# Multi-BERT: Investigating Methods for BERT Multitask Learning

Stanford CS224N Default Project

**Zach Benton**
Department of Computer Science
Stanford University
zbenton@stanford.edu

## Abstract

Multitask learning requires a model to recognize useful shared information across tasks—it is a complex problem to build a model that can do this well. We extend the $\text{BERT}_{\text{BASE}}$ model to succeed at three tasks simultaneously—sentiment classification, paraphrase detection, and semantic similarity—via experimenting with combinations of successful multitask finetuning methods such as multiple negatives ranking loss and annealed batch sampling. We find that annealed sampling and a concatenation architecture achieves the best performance, and find three qualitative categories of error that this best performing model tends to make.

## 1 Key Information to include

- Mentor: Jingwen Wu
- Sharing project: No

## 2 Introduction

BERT, released by Devlin et al. (2018), is a pre-trained model that generates sentence embeddings, using the novel idea of leveraging both right and left contexts in all of the layers of the model. The base model $\text{BERT}_{\text{BASE}}$ achieved a large step forward in performance on eleven language tasks, cementing itself as a breakthrough in the field of natural language processing. However, the original BERT paper only proved its ability to excel at a single task. How do we design a BERT model that performs well at an array of tasks, instead of a single one? This question is a complicated one; it asks us to integrate several different loss functions, different gradients, and different dataset sizes.

This paper explores this question of multitask finetuning—we investigate techniques to best extend BERT to achieve high performance on three distinct tasks simultaneously: sentiment classification, paraphrase detection, and semantic similarity. We apply several recent methods for multitask finetuning to our model—including adjusting batch sampling techniques to cope with large discrepancies in dataset size, adjusting the task heads to experiment with sentence comparison, and exploring improved loss functions discovered by recent papers in the field—in an effort to refine its multitask performance, and then we perform a quantitative and qualitative analysis on our best performing model. Lastly, we present a statement on potential ethical challenges with this work as well as methods for lowering their risk.

## 3 Related Work

One of the first papers to discuss finetuning BERT was Sun et al. (2019), a paper released only a few months after the original BERT paper. The paper mainly performs a hierarchical sweep, beginning

with finding the best hyperparameters (including learning rate, layer-wise learning rate decay, truncation methods, and more) for single-task finetuning. The authors then optimize hyperparameters for further pre-training, then for multitask finetuning, assuming that the best single-task hyperparameter setting still holds. One relevant finding in their paper was the inconsistency of multitask finetuning on improving within-task performance. We encounter this problem in the project, as we found that gradients for the paraphrase detection task conflicted slightly with gradients from other datasets.

Stickland and Murray (2019) was another BERT-based paper that explores multitask finetuning in more depth. The paper explores adapting self attention for multitask finetuning as well as coping with large discrepancies in dataset sizes between tasks. Inspired by the latter, our project conducts a comparison in performance between our own batch sampling method and *annealed sampling*, their fix for large discrepancies in dataset size. We explain these approaches in more detail in section 4.2.

There are several relevant papers released before the original BERT paper; these papers detail methods the authors found useful for a general setting of multitask finetuning or for similar tasks to the ones in this project. For example, Henderson et al. (2017) explores a loss function for training data consisting of positively labelled sentence pairs. In a batch of $K$ positively labelled sentence pairs, the loss function is able to extract $K^2$ examples by considering the negative examples formed with two sentences that are not from the same positive pair. We adapt this method for our setting, where training examples are not necessarily positively labeled. Yu et al. (2020) explores how to mitigate conflicting gradients from multiple tasks, proposing projecting one task's gradient onto the normal plane of the conflicting task's gradient to "zero out" gradient interference. We found that this was not necessary for our tasks, as there was—at worst—very slight interference between gradients.

# 4 Approach

## 4.1 Baseline

The multitask baseline we selected for this project uses a 12-layer BERT model with a single linear classification layer for each of the three tasks. These classification layers take as input the hidden state corresponding to the `[CLS]` token:

- **Sentiment Classification**: We use a task head that outputs five logits, one for each sentiment class. We use cross entropy loss for this task.
- **Paraphrase Detection**: For this task, we apply the BERT model to the concatenation of the two input vectors. The task head outputs a single logit encapsulating the prediction, and this task uses binary cross entropy loss.
- **Semantic Textual Similarity**: For the baseline, we apply the BERT model individually to both input sentences, and output the dot product of the resulting pooled embeddings. We use MSE loss to compare the baseline's prediction to the true label for the sentence pair.

At train-time, we fine tune all parameters of the model, including the task heads. We use a dropout rate of 0.3, a learning rate of 1e-5, and train for 10 epochs. We use round robin batch sampling, ending an epoch after seeing every example in the smallest dataset. We use a batch size of 32 for the baseline.

## 4.2 Model Refinements

After implementing the baseline, we reasoned that the STS task had the largest room for performance improvement. Sentiment classification is a difficult task, as there are five sentiment categories with some room for subjectivity in the true labels, so it may be difficult to improve accuracy beyond a low threshold. The baseline already performed well on paraphrase detection—an easy binary classification task—leaving little room for improvement. Thus, we turned to strategies for improving the model's performance on semantic similarity.

**STS Concatenation**: For the STS task, the baseline applies the BERT model and an extra linear layer to both input sentences, then outputs the cosine similarity between the two task head outputs. One direction we explored was to apply the BERT model a single time to the concatenation of the two sentences, then apply a task head that outputs a single value, similar to the baseline's approach for paraphrase detection. We apply tanh to the logit outputted by the linear layer before returning to

ensure that the model predicts a value between -1 and 1, so that we can scale it when computing the loss function.

**Annealed Sampling**: The baseline uses a round robin sampling method to choose the dataset that the next batch of training data will come from, and it thus trains on an equal number of batches from each dataset. This approach does not leverage the vast size of the QQP dataset. One alternative approach, which we call equitable sampling, randomly samples a dataset at each iteration for the next batch of training data, weighting the probability $p_i$ of a given dataset being chosen to be proportional to its size $N_i$:

$$p_i \propto N_i$$

This distribution ensures that the expected number of times that we see each example in an epoch is exactly 1. Annealed sampling is a similar approach explored by Stickland and Murray (2019) in which we introduce an exponent $\alpha$ that depends on the current epoch $e$ and the total number of epochs $E$:

$$p_i \propto N_i^{\alpha}, \quad \alpha = 1 - 0.8\frac{e-1}{E-1}$$

With huge discrepancies in dataset sizes, we expect equitable sampling to cause overrepresentation of gradients from large datasets, since there may be many gradient updates in a row from the same dataset. Annealed sampling seeks to mitigate this problem while also leveraging the size of larger datasets by initially sampling equitably, then sampling more and more uniformly as training proceeds.

**Cosine Similarity**: The baseline STS task head calculates the dot product between the hidden states of the [CLS] token for each of the input sentences. Instead, we experimented with outputting the cosine similarity of these two vectors, inspired by Reimers and Gurevych (2019). The cosine similarity of two vectors is simply the dot product between the normalized (length-1) scaling of each vector. We reasoned that the [CLS] hidden states for different sentences may vary hugely in length because they are large vectors, so normalizing the dot product could allow the model to adjust vector directions instead of their lengths.

However, we hypothesize that this adjustment may cause the model to lose some expressive power. Using cosine similarity, two sentences whose pooled embeddings have similar direction but widely different lengths are treated similarly, whereas the raw dot product does differentiate between two vectors with similar direction but varying magnitudes.

**Multiple Negatives Ranking Loss**: One strategy we use is Multiple Negatives Ranking (MNR) loss, a framework for loss functions proposed by Henderson et al. (2017). This method operates on a batch of $K$ positively labelled sentence pairs $(x_i, y_i)$, seeking to minimize the distance between each pair $(x_i, y_i)$ while maximizing the distance between $(x_i, y_j)$ for $i \neq j$. To do this, we learn a scoring function $S(x, y)$ that is large if $x$ and $y$ are similar and small if $x$ and $y$ are dissimilar. We use the scoring function to estimate the joint probability $P(x, y) \propto e^{S(x,y)}$ that $x$ and $y$ are similar sentences.

We can derive the equation for the loss function $\mathcal{L}_{\text{MNR}}$ using $P_{\text{approx}}(y \mid x)$:

$$\mathcal{L}_{\text{MNR}}(\mathbf{x}, \mathbf{y}, \theta) = -\frac{1}{K}\sum_{i=1}^{K} \log P_{\text{approx}}(y_i \mid x_i) \tag{1}$$

$$= -\frac{1}{K}\sum_{i=1}^{K} \log \frac{e^{S(x_i,y_i)}}{\sum_{j=1}^{K} e^{S(x_i,y_j)}} \tag{2}$$

Using a scoring function that is simply the dot product of the pooled hidden states of $x_i$ and $y_j$, it is efficient to compute $\mathcal{L}_{\text{MNR}}$. Given a matrix $M_x$ whose rows are the hidden states of the batch of the sentences $x_i$, and a matrix $M_y$ whose rows are the hidden states of the batch of sentences $y_i$, we can compute all $K^2$ dot products as a single matrix product:

$$D = M_x M_y^\top$$

Then, $D(i, j)$ is the dot product between the hidden state for $x_i$ and the hidden state for $y_j$. One problem with this approach in general is that it requires only positively labeled data, whereas sentence pairs from SemEval are labelled with a continuous score between 0 and 5. Our approach here is to only train on examples whose similarity score is at least 4. The drawback of this strategy is that we are left with a much smaller set of training data. Thus, instead of using MNR loss on its own, we use it in combination with cosine similarity loss; we finetune using MNR loss for the first half of the epochs, then cosine similarity for the rest of the finetuning epochs.

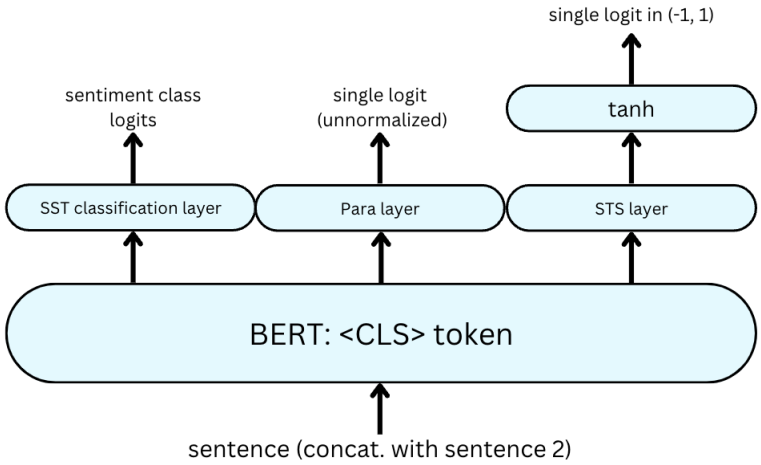


Figure 1: This diagram details the architecture of our models. We mostly experiment with different options for the STS task head.

## 5 Experiments

### 5.1 Data

For the baseline and all further models, we use the same datasets for training and evaluation. For sentiment classification, we use the Stanford Sentiment Treebank (SST) dataset originally created by Socher et al. (2013). This dataset consists of single sentences taken from movie reviews, and each sentence is labelled with a sentiment—0 refers to "negative", 1 to "somewhat negative", 2 to "neutral", 3 to "somewhat positive", and 4 to "positive".

For paraphrase detection, we use the Quora Question Pairs (QQP) dataset, released by Quora in 2017. This dataset contains pairs of questions, each labelled with either 1 (the questions ask the same thing) or 0 (the questions ask different things).

For semantic similarity, we use the SemEval dataset originally created by Agirre et al. (2013). It consists of pairs of sentences, each annotated with a value on the continuous scale from 0 to 5. This label refers to the similarity of the sentences; a high value means that the sentences are semantically similar, whereas a low value means the sentences are semantically unrelated.

| Dataset Name | Train Size | Dev Size | Test Size |
|---|---|---|---|
| SST | 8,544 | 1,101 | 2,210 |
| QQP | 283,010 | 40,429 | 80,859 |
| SemEval | 6,040 | 863 | 1,725 |

Figure 2: Dataset size details

### 5.2 Evaluation method

After each epoch, we evaluate the model on each of the three tasks. For sentiment classification and paraphrase detection, this accuracy score is simply the proportion of correct predictions in the dev set. For sentiment analysis, the score is the Pearson correlation between model's predictions and the true labels, which falls in the range 0 to 1. We also compute the overall performance of the model $\text{acc}_{\text{overall}}$ as a score between 0 and 1, which we use to determine the best model to save at the end of an epoch during training. Letting $\text{acc}_{\text{sent}}$ represent sentiment classification accuracy, $\text{acc}_{\text{para}}$ represent paraphrase detection accuracy, and $\text{corr}_{\text{sim}}$ represent semantic similarity correlation, we compute the overall performance as

$$\text{acc}_{\text{overall}} = \text{acc}_{\text{sent}} + \text{acc}_{\text{para}} + 0.5(\text{corr}_{\text{sim}} + 1)$$

We also tweaked the train-time evaluation function to speed up training. Between train epochs, instead of evaluating on the entire train and dev QQP datasets, we just evaluate on the first 1000 batches for both. Using a batch size of 32, this amounts to evaluating on almost the entire dev dataset, and about 10 percent of the train dataset. We reasoned that this evaluation gives a suitable estimation of the model's performance on both train and dev sets, since it evaluates on a large representative fraction of the datasets.

### 5.3 Experimental details

We trained six models in addition to the baseline model. We created three models with annealed sampling, and three with equitable sampling. Within each of these categories, we varied the STS task head. One model used concatenation, one model used cosine similarity, and one used MNR loss.

We trained these models for 10 epochs using a learning rate of 1e-5, a dropout probability of 0.3, and a batch size of 32 to hit about 90 percent memory utilization on the T4 GPU we trained with. During each epoch, the models sampled and trained on 3000 batches. Total train time for each model was 4 hours.

### 5.4 Results

On the test leaderboard, our best model—Ann + Concat—**obtained an accuracy of 0.536 on sentiment analysis, an accuracy of 0.891 on paraphrase detection, and a Pearson correlation of 0.870 on semantic similarity**.

| Model Name | SST Dev Acc. | QQP Dev Acc. | SemEval Dev Corr. | Overall Performance |
|---|---|---|---|---|
| Ann + Concat | **0.529** | 0.889 | **0.866** | **0.784** |
| Ann + CoSim | 0.524 | 0.900 | 0.388 | 0.706 |
| Ann + CoSim + MNR | 0.512 | 0.892 | 0.410 | 0.703 |
| Equit + Concat | 0.520 | 0.898 | 0.862 | 0.782 |
| Equit + CoSim | 0.514 | **0.904** | 0.428 | 0.710 |
| Equit + CoSim + MNR | 0.510 | 0.901 | 0.420 | 0.707 |
| Baseline | 0.505 | 0.822 | 0.496 | 0.692 |

Figure 3: Model performance results, with the highest performance in each column bolded. "Ann"/"Equit" refers to annealed/equitable sampling, "Concat" refers to input concatenation in STS task head, "CoSim" refers to cosine similarity, and "MNR" refers to MNR loss

It seems that the model that uses annealed sampling and a concatenation task head strategy for STS performs best overall. We expected annealing sampling to perform much better than equitable sampling, but they perform about the same. The creators of annealed sampling, Stickland and Murray (2019), used datasets that varied in size to a greater extent than us—by up to a factor of around 150—whereas the biggest size gap among our datasets is by a factor of 30. Thus, we suspect that

annealed sampling helps the most when datasets differ in size even more extremely than in our project.

We also expected MNR to improve upon cosine similarity, but this improvement did not occur. We suspect no significant improvement occurred because the model scores each example in a similar way between the two methods. MNR loss uses a dot product scoring function, while cosine similarity uses (of course) cosine similarity. These computations are very similar, as cosine similarity is simply the normalized version of dot product scoring, which may explain why the models trained only using cosine similarity perform similarly to the corresponding model trained with cosine similarity and MNR.

## 6 Analysis

We investigate some difficult examples for our top-performing model, the model with annealed sampling and STS concatenation. Our goal here is to provide reasoning for why the model finds these sentence pairs difficult.

**Example One**:

Entrenched interests are positioning themselves to control the network's chokepoints and they are lobbying the FCC to aid and abet them.

"It may be dying because entrenched interests are positioning themselves to control the Internet's choke-points and they are lobbying the FCC to aid and abet them."""

**Label**: 3.2 **Prediction**: 4.4 (+1.2)

One reason that this example was difficult may have been that the two sentences are longer examples than usual. Thus, the sentences may contain more similarities than shorter sentence pairs just as a result of them being longer sentences. Indeed, the two sentences contain long stretches of words that are identical—"... entrenched interests are positioning themselves to control the ...". Thus, we postulate that the model overrates the similarity of longer sentence pairs.

For a quick, unrigorous analysis, we looked at the first 500 examples in the STS dev dataset, isolating sentence pairs that totalled 300 characters or more. We found that only 4 of the 13 long sentence pairs were underestimates, whereas 9 of the 13 were overestimates. Therefore, it seems that the model tends to overestimate the similarity between long sentence pairs.

**Example Two**:

Man, 19, quizzed over teen murder

Man quizzed over Megan-Leigh murder

**Label**: 3.8 **Prediction**: 2.6 (-1.2)

One reason that this example was difficult may have been the small amount of context combined with the unknown words "Megan-Leigh". To successfully label these sentences as somewhat similar, the model would need to recognize that Megan-Leigh could plausibly be the name of the teen. This recognition may be difficult, since these words are likely unknown tokens with little surrounding context, so BERT may struggle to interpret these sentences.

**Example Three**:

Syrian fighter pilot defects to Jordan

Syrian PM defects to Jordan

**Label**: 1.4 **Prediction**: 3.7 (+2.3)

One reason that the model had difficulty with this example may be an inability to leverage knowledge about our world to label this data. World knowledge tells us that there is a huge political difference between the defection of a fighter pilot versus a prime minister, so these sentences should have a small similarity score. However, outside of their subjects, the sentences are perfectly identical, so it makes sense that an agent without access to world knowledge would classify them as more similar than a human.

## 7 Conclusion

In this project, we compared several techniques for optimizing BERT multitask finetuning—implementing methods both described in other research as well as original ideas. We focused on improving the STS task performance metric and experimented with different task heads for this task, including concatenating inputs in a single forward pass through BERT, computing the cosine similarity between the pooled outputs of BERT on each sentence, and using MNR loss with a dot product scoring function. We achieved high dev performance with a relative simple model that uses only annealed sampling and the concatenation task head.

One of the limitations of our work on this project was the limited amount of computing power that we had access to as student researchers. Using a more expensive, higher-powered GPU may have allowed us to run training much faster without having to use tricks to speed up training such as the train-time evaluation function tweak we implemented. It also would have allowed us to feasibly explore more sophisticated architecture and train our models for more epochs. Another limitation was the quality of the SST dataset. The SST classification task is difficult, with five subjective categories (is it always clear what differentiates "somewhat positive" from "positive"?), so there seems to be an upper limit far below 1.00 to the plausibly achievable accuracy for this task.

There is room for future work that builds off this project. One strategy that we wanted to implement, and that we expected would improve the Ann + Concat model performance, was to add shared weights between task heads. We expected this to potentially improve our best model's performance because it facilitates transfer learning, allowing the model to more easily leverage patterns that are useful across multiple tasks in the task head. There is also plenty of room for research that attacks some of the difficulties that we found our model faces with predicting semantic similarity scores.

## 8 Ethics Statement

One ethical challenge of my project in particular results from the fact that my project struggles with sentence embeddings that may require some knowledge of the world. We found in the analysis section that some sentence pairs require world knowledge to succeed at labelling their similarity. This finding complicates the ethics around any applications of a model like this, because this knowledge can inform the best decision to make for a response. For example, if my model were implemented as part of a social media hate post screening measure, then this inability to use real world knowledge means that the model cannot make screening decisions with complete contextual nuance. One way to mitigate this problem in this context would be to simply have a human committee to double check any flagged posts (to cover false positives), as well as a user-end post reporting feature (to cover false negatives) to check the work of the model.

A second ethical challenge of my project in particular is that it may perpetuate racial bias, as its datasets (and English as a language) display this bias. For example, the BERT base model was trained using Wikipedia articles, which may perpetuate racial bias via underrepresentation of topics about or related to racial minorities. Since my project uses the BERT base model, it likely exhibits this bias. Thus, it may perform worse when applied to a task involving race, such as sentiment analysis on social media posts related to race. One strategy to mitigate this bias may be to further train the model on more diverse datasets, or to incorporate fairness into the evaluation of our model (by evaluating it on language tasks involving race and ensuring it performs at a suitable level of fairness across all groups).

## 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. *CoRR*, abs/1810.04805.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

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.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.