Robust Embeddings using Contrastive Learning and Multiple Negatives on BERT

Shoaib Mohammed [1]
shoaibmh@stanford.edu

Ishan Sabane [2]
ishancs@stanford.edu

[1]Department of Computer Science [2]Department of Electrical Engineering
Stanford University

Introduction

We build a multi-tasking BERT model to perform well on three downstream tasks: sentiment analysis (SST), paraphrase detection (PD), and semantic textual similarity (STS).

When to use multi-task learning?

Multi-task learning should be used when different tasks have some level of correlation. For instance, paraphrase detection and STS are similar tasks as both care about similarity between two sentences. Moreover, multitask learning allows deployment of a single model which can be used across variety of applications. This proves to be beneficial in terms of deployment and management.

Background

Efficient Natural Language Response Suggestion for Smart Reply [2]

Multiple negatives ranking loss is a great loss function to use when you only have positive pairs which is the case for paraphrase detection and STS. We have K input sentences $\mathbf{x} = (x_1, x_2, \cdots, x_K)$ and their corresponding similar sentence $\mathbf{x} = (y_1, y_2, \cdots, y_K)$. We assume that any response y_j is a negative response for x_i if $i \neq j$ given the small batch size. The approximate probability can be given by:

$$P_{\text{approx}}(y|x) = \frac{e^{S(x,y)}}{\sum_{k=1}^{K} e^{S(x,y_k)}}$$

Supervised Contrastive Learning [1]

Consider a set of paired examples $D = \{(x_i, x_i^+)\}_{i=1}^m$, where x_i and x_i^+ which are semantically related sentences. Let h_i and h_i^+ be the embedding representations of the two sentences. The sentence pairs are extend to triplets by adding negative examples. Certain datasets have negative examples which can be used directly. Hence, we convert the pair (x_i, x_i^+) to (x_i, x_i^+, x_i^-). Our implementation uses the SNLI dataset which already contains the require sentence triplets. The contrastive loss function for N sentence pairs is given as follows:

$$\mathcal{L} = -\log \frac{e^{\text{sim}(h_i, h_{i+})/\tau}}{\sum_{j=1}^{N} \left(e^{\text{sim}(h_i, h_j^+)/\tau} + e^{\text{sim}(h_i, h_j^-)/\tau}\right)}$$

Methods

We tried various methods to improve the model performance by building on top of related works provided in the background. The final model architecture is shown in 1. The three big ideas are mentioned below.

Vanilla

TODO

Round-robin selection

We tried various fixed orderings of the data such as (SST, PD, STS) or (PD, SST, PD, STS, PD) in the same epoch. It was beneficial to train PD two times as much as SST or STS.

Multi-task deep neural networks (MT-DNN) [3]

We create a deep neural network where all three tasks are trained using a shared model. This is useful in our case since the three tasks are related. Then, during each epoch we randomly sample which task to train on using different loss functions such as multiple negatives ranking.

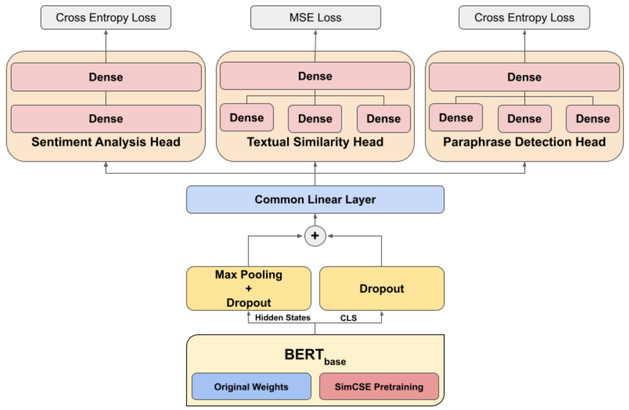

Figure 1. Model architecture

Results

Exp	Model	SST		STS		QQP		Avg
		Train	Dev	Train	Dev	Train	Dev	
1.0	Vanilla BERT	0.393	0.388	0.008	-0.009	0.235	0.38	0.253
2.1	Round robin	0.491	0.327	0.883	0.866	0.824	0.764	0.652
2.2	Round robin	0.859	0.498	0.876	0.851	0.921	0.874	0.741
2.3	Round robin	0.876	0.483	0.483	0.878	0.891	0.81	0.724
2.4	Round robin	0.846	0.499	0.948	0.875	0.896	0.822	0.732
2.5	Round robin	0.964	0.454	0.968	0.864	0.935	0.794	0.704
3.1	MT-DNN	0.968	0.51	0.988	0.871	0.855	0.851	0.744
3.2	MT-DNN	0.977	0.491	0.985	0.866	0.85	0.843	0.733
3.3	MT-DNN	0.986	0.521	0.99	0.875	0.845	0.842	0.746
3.4	MT-DNN	0.985	0.52	0.986	0.87	0.898	0.851	0.747
3.5	MT-DNN	0.989	**0.521**	0.986	0.872	0.896	0.852	0.748
3.6	MT-DNN	**0.993**	0.511	**0.993**	**0.885**	**0.946**	**0.883**	**0.76**

Table 1. Results from different experiments

Table 1 shows the results on training and development datasets. The number of epochs and the learning rate were changed depending on the previous obtained metrics.
Experiment 1 is the vanilla model pretrained with the frozen BERT model weights. Experiment 2 and 3 use a model fine-tuned using round-robin selection and MT-DNN respectively.

Analysis

The Vanilla BERT model uses single linear head on the pooler output of the task specific sentence embeddings which does not perform well without fine-tuning. Since BERT is trained using next sentence prediction, using the "SEP" token to pass sentence pairs produces richer embeddings for paraphrase detection and textual similarity tasks.

Another improvement is observed when the ordering of the two sentences is taken into account. Hence we generate two embeddings for each sentence pair and then concatenate and pass to the linear layer to compute the final logits. The order of training the model impacts model performance as it is biased towards the task trained last. Round-robin scheduling improves the model performance on all the three task by minimizing this issue as well as addressing the impact of different dataset size.

To optimize the gradient descent for all three task during each step, we sample examples from all the three datasets randomly and compute the three gradients for each step. Leveraging gradient surgery, we optimize the model fine tuning. Across our initial experiments, the learning rate higher than 1e-4 does not work and the gradient descent cannot converge. The effect of batch size does not affect the model performance but is adjusted to 16 based on the GPU memory constraints.

Conclusion

We benchmark our performance with BERT and we were able to achieve similar results on both a multi-tasking model as well as a model fine-tuned on each individual task. This allows deployment of single model for multiple downstream task while preserving the model performance.

References

[1] T. Gao, X. Yao, and D. Chen.
Simcse: Simple contrastive learning of sentence embeddings.
2021.

[2] M. Henderson, R. Al-Rfou, B. Strope, Y.-h. Sung, L. Lukacs, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil.
Efficient natural language response suggestion for smart reply.
2017.

[3] X. Liu, P. He, W. Chen, and J. Gao.
Multi-task deep neural networks for natural language understanding.
arXiv preprint arXiv:1901.11504, 2019.