Task-specific attention

Enzo Jia [1]

[1]Department of Computer Science, Stanford University

Abstract

In this project I explore applications of task-specific attention mechanism, which takes BERT foundation model output and is intentionally overfitted for each task. Experiments proved that this mechanism can help to improve model performance when finetune data is relatively large. Corresponding training cost is not significantly higher compared to traditional finetunes. In this exploration, task-specific attention layers improve paraphrase detection accuracy from 0.551 to 0.842, with a 10% increase of finetune training time.

Background and motivation: BERT's document representations

BERT (Bidirectional Encoder Representations from Transformers) is designed to be a foundation model and produces one document's representation which is the self-attention block output corresponding to the first token. In theory, when number of self-attention layers is large enough, final output of the first token's attention block is representative for the whole document. Are 12 and 24 large enough? An experiment answers this question. Table 1 gives BERT model's downstream task performances under different settings.

	BERT return	pretrain / finetune	sentiment dev acc	paraphrase dev acc	STS dev corr	Train time (sec)
Part I	first token	pretrain	0.389	0.401	0.298	15006
	output	finetune	0.504	0.595	0.587	37183
Baseline	mean sentence	pretrain	0.460	0.386	0.643	15608
	output (masks excl'd)	finetune	0.515	0.561	0.854	38161

Table 1. Results on dev datasets, utilizing BERT's outputs on each sentence's first token vs. mean of the whole sentence. One T4 GPU on Google Cloud was used for each task in each experiment.

Results from three tasks are recorded in table 1: sentiment analysis, paraphrase detection, and STS (sentiment textual similarity analysis). Row "first token output" corresponds to the scenario of using BERT's 'vanilla' output which is self-attention output of the sentence's first token, which in the case of this exploration is a prepended [CLS] token. On the contrast, results in "mean sentence output" row utilizes mean of the whole sentence's all tokens' outputs, after excluding masked tokens. Here we see how the "mean sentence output" helps the STS task's performance, and this improvement indicates that attention outputs corresponding to tokens other than the first one have remaining information not being collected by the first token's output.

Instead of taking mean of the whole sentence's attention outputs, I propose arranging different weights to tokens in the sentence for aggregation – this is what attention does.

Approach: Task-specific attention

- This mechanism collects information from each token's output in a smarter way. To be specific, I use one or multiple self-attention layer(s) for sentiment analysis, one or multiple cross-attention layer(s) for paraphrase detection, and another one or multiple cross-attention layer(s) for semantic textual similarity analysis.
- To produce a sentence's representation encode, this mechanism takes the last task-specific attention layer's output corresponding to the first ([CLS]) token, which uses a softmax to distribute weights to all tokens' transformed values calculated from previous layer's outputs. What this mechanism finally get is a weighted mean of the sentence's all tokens' previous attention outputs.
- Optional: initializing attention layer weights with $\mathcal{U}(0.9, 1)$ makes the starting point of optimization close to taking a mean of the sentence's all tokens' encodes from previous layer, which is what the baseline experiment does.

Experiment settings

All experiments are controlled to use the same setting except for those listed in table 2. Controlled setting items include: AdamW optimizer, $1e-3/1e-5$ learning rate for pretrain/finetune respectively, SST/Quora/SemEval data for three tasks respectively, and Cross Entropy / Binary Cross Entropy / Negative Pearson Correlation loss functions for three tasks respectively.

	sentiment	paraphrase	STS
Baseline	no attn 3840 param	no attn 0.59M param	no attn 2.36M param
Experiment #1	2 self-attn layers no initialization 4.72M param	1 cross-attn layer no initialization 3.54M param	3 cross-attn layers no initialization 11.2M param
Experiment #2	2 self-attn layers init: $\mathcal{U}(0.9, 1)$ 4.72M param	1 cross-attn layer init: $\mathcal{U}(0.9, 1)$ 3.54M param	3 cross-attn layers init: $\mathcal{U}(0.9, 1)$ 11.2M parame

Table 2. Experiment settings. Attention here refers to task-specific attention layers, not layers in BERT. Number of parameters are also task-specific, not including BERT parameters.

Experiment results

	Processing on BERT return	pretrain / finetune / test	sentiment accuracy	paraphrase accuracy	STS correlation	Train time (sec)	Train time vs. baseline
Baseline	mean sentence output (masks excl'd)	pretrain	0.460 (dev)	0.386 (dev)	0.643 (dev)	15608	1
		finetune	0.515 (dev)	0.561 (dev)	0.854 (dev)	38161	1
		test	**0.517**	**0.551**	**0.826**		
Experiment #1	Uninitialized attention	pretrain	0.253 (dev)	0.469 (dev)	0.251 (dev)	18086	1.159
		finetune	0.509 (dev)	0.843 (dev)	0.822 (dev)	41827	1.096
		test	**0.533**	**0.842**	**0.791**		
Experiment #2	Initialized attention	pretrain	0.262 (dev)	0.429 (dev)	0.148 (dev)	17648	1.131
		finetune	0.539 (dev)	0.838 (dev)	0.386 (dev)	41036	1.075
		test	**0.524**	**0.839**	**0.374**		

Table 3. Final results on dev and test datasets. One T4 GPU on Google Cloud was used for each task in each experiment.

- Best test leaderboard submission was scored at $(0.524 + 0.842 + 1.826)/3 = 0.759$, and ranked #44 out of 135 submissions on March 18th, 2024.
- Because the three downstream tasks were trained separately, running time was recorded as the sum of three task's training times.

Topics left for future work

- Loss function exploration for sentiment analysis. Cross entropy loss sees model outputs as categorical classfication results, however for sentiment analysis the label is ordinal data, with information unused.
- More training data, including CFIMDB data for sentiment analysis.
- Different foundation models.
- Different experiment settings, including different optimizers, learning rates, etc.

Discussions

Why task-specific attention helps paraphrase detection?

- This task has a relatively large training data size. To train the million-level extra parameters for each task (table 2), Quora paraphrase data has 17688 labeled pairs of sentences, compared to SST which has 1068 labeled sentences and SemEval STS data which has 755 labeled pairs of sentences. Task-specific attention consumes data to train.
- This task has a larger amount of potential unexplored. By March 18th, each task's best score on the leaderboard are 0.557/0.9/0.891 for SST/Para/STS respectively. My baseline scores for SST and STS are $> 90\%$ of corresponding best scores, however for Para it is 61%. Task-specific attention helps to dig out unexplored potentials, if there is some.

Do we need initialization?

- For STS and Para tasks, parameter searches with and without initialization may arrive at the same local optima. For STS, initialization leads to a different local optima where the model performance is worse. Not covered by these three task cases, there could be other cases where initialization leads to a better-performing local optima.
- Initialization does not cost more time. In this exploration initialization leads to shorter search paths so saves training time by a small margin (table 3).

When resources allow, trying different initialization settings has the potential to help improve the model's performance, however when resource is not redundant, I recommend using uninitialized task-specific attention.

Other discussions

- Computational resource requirements. Compared to baseline, both experiments #1 and #2's training times increase between 7% and 16%. Addition of task-specific attentions does not increase the model's asymptotic complexity because both baseline and experiment #1/#2 settings have the same training time complexity at $O(doc\_len^2)$ for each doc. Computational resource consumption depends more on training data sizes.
- Task-specific attention layers cannot be replaced by attention layers inside foundation models, because task-specific attention is trained more precisely only on finetune data, so it's intentionally overfitted for this type of tasks.

Conclusions

Task-specific attention helps to collect information from the whole document's tokens' encodes in an efficient way, so this is a method worth being tried when building an application of a foundation model. This mechanism has a higher probability to succeed when finetune data size is large. However the author wants to emphasize that task-specific attention is a downstream mechanism on top of results by finetuning the foundation model, so the scientist or engineer who is designing the application shall first focus on upstream tasks, including foundation model selection, task loss function design, etc., and then use task-specific attentions as an option to check whether the upstream outputs have remaining information which can be captured by this machanism.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Neural Information Processing Systems*, 2017.

enzojia@stanford.edu [frequently-checked email: enzo.jia@gmail.com]