BERT: A Master of All Trades or Jack of None? - Design Document

Stanford CS224N Default Project

Tom Pritsky
Department of Biomedical Data Science
Stanford University
`tom5@stanford.edu`

Josselin Somerville
Department of Computer Science
Stanford University
`josselin@stanford.edu`

Marie Huynh
Department of Biomedical Data Science
Stanford University
`mahuynh@stanford.edu`

1 Overview

We aim to leverage multi-task learning to reduce the number of required model parameters while maintaining high performance across multiple tasks. This goal is critical for building lightweight language models intended for mobile and wearable devices with limited computing, battery, memory, and processing. Instead of storing multiple models, we can store a single multi-task model that addresses multiple tasks. In this project, we aim to set up a multi-task model that can perform three separate linguistic tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Training and evaluating on respectively the Stanford Sentiment Treebank (SST), the Quora Dataset and the SemEval Benchmark Dataset, we will build a baseline BERT-based multitask model and will evaluate multiple published fine-tuning approaches (including training scheduling, gradient surgery, etc.) on a our baseline.

2 Background

In NLP, models often have a few hundred million trainable parameters, which makes "adaptations to new tasks computationally infeasible" as underlines this paper [1]. Multi-task learning can help us create better language representations, improve generalization of models and spare resources. In our case, PALs [2] is a novel approach to multi-task learning that uses projected attention layers to share information between tasks while efficiently adapting to new tasks. In effect, they introduce a technique for adapting pre-trained models to new tasks that requires fewer parameters than traditional fine-tuning approaches. This more efficient method could allow a model to quickly adapt to new tasks while retaining the learned representations from pre-training. Furthermore, they can help us build better representations of language by providing an effective mechanism for sharing information between related NLP tasks and improving a model's abilities to capture different features from the input data. In addition to achieving state-of-the-art results on various benchmarks, this method showed promise for being more broadly applicable to other tasks–beyond the ones it was tested on. Such applications include various fields in NLP, Speech Recognition and Computer Vision as well. The notion of "Residual adapter modules" that inspired this paper was itself inspired by Rebuffi et al in Computer vision [3], adapting large pre-trained residual networks for multi-task learning.

3 Goals

We aim to set up a multi-task model that can perform three separate linguistic tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Specifically, the inputs to the model will be one or two input sentences (depending on the task) and the task label. The output will be a single categorical value, where the possible categories depend on the chosen task.

4 Non-Goals

We do NOT aim to optimize sequentially three different models and do a final ensemble model.

5 Future Goals

Future improvements include training larger PAL layers from scratch, adding more-complex architectures to our model (like more complex layers and potentially a bigger BERT model which was not the goal for this project). Since we will run many experiments, it would be nice to implement the finetuning of the hyperparameters once we are satisfied with the performance of a combination of features for our model. Additionally, we could implement specific approaches that are known to perform well on our datasets, such as Heisen routing for SST [**?**].

6 Detailed Design

6.1 Data

We will be using the provided datasets for the default project with the following splits :

Datasets	Labels	Size (Train, Dev, Test split)	Task
Quora Dataset	Question pairs with labels indicating whether particular instances are paraphrases of one another	400,000 question pairs: • 141,506 / 20,215 / 40,431	Paraphrase detection
SemEval STS Benchmark Dataset	Sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning)	8,628 different sentences: • 6,041 / 864 / 1,726	Sentence similarity
Stanford Sentiment Treebank (SST) dataset	Unique phrases extracted from movie reviews with a label of negative, somewhat negative, neutral, somewhat positive, or positive	11,855 sentences: • 8,544 / 1,101 / 2,210	Sentiment analysis

Table 1: Datasets (Default project)

6.2 Baseline

As a baseline, we will implement a simple classifier consisting of a BERT model that has on top one dropout and a linear layer trained for each task. Sentiment classification and paraphrase detection are both considered as classification, whereas semantic similarity is handled as a regression task. Our model will be first pre-trained (only the classification layers are trained) and then finetuned (all the parameters, including the BERT layers, are trained) using a random task scheduler, meaning that at each step of the training, the task used for training is sampled randomly.

6.3 Data Preprocessing and Augmentation

We will filter out the top (2%) of the longest inputs: the longest sentences for the sentiment analysis and the longest sums of sentence pairs for the paraphrase and the similarity since we will be using this concatenation in our model.

Furthermore, we observed class imbalance in the training sets of the SST dataset (Sentiment) and the SemEval dataset (Similarity). Thus, we want to generate an augmented dataset using Easy Data Augmentation (ESA) [4]. For each underrepresented class in the Sentiment and Paraphrase training sets, we augment them to be as present as the most represented class.

For each training sentence to augment, we will perform the following 4 simple operations. During synonym replacement, we randomly choose some words–excluding stop words–from the sentence and replace each of those with one of its synonyms chosen at random. During random insertion, we find and insert a synonym of a random word–excluding stop words–into an arbitrary position in the sentence. During the random swap, we randomly choose two words in the sentence and swap their positions. During random deletion, we randomly remove words with probability p. We found a repository on Github that implements the latter (source : https://github.com/jasonwei20/eda_nlp).

6.4 Memory optimization

As our model (Hugging Face bert-base-uncased) is very large, our GPU is likely to not handle a very big batch size. This may impact the training : it may be slow and the updates may be very irregular as the gradients will be averaged over a small batch. To overcome this issue, we will implement two optimizations: Automatic Mixed Precision and Gradient Accumulations. The former will enable us to use nearly half the memory by using half-precision, while the latter will only updats the model every x batches which will allow us to simulate larger batches and accelerate the training process.

6.5 Task Scheduling

One main challenge of this project is to make sure that training the model for a task does not ruin the accuracy of another task. This becomes even more challenging when the sizes of the datasets for each task are very different as the task with the largest dataset will need more time to learn. With traditional random scheduling (and the similar round robin scheduling), we face two types of challenges: with a small amount of training, tasks with large datasets will be underexplored, and with a large amount of training, tasks with small datasets will overfit.

This is why we will implement the task scheduling proposed in the Projected Attention Layer article[2]. This consist of randomly selecting a task with a probability proportional to: N_i^α where N_i is the size of the dataset of task i and $\alpha = 1 - 0.8\frac{e-1}{E-1}$ with e the current epoch and E the total number of epochs.

6.6 Finetuning with gradient treatment methods and SMART regularization optimization

Another problem in multitask learning is that the gradients of one task can oppose the gradients for another task. This is why we want to explore Gradient Surgery. Gradient surgery consists of performing all tasks, then projecting the gradient of each task on the normal planes of the other gradients before doing an update. We will use the implementation from this repository: https://github.com/anzeyimana/Pytorch-PCGrad-GradVac-AMP-GradAccum. In addition, we will also try gradient surgery as proposed by Finn et al [5]. This technique projects competing gradients from one task onto the normal plane of another task, preventing the competing gradient components from being applied to the network and impairing optimization. In our implementation, we will identify whether multiple task gradients were conflicting (by checking if their cosine similarity was negative), remove the conflicting components of the gradients, and pass these updated gradients to our optimizer of choice.

Finally, aggressive fine-tuning often causes over-fitting and thus failure to generalize to unseen data. The SMART fine-tuning framework developed by [6] aims to improve the latter in two steps. The first step is Smoothness-Inducing Adversarial Regularization, which reduces overfitting and improves generalization by adding a smoothness inducing adversarial regularizer to the loss function. The second step is Bregman proximal point optimization, which acts as a strong regularizer to prevent aggressive parameter updates during fine-tuning. This step improves the stability and convergence of the training process by preventing the updates from deviating too heavily from the previous ones. The SMART framework has been shown to improve the performance of fine-tuning on various NLP tasks with limited data.

6.7 Individual Pretraining

During pretaining, the parameters of BERT are frozen, so the three classifiers are independent. For this reason, we want to try **individual pretraining which consists of training our multitask classifier on each task with frozen BERT layers, and then loading the 3 best sets of parameters for each fully connected layer used in the classifier**. This could improve our accuracies later.

6.8 Enhancing the BERT model with PALs

Finally, we want to try to implement the Projected Attention Layer (PAL). It consists of adding to each Bert Layer a very low-rank task-specific attention mechanism in addition to the task-common large attention mechanism.

6.9 Evaluation method

For the classification tasks (sentiment analysis and paraphrase detection) we will use a simple accuracy metric, calculated by dividing percent correctly classified examples by total examples. For the regression task of semantic textual similarity, we will use Pearson correlation of the true similarity values against the predicted similarity values for the SemEval STS Benchmark Dataset.

7 Environment

We will have a shell file **set_up.sh** that can be run with the command **source set_up.sh** that will install a conda environment **cs224n_dfp** with all needed requirements.

8 Work Estimate

This project will take a lot of time. We plan on spending between 20 hours per person per week (for about 4 weeks) for building the baseline, implementing the extensions we mentioned and running various experiments.

9 Alternative Approaches and Related Works

As highlighted in [7], "the existing methods of MTL have often been partitioned into two groups with a familiar dichotomy: hard parameter sharing vs. soft parameter sharing". On one hand, in hard parameter sharing, you share the model weights between multiple tasks and each weight is trained to jointly minimize multiple loss functions [7]. On the other hand, in soft parameter sharing, each task has its model with its individual weights and we penalize the distance between the model parameters of the different tasks in order for the parameters to be similar.

In recent years, BERT (Bidirectional Encoder Representations from Transformers) [8] has become a popular choice for multitasking in Natural Language Processing (NLP) due to its strong performance on various NLP tasks. Inspired by [2], we assume soft-parameter sharing with the whole of BERT would require too many parameters. In this context, we will consider hard parameter sharing while adding adapters to shared layers and keeping several task-specific output layers with the objective of building a multitask BERT on three separate linguistic tasks : sentiment analysis, paraphrase detection and semantic similarity.

10 Designation of Work

1. We will build the baseline all together while pair-programming during three sessions. We will focus each on some extensions.
2. **Josselin** : Gradient Treatment Methods, PAL scheduling, variation of PAL layers (Seems complicated), Running Experiments
3. **Marie** : SMART regularization, Data Augmentation for Class Imbalance, Data Preprocessing, Running Experiments
4. **Tom** : Hyperparameter Tuning, Building Bigger Models, Running Experiments

References

[1] Łukasz Maziarka and Tomasz Danel. Multitask learning using bert with task-embedded attention. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2021.

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

[3] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[4] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China, November 2019. Association for Computational Linguistics.

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

[6] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, pages 2177–2190, Online, July 2020. Association for Computational Linguistics.

[7] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

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