Multitask BERT

Stanford CS224N Default Project

Bradley Hu
Department of Computer Science
Stanford University
bradleyh@stanford.edu

Shannon Xiao
Department of Computer Science
Stanford University
sxiao1@stanford.edu

Abstract

In machine learning, it is common to focus on optimizing models for a single, specific task. However, multitask learning is a useful adaptation, particularly in scenarios like multilingual machine translation, where training a model on multiple tasks at once can be beneficial. Multitask learning not only helps in situations of managing numerous tasks, such as in large-scale web applications with limited storage for model parameters, but also reduces the total number of parameters needed. This approach aligns with insights from Stickland and Murray (2019), who highlight the advantages of shared parameters across tasks to enhance efficiency and manageability. Our project explores multitask learning to achieve competitive results, comparable to state-of-the-art models trained on individual tasks. We constructed a model capable of performing three distinct linguistic tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We utilized the Stanford Sentiment Treebank (SST) for sentiment analysis, the Quora Dataset for paraphrase detection, and the SemEval Benchmark Dataset for semantic textual similarity. Initially, our model's baseline performance across these tasks resulted in an average dev accuracy of 0.514. To improve performance, we experimented with various fine-tuning strategies. This experimentation led to notable advancements, with our development set accuracy reaching 0.644 and our test set accuracy improving to 0.652. A key innovation in our approach was the introduction of proportional task sampling Stickland and Murray (2019). This method begins by selecting tasks for training based on the size of their datasets and gradually reduces the emphasis on dataset size as training progresses. Additionally, we integrated Projected Attention Layers (PALs) (Stickland and Murray, 2019) into our model. These layers facilitate the sharing of the model across multiple tasks by introducing minimal task-specific parameters, thereby streamlining the learning process and decreasing training time significantly.

1 Introduction

In the field of Natural Language Processing (NLP), deep learning models, particularly those based on the transformer architecture like Bidirectional Encoder Representations from Transformers (BERT), have significantly advanced our ability to process and understand language. Despite these advancements, creating models that grasp the semantic meaning of sentences for various tasks remains challenging. This project aims to enhance the BERT model so it can more effectively perform multiple NLP tasks by understanding the nuances of language at the sentence level.

One of the main challenges in achieving this goal is the computational burden of training and fine-tuning large models for each specific task, which is not only resource-intensive but also limits the model's ability to generalize across different tasks. Furthermore, adapting these models to new or varied tasks is often impractical due to their vast number of parameters. To tackle these issues, we propose a multitask learning framework that allows our enhanced BERT model to simultaneously address three key sentence-level NLP tasks: sentiment analysis, paraphrase detection, and seman-

tic textual similarity. By focusing on these tasks, we aim to create a versatile model capable of understanding and analyzing the semantic content of sentences in different contexts.

A novel aspect of our approach is the incorporation of Projected Attention Layers (PALs), inspired by Stickland and Murray (2019), into the BERT architecture. These layers enable the model to apply task-specific attention mechanisms within the shared structure of BERT, without significantly increasing the model's size or computational cost, and adapts the model toward task-specificity. We leverage this technique both within the layers of the BERT model as well as at the top layer, with the aim of improving both our model's accuracy and computational efficiency, respectively. However, drawing upon this approach presents its own set of challenges, such as tasks competing with one another to achieve a better learning representation during training. To combat this, we draw on strategies like proportional task sampling, which helps in selecting tasks for training based on their dataset sizes, aiming to balance learning across tasks.

In essence, this project seeks to explore and extend the capabilities of BERT through multi-task learning, focusing on sentence-level semantic understanding across different NLP tasks. By doing so, we aim to contribute to the development of more generalizable NLP models, addressing both the technical and practical challenges of working with large-scale language models.

2 Related Work

This section provides an overview of the foundational works and innovative approaches that informed and inspired our research direction.

2.1 BERT Model and Multitasking

The introduction of the BERT model by Devlin et al. (2018) marked a significant advancement in NLP, demonstrating significant performance improvements across a range of NLU tasks. BERT's pretraining on a large corpus of unlabeled data using tasks like masked word prediction and next sentence prediction enabled it to understand nuanced language patterns before being fine-tuned on specific NLP tasks. This approach not only enhanced the model's performance but also reduced the reliance on large volumes of labeled data for task-specific training.

Building on the versatility of BERT, Liu et al. (2019) expanded its application through a multitask learning framework that allowed for the simultaneous training on multiple text classification attributes, including sentiment classification and pairwise classification tasks such as paraphrase detection and semantic text similarity. By sharing the lower, text-encoding layers across tasks while employing task-specific top layers, their framework demonstrated state-of-the-art results on ten NLU tasks. This method laid the foundation for leveraging shared representations to improve accuracy and efficiency across diverse NLP tasks.

2.2 Projected Attention Layers (PALs)

Projected Attention Layers (PALs), as conceptualized by Stickland and Murray (2019), presented a pivotal enhancement to the BERT architecture specifically tailored for multitask learning. The integration of PALs into the BERT framework is motivated by the need to improve the model's efficiency in handling multiple tasks concurrently without a significant increase in parameter count. This is achieved by introducing low-dimensional, task-specific multi-head attention mechanisms that operate in parallel to the conventional attention mechanisms within BERT. This enables the creation of a task-specific global attention layer without exponentially increasing the total number of parameters across tasks, as the majority of parameters are shared within the standard multi-head attention layer.

2.3 Training Scheduling

Scheduling tasks for training is a challenge of multitask learning, and traditional methods for task selection such as round-robin sampling (tasks are addressed sequentially, one after another in fixed order) can be problematic when the dataset for one task is larger than another. In such cases, the model may cycle through all instances of the smaller task multiple times before completing a single pass through the larger task's dataset. This discrepancy can result in overfitting to tasks with more

frequent repetitions and underfitting to tasks with fewer repetitions. Stickland and Murray (2019) proposed a proportional task sampling method for training, which initially emphasizes tasks based on their dataset size and gradually transitions to a more uniform sampling strategy. This method aims to balance the learning process across tasks, mitigating the risks of overfitting on smaller tasks and underfitting on larger ones.

3 Approach

3.1 Base BERT Implementation

Our foundational base model comprises a stack of 12 transformer layers, with each layer incorporating the self-attention mechanism of transformers. A BERT layer, denoted as $BL(h)$, is defined by the following transformation:

$$BL(h) = LN(h + SA(h))$$

where LN represents layer normalization and SA is the self-attention function. The self-attention mechanism within each Bert-Layer, $SA(h)$, is further denoted as:

$$SA(h) = FFN(LN(h + MH(h)))$$

where FFN is the feed-forward network and MH is the multi-head attention component, which enables the model to focus on different parts of the input sequence when predicting each word.

3.2 Baseline Multitask Classifier

Our baseline multitask BERT model utilized our base minBERT model, leveraging its pretrained weights as the foundation for feature extraction across all tasks. The model architecture was extended with three task-specific heads, each comprising a linear layer. These heads were tailored to the output requirements of the respective tasks: a 5-class output for sentiment analysis, a single logit for paraphrase detection (interpreted with a sigmoid function for binary classification), and a single logit for semantic textual similarity (interpreted as a continuous similarity score). The BERT model configuration was set to a hidden size of 768, divided across 12 attention heads (resulting in an attention head size of 64). We employed dropout with a probability of 0.3 to mitigate overfitting. For sentiment classification, we employed cross-entropy loss, which quantifies the difference between the predicted probability distributions and the one-hot encoded target vectors. For paraphrase detection, we leveraged binary cross-entropy loss, which evaluates the same discrepancy but for binary labels (paraphrase or non-paraphrase). For semantic textual similarity, we used MSE loss since the predicted and target logit represented a continuous score for similarity.

Our model has a "pretrain" and "fine-tune" mode; for "pretrain", the BERT parameters are frozen and only the task-specific heads are updated during training while in "fine-tune" mode, all model parameters including those of the base BERT model are updated based on task-specific data. For this baseline model, we used a batch size of 8 for all tasks, and trained for a total of 10 epochs.

3.3 Proportional Task Sampling

For an extension to our baseline multitask classifier, we implemented a proportional task sampling algorithm to learn more robust parameters generalizable across tasks. For each epoch, the baseline model iterates through the three tasks sequentially and tunes the base parameters and specific linear layer with all the task's training examples before moving on to the next task (without revisiting previous tasks). We realized this may lead to the model overfitting on each task and then "forgetting" everything it learns when moving on to the next task, so we wanted to update our training algorithm to interleave examples from all three datasets during the finetuning process.

The algorithm we ultimately implemented for training our model was inspired by the method for scheduling training introduced in Stickland and Murray (2019). Due to the imbalanced nature of the datasets for the 3 tasks (i.e. the dataset for the paraphrase detection task was $\sim$20 times the size of the datasets for the other two tasks) we used a proportional sampling algorithm to select which task examples to train on and their order. Concretely, at each training step, we randomly sample task t with a probability of p_t and train our model on the next batch of examples from task t's training dataset. The sampling distribution is determined from each task's original training data size (N_t) and

how far into training the step is. Specifically, at each epoch e (with E as the total number of epochs), we use Stickland and Murray's formula from their 'annealed sampling' method to compute α:

$$\alpha = 1 - 0.8\frac{e-1}{E-1},$$

and set each task's probability p_t proportional to their original training data size raised to the power of α (N_t^α). This proportional task sampling algorithm lets us see more examples from tasks with larger data sizes at the beginning of training and then gradually trains on the tasks more equally as training progresses.

3.4 Projected Attention Layers

To integrate Projected Attention Layers (PALs) into our Multitask BERT framework, we leveraged the approach of enabling low-dimensional, task-specific attention mechanisms in parallel with the shared BERT layers, as well as introducing parameters to the 'top' of the model, i.e. just before the classification layer.

3.4.1 Parallel PALs

We modified the BERT function to add task-specific parameters to each layer of the BERT model. Specifically, we added a task-specific function in parallel with each BERT layer of the following form:

$$h^{l+1} = LN(h^l + SA(h^l) + TS(h^l))$$

with l indexing the layer. For task specific attention TS, we implement functions of the following form:

$$TS(h) = V^D g(V^E h)$$

consisting of an encoder matrix V^E and a decoder matrix V^D, with an intermediary non-linear transformation $g(\cdot)$ which we determine to be multi-head attention, with shared V^E and V^D across layers. The dimensions of V^E and V^D are determined by d_s and d_m, where $d_s = 204$ is the lower-dimensional space (as recommended in Stickland and Murray (2019)), and $d_m = 768$ is the dimensionality of BERT's hidden layers. V^E is a linear layer that projects the BERT hidden states from d_m to d_s dimensions, and V^D is a linear layer that projects the transformed hidden states back to the original dimensionality d_m.

3.4.2 Sequential PALs

In integrating PALs at the top of our model, just before the classification layer, we similarly leverage task specific functions of the form $TS(h) = V^D g(V^E h)$, with the difference that we only leverage encoder and decoder matrices V^E and V^D once per task (as opposed to at each layer). We obtain the final hidden state for the $[CLS]$ token from our minBERT model, and pass this through the PAL for the task, which encodes the output, applies multihead attention on the reduced dimensionality, and projects back up. We then pass this transformed output to the classification layer.

4 Experiments

4.1 Data

In training our basic BERT model with a single task of sentiment classification, we used two datasets compiled from various movie reviews: the Stanford Sentiment Treebank (SST) dataset, containing 11,855 single sentences with 215,154 unique phrases after parsing through the Stanford parser, and the CFIMDB dataset, consisting of 2,434 movie reviews. For the multitask classification portion of our methodology, we used the aforementioned SST dataset for sentiment analysis, a Quora dataset to test our model's paraphrase detection abilities, and the SemEval STS Benchmark dataset for judging semantic textual similarity. We ran our model on a subset of 202,152 examples from the full Quora dataset and applied 8,628 examples from the SemEval dataset. For preprocessing of the data, we implemented tokenization to split the original input sentences into word pieces (with a predefined set of 30K different tokens) to feed into the rest of the model.

4.2 Evaluation method

As we trained our model, we evaluated its performance on both training and development datasets. We used accuracy to evaluate the sentiment analysis and paraphrase detection tasks, finding the percentage of correct predictions when the highest predicted probability class is compared to the actual class label. For semantic textual similarity, we use Pearson correlation for evaluation, determining the linear correlation between our model's predicted similarity scores with the target scores. Additionally, we monitored the training loss to evaluate the model's learning progression over time and measure the model's efficiency in minimizing the loss function. For each epoch of training, we recorded the model's performance on each task's training and dev sets and calculated the overall loss and weighted accuracy/correlation score.

4.3 Experimental details

We conducted a total of 12 experiments with our model to test different variations of hyperparameters, model architecture, and training algorithms. Through our experiments, we primarily wanted to determine how our extension features (i.e. task sampling and PALs) affected the overall model performance, as well as observe how the accuracy and efficiency of the models changed with different hyperparameter configurations (e.g. learning rate).

To obtain our baseline statistics, we first ran two trials with our baseline multitask BERT model, training the model in "pretrain" mode (where we kept the base BERT parameters frozen) and then "finetune" mode with 10 epochs each and a learning rate of 1e-5. For our extension, we first conducted two experiments by adjusting the number of steps per epoch – 3000 versus 6000 steps – for our multitask model with the proportional task sampling algorithm (without PALs) while keeping all other configurations the same: "finetune" mode, 10 epochs, and a learning rate of 1e-5. For the rest of our experiments, we maintained the setting of 3000 steps per epoch for 10 epochs on "fine-tune" mode and only adjusted learning rates, running our model with task sampling and learning rates of 1e-3, 1e-5, and 1e-6, our model with PALs added on top of the base BERT (sequential PALs) with learning rates of 1e-5, 1e-6, and 1e-7, and finally our model with PALs added within the base BERT model (parallel PALs) with learning rates of 1e-4, 1e-5, and 1e-6.

All models were trained on a single NVIDIA Tesla P4 GPU through Google Cloud Platform.

4.4 Results

As described in section 4.3, we conducted a total of 12 experiments and we recorded the best overall dev score and individual task performances achieved for each trial. The table of our highest performance results on the dev set for each experiment trial (Table 1) and plots of each experiment's dev accuracy and correlation per training epoch (Figure 1) are provided.

Experiment Setup	Avg Dev Acc	Sentiment Acc	Para Acc	STS Corr
Pretrain Baseline	0.514	0.305	0.651	0.171
Finetuned Baseline	0.633	0.450	0.766	0.366
Task Sampling 3k steps, lr: 1e-3	0.462	0.262	0.625	nan
Task Sampling 3k steps, lr: 1e-5	**0.651**	0.510	0.759	0.368
Task Sampling 6k steps, lr: 1e-5	**0.653**	0.512	0.770	0.353
Task Sampling 3k steps, lr: 1e-6	0.637	0.489	0.747	0.348
Sequential PALs, lr: 1e-5	0.555	0.253	0.731	0.358
Sequential PALs, lr: 1e-6	0.499	0.253	0.733	0.022
Sequential PALs, lr: 1e-7	0.461	0.253	0.625	0.010
Parallel PALs, lr: 1e-4	0.476	0.253	0.625	0.100
Parallel PALs, lr: 1e-5	**0.664**	0.512	0.778	0.402
Parallel PALs, lr: 1e-6	0.633	0.479	0.750	0.343

Table 1: Summary of Best Development Accuracies and Correlations

From Table 1, we see that, in general, replacing our training schedule with proportional task sampling and using PALs in parallel with the base BERT layers helped improve the overall dev score from the fine-tuned baseline model. On the other hand, the sequential PALs trials only beat the "pretrain"

baseline model but fell short of the rest of the experiment trials. Further analysis and potential explanations for these results are presented in section 5.

Based on these results on each task's dev sets, we chose to submit the predictions from our top three highest scoring experiments: "Task Sampling 3k steps, lr: 1e-5", "Task Sampling 6k steps, lr: 1e-5", and "Parallel PALs, lr: 1e-5". The test predictions ended up having the highest performance on the test set, achieving an SST accuracy of 0.526, Paraphrase accuracy of 0.761, and STS test correlation of 0.338, with an overall test score of 0.652.

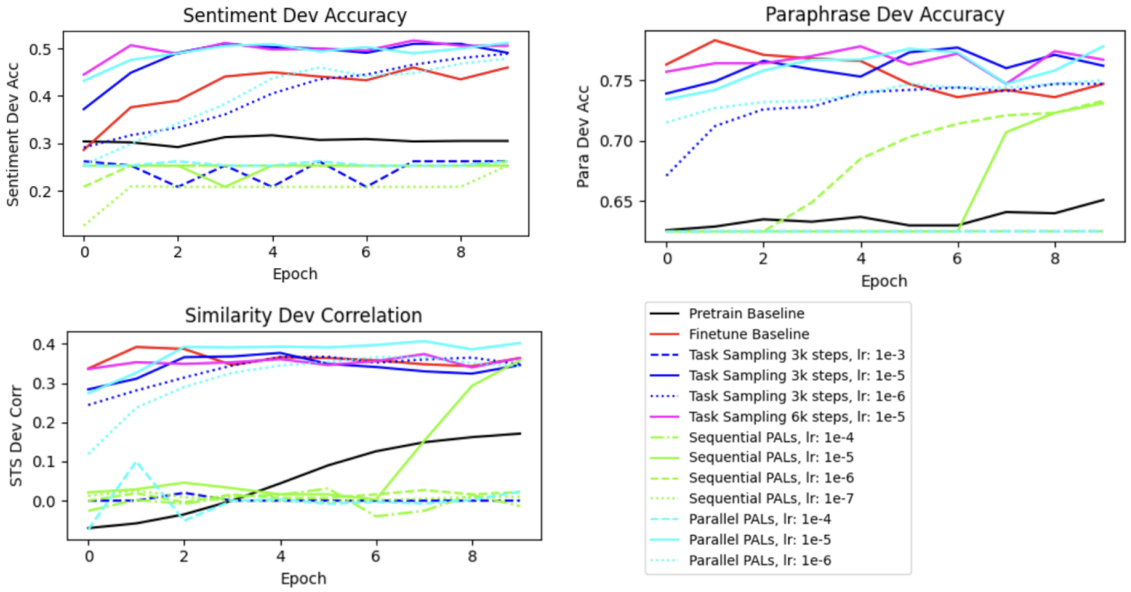

Figure 1: Performance metrics over time for each experiment

5 Analysis

For the most part, our experiment results were within our expectations and we were pleased to see a few experiments with our model extensions achieve improved multitask performance compared to the "fine-tuned" baseline classifier. Out of our three extension approaches, implementing the task sampling training algorithm and using the PALs layer in parallel with the BERT layers yielded the best results, while the Sequential PALs approach resulted in worse performance for the classifier.

We expected the proportional task sampling to improve overall performance, even if we maintained the model architecture of our baseline Multitask BERT, which is reflected in our results. The improvement in dev scores is attributed to an increase in accuracy specifically for the sentiment analysis and paraphrase detection tasks. As shown in the first plot in Figure 1, the models that used proportional task sampling (without PALs) with a learning rate of 1e-5 consistently achieved higher sentiment dev accuracies than the baseline model for all 10 epochs. These results confirm that task sampling helped reduce interference, where the model would "forget" its stored knowledge on previous tasks when training on one task for many steps. Since the baseline model iterated through the entire training set for each task sequentially, the examples from the last task (STS dataset) would interfere with what the model learned from the first two tasks (SST and Quora dataset). Thus, by sampling tasks at each training step with probabilities proportional to the training epoch, we minimize this interference between tasks (tasks are trained more equally towards the end), while also ensuring that datasize imbalances between tasks (i.e. model trains on more examples from tasks with larger datasets in the beginning) are accounted for.

Both Parallel PALs and Sequential PALs incorporated the new task sampling algorithm, but had opposing results. IN accordance with Stickland and Murray's findings, we expected the Parallel PALs model architecture to yield the best performance. Our experiments met those expectations, with the

Parallel PALs model trained upon a learning rate of 1e-5 showing the best average performance on the dev sets for the three tasks. This shows that the task specific attention layers introduced with the PALs framework succeeded in helping the model learn more robust information for multiple tasks, even while sharing a single base BERT model. Furthermore, the enhanced representation of the data with PALs did not result in a significant increase in model complexity, as the training time stayed relatively constant across experiments. Our results with implementing PALs on top of the base BERT model, however, did not meet our expectations. We expect Sequential PALs to improve computational efficiency and yield results similar to the baseline when the objective is to address separate tasks using the same input data. Since this project involved learning from different sentence embeddings specific to each task, we didn't expect our results with sequential PALs to necessarily yield improved accuracy, but we did think it would match up with the baseline. Our results showed that even the best trial for this approach (sequential PALs architecture trained on a learning rate of 1e-5) showed a lower dev accuracy than the fine-tuned baseline. A potential explanation for this could be that the increased model complexity with added layers of PALs on top of the BERT model required increasing training to reach optimal results. We did notice an upwards trend for Sequential PALs (specifically with learning rate 1e-5) for the three tasks near the end of training, shown in the plots in Figure 1. It is thus possible that the sequential PALs model could have continued to show an improvement in performance given more time to train, i.e. 15 or 20 epochs. Further experimentation would be needed to confirm this explanation.

Our experiments also showed varying results within each extension approach as we adjusted hyperparameters such as learning rate and the number of steps per epoch. For the task sampling trials where we tested out an increase in the number of steps per epoch from 3k to 6k, we observed that the 6k trial showed higher performance in the first few epochs but overall performance was similar in the end. This matched our expectations since increasing the steps per epoch would have given the model more time to improve in the beginning, but should not impact the optimal results given a sufficient number of total epochs. As for experimenting with learning rates across tasks, we found that a rate of 1e-5 was optimal, as any lower (i.e. 1e-7) would make training much slower and potentially cause it to become stuck at a local minima, while any higher (i.e. 1e-3) opens up the potential to overshoot the minima and lead to unstable training.

6 Conclusion

In this project, we explored the potential for multitask learning in NLP, leveraging a BERT-based architecture to address three linguistically nuanced tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. We have seen firsthand how scheduling tasks with differing dataset sizes for training poses inherent issues of overfitting to certain tasks, and adopting a proportional task sampling approach improved our model's performance across all experiments. In integrating PALs, our main takeaway is that integrating task specific attention mechanisms in parallel with the original layers of BERT, allowing for task-specific contextual processing, does enhance the model's ability to tailor its attention mechanism to the specific requirements of each task. We believe that given more time, we would have been able to run a myriad of additional experiments to systematically determine a more optimal learning rate and training epochs count, as our analysis showed potential for our Parallel PALs model's accuracy to continue improving beyond the 10th epoch. For our Sequential PALs implementation, we look forward to testing our model on a dataset in which it makes more sense to apply the three separate tasks to the same sentence embeddings. In these settings, it is clear through our multitasking architecture that leveraging shared parameters within task-specific attention mechanisms enables our model's enhanced efficiency.

7 Team contributions

We contributed to all parts of the coding and writeup for this final project. However, Bradley primarily led the related works exploration, minBERT implementation, and PALs extensions, while Shannon led the multitask classification implementation, task sampling extension, and experimentation.

References

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*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

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