

For extending baseline minBERT using gradient surgery, we maintained all suggested hyperparameters.

We used a NVIDIA A100 Tensor Core GPU via Google Colab to run these experiments. The training time for each experiment hovered around 40 minutes.

5.4 Results

5.4.1 Our minBERT Implementation

We first evaluate our minBERT implementation on the sentiment analysis task and compare it to the CS 224N teaching team-provided baseline accuracy metrics on the SST and CFIMDB dev sets (Table 1).

Method	Dev Accuracy			
	PT for SST	PT for CFIMDB	FT for SST	PT for SST
Our Implementation	**0.465**	**0.878**	**0.520**	**0.967**
CS 224N Baseline	0.390 (0.007)	0.780 (0.002)	0.515 (0.004)	0.966 (0.007)

Table 1: Dev accuracy results of our minBERT implementation compared to baseline metrics provided by the CS 224N teaching team. Standard deviation in parentheses. PT: pretraining; FT: finetuning.

We see that our minBERT implementation significantly outperformed the CS 244N teaching team-provided baseline metrics for all four sentiment analysis tasks. Indeed, for three out of the four tasks, our minBERT implementation yielded an accuracy greater than one standard deviation above that of the CS 224N baseline, demonstrating that our minBERT implementation works and allowing us to confidently conduct experiments on minBERT extensions.

5.4.2 Gradient Surgery

Note that implementing cosine-similarity embedding loss with a dynamic margin and gradient surgery were two separate extensions. We first ran experiments just using gradient surgery, and after evaluating its effect on the metrics, we saw if it was good to implement with the cosine-similarity embedding loss. We tested gradient surgery on all three tasks with the default using all suggested hyperparameters as described in Section 5.3. Our findings are reported in Table 2.

Method	Finetuning Dev Metrics: Surgery		
	Sentiment Analysis (SST)	Paraphrase Detection (Quora)	STS (SemEval)
Baseline Implementation	0.501	0.392	0.312
Gradient Surgery	0.524	0.477	0.393

Table 2: Dev results of minBERT with gradient surgery.

We found that gradient surgery helped improve our performance by quite a bit. It seems that these tasks are dissimilar enough for gradient surgery to mitigate the negative effect from training on conflicting tasks. As such, we implement gradient surgery in all future experiments.

5.4.3 Cosine-Similarity Embedding Loss + Gradient Surgery

We then ran tests harnessing the power of gradient surgery using cosine-similarity embedding loss as the training objective rather than standard binary cross-entropy loss. We tested cosine-similarity embedding loss using a static margin of 0 as well as a dynamic margin using Algorithm 1, as described in Section 4.2. Our results are in Table 3.

Variable			Finetuning Dev Metrics		
Type	Initial Margin	Margin Step	Sentiment Analysis (SST)	Paraphrase Detection (Quora)	STS (SemEval)
Static	0	N/A	**0.511**	0.461	0.870
Dynamic	0	0.05	0.503	**0.479**	**0.880**
Dynamic	0.5	0.05	0.509	0.463	0.872
Dynamic	0.5	0.1	0.510	0.454	0.862

Table 3: Dev results of minBERT using cosine-similarity embedding loss using a static margin or dynamic margin.

We found that there was a very marginal improvement in performance when using a dynamic margin compared to a static margin. The static margin performed the best on the sentiment analysis task with an accuracy of 0.511, which makes sense because a dynamic margin wouldn't help in a task such as sentiment analysis because two sentences aren't being compared anyways. However, the model with an initial margin of 0 and dynamic margin step of 0.05 marginally outperformed all other models in paraphrase detection accuracy and STS Pearson correlation coefficient. As such, albeit slightly, a dynamic margin allows the model to adjust its training to be more or less sensitive across epochs, enhancing its performance.

5.4.4 Final Test Leaderboard Submission

Our best model from Subsections 5.4.2 and 5.4.3 that obtained the highest overall test accuracy was the one that combined cosine-similarity embedding loss with a dynamic margin (initial margin of 0 and margin step of 0.05) and gradient surgery. We submitted the predictions from this model to the Test Leaderboard and obtained the metrics shown in Table 4.

Test Metrics			
Sentiment Analysis (SST)	Paraphrase Detection (Quora)	STS (SemEval)	Overall Score
0.524	0.477	0.393	0.566

Table 4: Test results of our cosine-embedding similarity loss with a dynamic margin + gradient surgery model.

Our model ended up performing moderately well on the test set but unfortunately quite poorly on STS. Perhaps the dynamic margin caused the model to overfit to the training data and performing poorly on STS. Paraphrase detection was second-best. The model surprisingly had the best performance on the sentiment analysis task, which we attribute to gradient surgery.

6 Analysis

Qualitatively, we can see that our system performs well on sentiment analysis and not so well on semantic textual similarity. Diving deeper into the input, we can see that our model does indeed overfit for STS. There are many similar sentence pairs with a highly dissimilary cosine similarity, and vice versa. We believe that normal loss functions such as standard binary cross-entropy loss would help the model reduce overfitting on the traiing set and improve its performance on the test set. Furthermore, we saw a similar phenomenon with paraphrase detection: several sentences that seem very similar to the human eye were classified as not paraphrased, and vice versa. We again attribute this to our dynamic margin in the cosine-similarity embedding loss. When digging deeper into sentiment analysis, we see that although there is variation between ground-truth and predictions, it does a better job analyzing sentiment than it does comparing two sentences. Our dynamic margin seemed to be overkill.

7 Conclusion

In our project, we implement a working minBERT and successfully extend it to 1) accommodate cosine-similarity embedding loss with a dynamic margin, and 2) build on top of this using gradient surgery to facilitate multi-task learning. Our baseline minBERT implementation outperformed the CS 224N teaching team-provided baseline metrics on all tests. After integrating our extensions, we found that the strongest model was the one that combined cosine-similarity embedding loss with a dynamic margin and gradient surgery. Furthermore, we are the first research group to successfully implement a dynamic margin into the cosine-similarity embedding loss objective fuction to address an NLP task.

Our biggest limitation was computational complexity. We wanted to run more models testing more initial margins and margin steps, but we didn't have the resources or time to do so. In the future, we would run more experiments with the dynamic margin to further hone the best algorithm for adjusting it as well as exploring the possibility of having a unique margin for each training sample rather than for each epoch.

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*, pages 32–43, Online. Association for Computational Linguistics.

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, Online.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*, pages 3982–3992, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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, Online. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Project attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995, Online. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, Online.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.