Multi-task NLP with BERT

Stanford CS224N Default Project

Chris King
Department of Computer Science
Stanford University
kingces@stanford.edu

Abstract

We extend BERT to preform multiple Natural Language Processing (NLP) tasks. The tasks are (1) multi-class sentence classification task in the form of sentiment analysis of the Stanford Sentiment Treebank dataset of movie reviews, (2) multi-class sentence pair classification task in the form of predicting the similarity between sentences of the SemEval STS Benchmark dataset, and (3) a binary-class sentence pair classification task in the form of de-duplicating Quora questions.

The original model is provided by Stanford CS224N teaching staff. Our best preforming model changed that base model to instead operate as suggested in the BERT paper whereby a linear layer (plus a task discriminating feature) is placed a top the BERT model which classifies the [CLS] token generated by feeding in a single or multiple sentences. We attempted gradient surgery (4), task embedding (our own novelty), and task attention embedding (3) but in the end all models preformed more or less the same although their time to best result differed.

1 Approach

We seek to maximize the average score of three NLP tasks run against a model we extend from BERT. We refer to the three tasks as sentiment, semantic, and paraphrase where the first is an instance of sentence classification and the latter two are instance of sentence pair classification task. For detailed descriptions of each of the three tasks we refer the reader to the Stanford CS224N 2022 Winter Quarter Default Project Handout[1].

Our goal is to maximize the average score on the three NLP tasks with minimal additional parameters. We restrict ourselves to models without task specific parameters. To that end, we abstract the three tasks into a meta-task plus discriminator and use task embedding, gradient surgery, task discrimination features to enable the learning machinery to make more intelligent gradient updates.

2 Experiments

We present five model named Naive, Baseline, Task Embedding, Task Attention Embedding, and PC Grad (gradient surgery). We continue to run epochs until our score stops increasing or to roughly a dozen epochs. We use default hyper-parameters for our runs unless otherwise noted. We summarize the experiments, our sampling methodology, our results in tables, and conclude this section with more a detailed description of our five experiments when experiment summaries do not suffice.

1. **Naive** We conjecture the default implementation classification approach of using different linear layers to classify each task should be replaced with a single linear layer that accepts a discriminator feature.

[1] https://web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf

2. **Baseline** We conjecture the default implementation sentence pair classification approach of generating two CLS encodings, one for each sentence via separate forward passes through BERT, should be replaced with classifying a single CLS token generated by a single forward pass through BERT of the SEP joined sentences.

3. **Task Embedding** We conjecture adding a task embedding ala the position embeddings to BERT will allow the learning machinery to intelligently separate BERT parameter gradient updates for each task into different dimensions of a shared set of parameters.

4. **Id** We conjecture pre-pending each example with a task id (followed by a space) will allow the attention mechanism to pay attention to task type. Where Task Embedding provided a task specific signal by modifying the model, this experiment would achieve a similar effect by modifying the data.

5. **Task Attention Embedding** We conjecture adding task attention embeddings as described in (3) will similarly allow the learning machinery to intelligently separate BERT parameter gradient updates for each task into different dimensions of a shared set of parameters. Essentially each BERT layer learns an embedding per key, value, query per task shared across heads. Each embedding is of size equal to the corresponding key, value, or query matrices. The embedding is added to the linear projection of the key, value, and query.

6. **Gradient Surgery** We conjecture that gradients for different tasks are more likely point in opposite directions and so projecting them onto each other as described in (4) would reduce the tension between task updates.

7. **Ensemble** We conjectured an ensemble of these approaches would produce the best result. For the ensemble, we modified Gradient Surgery to *not* preform updates on the Task Embeddings or Task Attention Embeddings since these parameters were specific to each task.

2.1 Sampling

For the milestone experiments (Table 1), samples are drawn randomly from each dataset without replacement until each dataset is exhausted. For the remaining experiments (Table 2 onward), 10k, 20k, and 40k samples are randomly drawn without replacement from the union of the three training sets. If a dataset is exhausted, then we recycle the data set (Looped Sampling) and being again drawing random samples from that dataset without replacement. Samples from different datasets are randomly interleaved for all experiments except when PCGrad is enabled. When PCGrad is enabled the datasets are randomly sampled without replacement in a round-robbin fashion so that each batch is gaurenteed to have equal examples of each task. We also experiment with drawing uniformly by labels within each dataset (Looped Balanced Sampling) which has the effect of hiding the prior label distribution.

2.2 Batch

For the milestone experiments we used the default batch size of 8. For the remaining experiments we used a batch size of 9 except for Gradient Surgery where we used a batch size of 3. Using a batch size of 3 with round robbing selection of datasets ensured each batch contained a single example from each dataset which we conjectured would allow for maximum opportunity to correct gradient contention before moving to far in any one direction.

2.3 Results

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Naive	0.490	3	0.466	0.698	0.305
Baseline	0.711	6	0.503	0.823	0.806
Task Embedding	0.715	7	0.505	0.809	0.831

Table 1: Milestone Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Baseline	0.716	3	0.506	0.818	0.823
Id	0.719	10	0.519	0.826	0.813
Task Embed	0.719	8	0.512	0.826	0.818
Task Atten Enbed	0.718	3	0.494	0.820	**0.840**
Gradient Surgery	0.712	2	0.514	0.827	0.821

Table 2: 10k Looped Samples Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Baseline	0.724	2	0.513	0.834	0.825
Id	0.724	7	0.509	0.841	0.823
Task Embed	0.724	6	0.510	0.840	0.822
Task Atten Enbed	0.722	8	0.518	0.840	0.808
Gradient Surgery	0.717	2	0.506	0.829	0.816

Table 3: 20k Looped Samples Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Baseline	**0.731**	11	0.518	**0.869**	0.807
Id	0.730	7	**0.520**	0.857	0.814
Task Embed	0.727	11	0.500	0.864	0.817
Task Atten Enbed	0.726	6	0.500	0.855	0.823

Table 4: 20k Looped Balanced Samples Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Baseline	07.21	2	0.506	0.859	0.797
Task Embed	0.727	11	0.500	0.864	0.817
Task Atten Enbed	0.723	4	0.490	0.868	0.811

Table 5: 40k Looped Balanced Samples Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
Baseline 03e-6	0.730	7	0.520	0.857	0.814
Baseline 07e-6	0.727	7	0.514	0.857	0.812
Baseline 0e-5	**0.731**	11	0.518	**0.869**	0.807
Baseline 13e-6	0.711	7	0.490	0.834	0.810
Baseline 17e-6	0.712	1	0.502	0.828	0.805

Table 6: 20k Looped Balanced Samples Hyperparameter Search Experiment Summary

Name	Score	Epochs	Sentiment	Paraphrase	Semantic
All experiments	0.714	10	0.486	0.832	0.823

Table 7: 20k Looped Balanced Samples Search Experiment Summary

2.4 Naive

We establish a naive baseline performance using a model requiring minimal changes to the default model provided by course staff while still allowing each dataset to non-trivially contribute to fine-tuning of the BERT parameters. To this end, we (1) abstract the sentence classification task as a sentence *pair* classification task by duplicating the sentence, and (2) use the union of the labels of the three sentence pair classification tasks as the set of classifications.

Fine-tuning our model proceeds by generating pool vectors for each sentence $A \in \mathbb{N}^h$ and $B \in \mathbb{N}^h$ for the sentence pair classification dataset. Each pool embedding is constructed as specified in the Default Project Handout (e.g. it corresponds to the first input token ([CLS]) passed through a single layer neural network with Tanh activation). We next construct a task vector $C \in \mathbb{R}^{2d+1}$ by concatenating the pool embedding A and B and a discriminator $D \in \mathbb{R}$. The discriminator for each of the three original tasks is a unique integer. We pass the task vector C through a linear layer $W \in \mathbb{R}^{k \times 2d+1}$ where k is the number of labels. Finally, we compute a standard classification loss i.e. $\texttt{log}(\texttt{softmax}(CW^T))$ and back-propagate.

2.5 Baseline

We join the sentences in the sentence pair classification tasks with a SEP token and run our classification on the resulting CLS token. The BERT paper directs sentence pair classification proceed in this manner and, indeed, it results in a significant improvement in performance over the naive approach.

2.6 Id, Task Embedding, Task Attention Embedding, and Gradient Surgery

None of these approaches reliably improved performance. Differences are attributed to noise.

1. **Id** While this improved performance in the early epochs the performance returned to baseline. From this we assume the model quickly learned to differentiate between tasks but that simply knowing what task is being evaluated is not sufficient to improve performance.
2. **Task Embedding** Since the position enbedding signal must travel to the top of the BERT stack to be used by the attention layer there, we assumed so too must an task embedding travel to the top of the stack and so could be used everywhere to make more intelligent gradient updates. Ultimately we could detect no reliable improvement in performance.
3. **Task Attention Embedding** This experiment accounted for our largest semantic score when run against 10k samples, but our lowest score when run against 20k samples and average scores for other sampling cohorts. Again, we could detect no reliable improvement in performance.
4. **Gradient Surgery** We made use of code provided by the paper to implement gradient surgery[2].

3 Conclusion

Our best model added a single linear layer with a single task discrimination feature atop BERT to classify the [CLS] token for three different tasks fed to BERT as prescribed in their paper (we used the [SEP] token). We trained our model by drawing 40k samples from a weighted random distribution of the union of all datasets. The weights ensured the labels per dataset were sampled at a uniform rate. This straightforward model preformed in the top 30/110 on a single backbone and **14th**/110[3] when trained on individual datasets.

4 Future work

In the future we would

1. work to increase our semantic score alone. Our baseline achieves scores of 0.530, 0.870, and 0.830 for sentiment, paraphrase, and semantic respectively. The first two scores are reasonably high, but the latter needs something more than pre-training the BERT parameters.
2. do as (1) suggests and pre-train on the tasks BERT originally trained with and finetune parameters layered on top of BERT; The default model presented in the default project directs student to use cosine similarity to compare sentences and makes it easy to pre-train on the downstream tasks. Instead we should do as BERT suggests which is pre-train on the Masked LM task and use the [SEP] token to compare sentences.

[2] `https://github.com/WeiChengTseng/Pytorch-PCGrad/blob/e987ac603fa1accd386820a985a6dc2fd92dec5b/pcgrad.py`

[3] as of this writing

3. run layer abatement experiments guaranteed to produce performance gradients. BERT essentially ignored the changes we made to its model resulting in no performance gradient hence no direction to focus efforts. Denying a task an increasing number of layers is guaranteed to eventually produce a change in performance[4]. Using those results we could allocate tasks a subset of layers to each task. This would allow the remaining layers to be dedicated to the remaining tasks reducing gradient update contention in a more straightforward way than gradient surgery.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805, 2018.

[2] A. Thakur, *How to Fine-Tune BERT for Text Classification*. arXiv preprint arXiv:1905.05583, 2019.

[3] Ł. Maziarka and T. Danel, *Multitask Learning Using BERT with Task-Embedded Attention*. In 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021 doi: 10.1109/IJCNN52387.2021.9533990.

[4] A. Author, B. Author, and C. Author, *Gradient Surgery for Multi-Task Learning*. NeurIPS 2020 arXiv:2001.06782.

[4] Saving all [CLS] tokens for every forward pass would allow this analysis to be done in parallel!