Robust Embeddings using Contrastive and Multiple Negatives on BERT

Stanford CS224N Default Project

Shoaib Mohammed
Department of Computer Science
Stanford University
shoaibmh@stanford.edu

Ishan Sabane
Department of Electrical Engineering
Stanford University
ishancs@stanford.edu

Abstract

We build a multi-tasking BERT model to perform well on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. 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. We aim to integrate contrastive learning to further improve the sentence embeddings.

1 Key Information to include

- External collaborators (if you have any): N/A
- External mentor (if you have any): N/A
- Sharing project: N/A

2 Approach

We have experimented *six approaches* for multitask classification. Table 1] shows the results the different experiments. At a high level, these approaches and respective experiments analyse the performance changes as a result of changes in the prediction heads for each task, dropout level, and epochs. Below provides more information on the different approaches:

- **Vanilla Multitask:** Using the starter code, we implemented the prediction heads for each task as single linear layer followed by activation function. For SST, we use a simple linear layer with CrossEntropyLoss. For paraphrase detection and STS, we pass the concatenated embeddings of the individual sentences to a linear layer to get a single logit with BCELoss and MSELoss respectively.
- **Concatenate Attention Mask:** Here we improve paraphrase detection and STS. We concatenate the input ids and attention masks of the given sentences and pass it to the BERT model to get a single output embedding. We repeat this by switching the ordering of concatenation. The two embeddings are passed to individual linear layers to get single logits which get passed to a linear layer to get a final logit. We train sequentially on all the three datasets with cosine scheduling.
- **Uniform Training:** The results suggest that the sequential training affects the previous task metrics and the individual task performances drop. To tackle this, we match the dataset skewness by training STS and SST for twice epochs than paraphrase detection. We see considerable improvements using this strategy.

For the baselines, we refer to the Original BERT paper [1] for paraphrase detection and STS. We use the paperswithcode as the SST fine grained baseline. We achieve these baselines on individual tasks.

3 Experiments

3.1 Data

We are using the following three datasets: Stanford Sentiment Treebank (SST) [2] dataset for sentiment analysis, the Quora Question Paraphrase (QQP) [3] for paraphrase detection and Semantic Textual Similarity Benchmark dataset [4]. Table 2 shows the data splits. The fine-tuning time for each epoch on the QQP dataset is *20-min*, whereas it is less than *1-min* for the other two datasets and this is due to differences in dataset sizes as shown in table 2. We plan to use SNLI dataset [5] for pretraining which consist of entailment, neutral and contradiction sentence pairs.

3.2 Evaluation

- **SST:** Since it is a classification task, we use the percentage of correct labels as the accuracy metric. We plan to use mean accuracy over the classes, F1-score and ROC-AUC scores to better understand the classification performance.
- **STS:** We use the Pearson score as the default metric and plan to use cosine similarity of the two sentences.
- **Paraphrase detection:** Since it is a binary classification task, we use the percentage of correct labels as the accuracy metric.

We take the average of SST accuracy, STS Pearson correlation coefficient and the Paraphrase detection accuracy as our score to store the model whenever it improves. This ensures that we do not store the model when it improves on a single task but performs poor across all the three tasks.

3.3 Results

Table 1 shows the results on training and development datasets. We also performed well on the leaderboard ranking in the **top 5**. The number of epochs and the learning rate were changed depending on the previous obtained metrics. More details about the experiments can be found in tables 3 and 4 of the Appendix.

Exp	SST		Paraphrase		STS		Overall
	Train	Dev	Train	Dev	Train	Dev	
1	0.393	0.388	0.008	-0.009	0.235	0.38	0.253
2	0.491	0.327	0.883	0.866	0.824	0.764	0.652
3	0.859	0.498	0.876	0.851	0.921	**0.874**	**0.741**
4	0.846	**0.499**	0.948	0.875	0.896	0.822	0.732
5	0.876	0.483	0.950	**0.878**	0.891	0.810	0.723
6	**0.964**	0.454	**0.968**	0.864	**0.935**	0.794	0.704
7	To be Implemented						

Table 1: Results from different experiments

4 Future work

In an effort to improve over our existing approach, we plan to implement the following extensions.

MT-DNN: As per the MT-DNN approach [6], shuffling the data together and averaging the gradients would lead to performance similar to training the model on individual tasks. For this task we plan to use a single data-loader which can shuffle the three datasets. We compute the three loss functions for the task specific data points in each batch and average and clip the gradients.

SimCSE: As proposed earlier we plan to improve the sentence embeddings by leveraging contrastive learning approach discussed [7]. Implementation of the training pipeline for contrastive learning over BERT base requires a new contrastive loss function and sentence pair datasets [5].

Negative Loss: We found that that negative log loss with negative sentence example pairs improves similar sentence clusters. After implementing contrastive learning it would be easier to add in this loss for comparison. We plan to use the same SNLI dataset [5] for evaluating this improvement.

References

[1] 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.

[2] Papers with Code - SST-5 Fine-grained classification Benchmark (Sentiment Analysis) — paperswithcode.com. `https://paperswithcode.com/sota/sentiment-analysis-on-sst-5-fine-grained`. [Accessed 04-Mar-2023].

[3] Papers with Code - Quora Question Pairs Dataset — paperswithcode.com. `https://paperswithcode.com/dataset/quora-question-pairs`. [Accessed 04-Mar-2023].

[4] Papers with Code - STS Benchmark Dataset — paperswithcode.com. `https://paperswithcode.com/dataset/sts-benchmark`. [Accessed 04-Mar-2023].

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

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

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. 2021.

Appendix

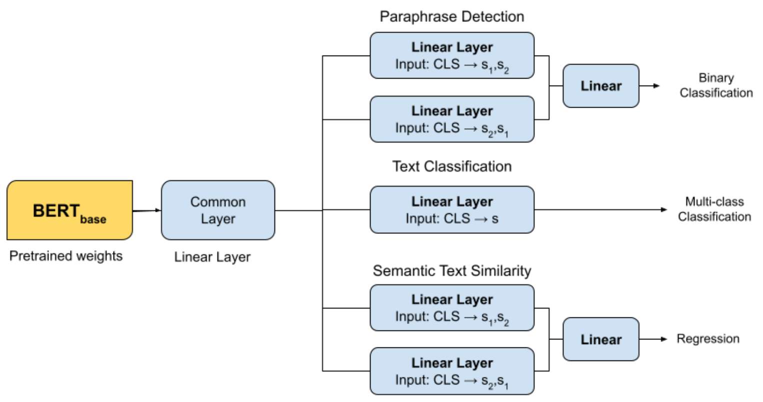

Figure 1: Model architecture

split	SST	CFIMDB	Quora	SemEval STS
train	8,544	1,701	141,506	6,041
dev	1,101	245	20,215	864
test	2,210	488	40,431	1,726

Table 2: Dataset sizes for different splits {`train`, `dev`, `test`}

Experiments	Model	Training Method
1	Vanilla	Pretrain
2	Single Layers with logits	Finetune
3	Single Layers input concat pairs with 2 logits	Finetune
4	Additional Common layer	Finetune
5	Additional Common layer	Finetune
6	Additional Common layer with dropout	Finetune
7	MT-DNN	Finetune

Table 3: Model and training method for different experiments

Exp	Train Order	Epochs	lr	Batch Size	Dropout
1	SST only	10	0.00001	32	0.3
2	QOP,SST,STS	3	0.00001	32	0.5
3	SST,STS,SST,STS,QQP	3	0.00001	32	0.5
4	STS,SST,QQP,SST,STS	3	0.00001	32	0.5
5	STS,SST,QQP,SST,STS	3	0.00001	16	0.5
6	STS,SST,SST,STS,QQP	5	0.00001	32	0.1
7	Mixed	5	0.00005	32	0.1

Table 4: Changing hyperparamters for different experiments