PolarBERT: Enhancing Robustness and Generalizability of BERT Sentence Embeddings through Multiple Negatives Ranking Loss and Contrastive Learning

Stanford CS224N Default Project

Tyler Hanson
Department of Computer Science
Stanford University
thanson2@stanford.edu

Jaisal Kothari
Department of Computer Science
Stanford University
jkothari@stanford.edu

Lucy Zhu
Department of Computer Science
Stanford University
lzhu21@stanford.edu

Abstract

Our project focuses on enhancing the performance of a BERT model for sentiment analysis, paraphrase detection, and semantic textual similarity. To achieve this, we have utilized two techniques: multiple negative ranking loss and contrastive learning. We have also changed the model architecture to be deeper with more linear layers and modified our similarity functions. The multiple negative ranking loss method compares a valid response to multiple invalid responses, penalizing the output if it is not ranked higher than the others. The 'Efficient Natural Language Response for Smart Reply' paper states a consistent 20% error reduction using this technique. The contrastive learning method uses dropout to create positive and negative pairs with no additional data. Data from the forward pass will be randomly dropped from a sentence; the modified sentence with another modified version of itself will serve as a positive pair, and the modified sentence with a different modified sentence will serve as a negative pair. This effectively creates more training data and has done well for paraphrasing and similarity detection. We have found the contrastive learning to improve our accuracy by 3% over the previous model and multiple negative ranking loss to perform on par with other loss functions such as MSE.

1 Key Information to include

- Mentor: Gabriel Poesia Reis e Silva
- External Collaborators (if you have any): None
- Sharing project: https://github.com/thanson2/CS224n_project

2 Introduction

As a transformer-based model, BERT (Bidirectional Encoder Representations from Transformers) was a revolutionary transformer-based model released in 2018 that utilized bidirectional word representations. BERT is capable of simultaneously performing multiple sentence-level tasks. We focus on three of the tasks: sentiment analysis which classifies a text's polarity (positive/negative/neutral),

paraphrase detection which finds paraphrases of texts in bodies of text, and semantic textual similarity which determines semantic equivalence between texts. While it may be sufficient to produce a model for each task separately, there is a great convenience and potential improvements in performance through pooling together the knowledge from the otherwise separate models, which can be achieved with BERT's embeddings. The original BERT model has performed poorly in sentence embeddings (it underperforms conventional word embedding techniques like GloVe) Anisotropy in BERT makes the token embeddings occupy a narrow cone, resulting in a high similarity between any sentence pair. The primary challenge we face in designing a model that can tackle all three tasks simultaneously is in combining the learning from each task. Our approach is to interleave learning on batches from each dataset. Further performance gains are then pursued through contrastive learning and multiple negatives ranking loss. Multiple negatives ranking loss is employed with the intent to improve embeddings for the paraphrase detection and semantic textual similarity tasks through augmenting the batches with additional negative samples and maximizing the difference between the distances of the scores of the positive and negative pairs

3 Related Work

Our project was in part based on the research done in 'Efficient Natural Language Response for Smart Reply' (Henderson et al., 2017), SimCSE: Simple Constrastive Learning of Sentence Embeddings (Tianyu Gao, 2021).

Smart Reply paper details the approaches taken to improve the response selection step of the Smart Reply system, which tackles the task of response suggestions in human-to-human conversations. One of the approaches is to implement a new learning method, the Multiple Negatives Ranking Loss to improve sentence embeddings. This implementation augmented each batch by generating additional negative samples through pairing up each sample with all other samples within each batch and aimed to minimize the distance between the positive pairs' sentences' embedding while simultaneously maximizing the distance between the negative pairs' sentences' embeddings. Application of this new loss function demonstrated a consistent 20% error reduction over the Smart Reply dataset when compared against training with sigmoid loss.

Since the loss is performed on sample pairs, we found it to be applicable to both the paraphrase detection and the semantic textual similarity tasks as they utilize sentence pairs where negative samples can be easily generated and seamlessly included in each batch.

SimCSE proposes a simple supervised and unsupervised method for improving sentence embeddings and serves as a natural way to regularize the model. The unsupervised method create positive and negative sentence pairs, where the positive pair is the same sentence twice, using only random dropout to augment the data, and the negative pair is two different sentences with the same random dropout applied. The model is then trained to predict positive and negative pairs from the data.

The supervised method is similar to the unsupervised method, however labeled contradicting and entailment pairs are fed into the model. In the paper, the model is tested on a semantic textual similarity task. We anticipated that improvements might also be found in the paraphrase detection tasks as the two are quite similar. The authors found a 4.2% and 2.2% improvement on unsupervised and supervised Spearman's correlation compared to the previous best results.

We chose to adapt the unsupervised method to our model as the dataset could be trivially created from our existing datasets and it provided the higher gains of the two methods.

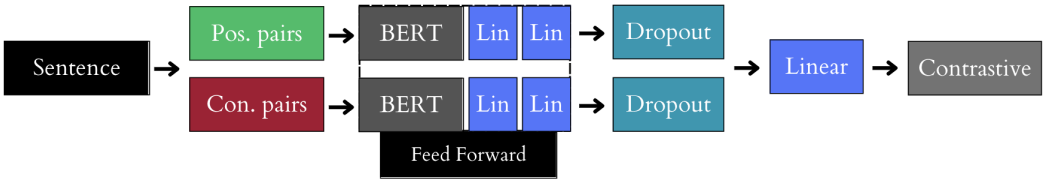
4 Approach

We built our model on top of a BERT model. Our forward layer called the base BERT forward layer and then passed through 2 linear layers. Our model had 4 prediction functions: predict_contrast, predict_similarity, predict_sentiment and predict_paraphrase.

We extended the model with contrastive learning which improved the embeddings of the model with no extra data. During training we created contrastive learning batches by taking a batch from the Stanford Sentiment Treebank (which included a single sentence and masks), duplicating the sentences and masks, and flipping the order of the second half of the sentences, thus the first half served as positive pairs and the second half as contrastive pairs. These two sentences are fed through

the forward model, then a dropout layer, and then concatenated and passed into a forward layer which outputs a single logit. The dropout augments the data to make the prediction task not trivial. We found that inputting only one sentence through the dropout layer instead of both, and using a single linear layer on the concatenated outputs instead of cosine similarity improved the dev accuracy notably.

Figure 1: Contrastive Learning Architecture

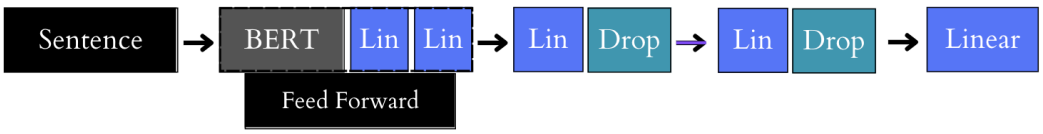


We used round robin training for the **predict_similarity** and **predict_sentiment** tasks, as we found if we trained on a full set, then the next set a lot the accuracy of the first tasks were diminished. We originally included predict paraphrase in the round robin, but as it is a much larger dataset, the round robin only trained on a small subset of the data, so we moved the paraphrase data to outside the round robin which empirically performed much better. We chose to train the contrastive learning on its own before the other training because contrastive learning is not one of the objectives of our model and served as a form of pretraining, so it did not matter if its accuracy diminished to the benefit of the other tasks.

predict_sentiment and **predict_similarity** were nearly identical in structure and even shared some layers. The two inputs were concatenated together and passed to the forward method. The output was passed through 3 linear layers each separated by a dropout layer with the final linear layer outputting a single logit as the prediction. We found this method far outperformed processing the inputs separately and using cosine similarity or other similarity functions at the end.

predict_sentiment differed from the other tasks as it was only given a single input sentence. We fed it through forward two linear layers separated by a dropout and then passed the results through a softmax. We found the softmax improved our results as the predictions were closer to probabilities.

Figure 2: General Multi-task Architecture



1. Predict Sentiment: final linear layer changes input_dim from hidden_size to 5; uses softmax and then rescales it by half of batch_size; uses cross-entropy loss using sum reduction.
2. Predict Paraphrase: final linear layer changes input_dim to 1; uses MSE using sum reduction / Multiple Negatives Ranking loss.
3. Predict Similarity: final linear layer changes input_dim to 1; divides logits by 5; uses MSE loss using sum reduction.

As an attempt to improve the embeddings, we implemented multiple negatives ranking log loss and applied it to the logits output by **predict_similarity** and **predict_paraphrase** as both involved sentence pairs, which the K-pairs structure of the loss can be adapted towards. The logits were modified to generate all K^2 possible pairs where K is the batch size of the dataset the loss is applied to. The logits obtained from the original and new pairs are then passed into the loss learning that simultaneously minimizes the distance between the positive sentence pair's embeddings and maximizes the distance between the negative sentence pair's embeddings.

For a given K sentence pairs $[(x_1, y_1), \ldots, (x_n, y_n)]$ between $\mathbf{x} = (x_1, \ldots, x_K)$ and $\mathbf{y} = (y_1, \ldots, y_K)$ where (x_i, y_i) are labeled as similar sentences, and (x_i, y_j) where $i \neq j$ are labeled as not similar sentences. The loss function is as follows:

$$\mathcal{J}(\mathbf{x}, \mathbf{y}, \theta)$$
$$= -\frac{1}{K}\sum_{i=1}^{K} \log P_{\text{approx}}(y_i \mid x_i)$$
$$= -\frac{1}{K}\sum_{i=1}^{K} \left[S(x_i, y_i) - \log \sum_{j=1}^{K} e^{S(x_i, y_j)} \right]$$

For each batch, the $K-1$ negative samples are generated from the existing sentences within the original batch. This formula aims to minimize the distance between x_i, y_i similar sentence pairs and simultaneously maximize the distance between x_i, y_j not similar sentence pairs by minimizing the approximate mean negative log probability of the data. θ represents the word embeddings and neural network parameters. S is the scoring function.

The batches given by the datasets don't necessarily include only positive pairs, so we implemented additional batch processing. To do so, we still created all the possible pairs between each sentence in the batch, computed the logits for each one. Once we computed the logits, for the true positive pairs part of the equation aka $S(x_i, y_i)$ where x_i, y_i are pairs of sentences that are to be considered as positive samples, they're denoted with a nonzero value in labels while the rest are denoted as negative samples.

5 Experiments

5.1 Data

The datasets being used are the Stanford Sentiment Treebank[1] and the CFIMDB dataset for sentiment analysis that assigns sentences a sentiment score ranging from 0 (negative) to 4 (positive), Quora[2] dataset for paraphrase detection, and the SemEval STS Benchmark dataset for the semantic textual similarity task.

We also created a dataset from the Stanford Sentiment Treebank for the contrastive learning objective. This was done by taking a batch from the SST set, duplicating the tokens ids and masks, and flipping the second half of the token and masks tensors. Therefore the first half of the two sets would be identical, serving as the positive pairs, and the second half would be different, serving as contrastive pairs. The labels generated were 1 for the first half and 0 for the second half.

To apply multiple negatives ranking loss, we augmented the Quora and SemEval STS Benchmark datasets to add hard negative sentence pairs such that they both

5.2 Evaluation method

We utilized the accuracy on both the SST and Quora test datasets, and Pearson correlation of the true against the predicted similarity values on the SemEval dataset for the quantitative evaluation. The three scored were averaged for the final dev and train accuracies.

The baseline we compare our model to is a model trained only on the SST dataset and applies cosine similarity loss learning to all three tasks. While outperforming a random selection in the SST task, the model's performance is not different from that of a random selection model that picks correctly the correct class out of 2 for the paraphrasing task and displays a low Pearson correlation for the semantic textual similarity tasks. This is to check whether our model outperforms the method of randomly picking a value for each sample.

5.3 Experimental details

Pretraining and finetuning were conducted for 10 epochs with a learning rate of e^{-5} and a dropout rate of 0.3. Across all datasets, the batch size is set to 8.

[1] `https://nlp.stanford.edu/sentiment/treebank.html`

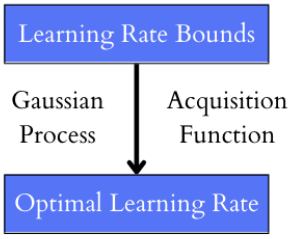
[2] `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

The highest dev accuracy was typically found around epoch 6-8, and tended to overtrain after that.

Increasing the batch size led to lower performance, and decreasing the batch size to 1 led to small gains but intolerably slow training times.

We also carried out Bayesian Optimization to find the optimal learning rate as part of hyperparameter optimization. (Snoek et al., 2012) To achieve this, we created bounds for our learning rate and ran the Gaussian Process on our black box training function. Over iterations, the acquisition function found regions of optimal values.

Figure 3: Bayesian Optimization Process



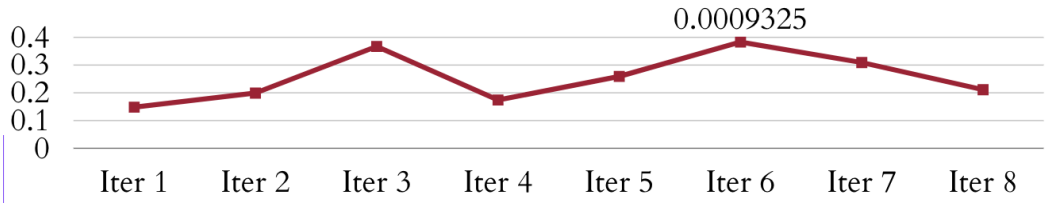
Our experimentation with Adamax (Kingma and Ba, 2014) and other optimizers did not give any noticeable improvements.

5.4 Results

Experimental Results:

Experiment (results on Epoch 0)	Training	Dev	Paraphrase	Sentiment	STS
Baseline (Cosine similarity + Only SST dataset)	0.292	0.282	0.506	0.318	0.023
Train on all datasets	0.304	0.306	0.375	0.272	0.27
Extra linear layer on each function (Cosine similarity)	0.447	0.418	0.375	0.434	0.446
Combine inputs in one layer	0.629	0.599	0.505	**0.5**	0.791
With multiple negative ranking loss	0.627	0.592	0.491	0.485	0.799
Contrastive learning using cosine similarity	0.603	0.563	0.425	0.482	0.782
Contrastive learning using combined inputs	0.658	0.617	0.573	0.472	0.807
Two more linear layers and dropout layers	0.676	0.632	0.607	0.475	0.813
Remove dropout on one contrastive learning	**0.679**	**0.643**	**0.632**	0.481	0.815
Additional Linear Layer	0.582	0.551	0.438	0.417	0.799
Train on paraphrase, then round robin STS, SST	0.661	0.626	0.56	0.48	**0.84**
Train on paraphrase, round robin STS, SST, Epoch 4	**0.864**	**0.721**	**0.825**	**0.516**	**0.822**

Figure 4: Learning Rate Optimization Iterations



We ran Bayesian Optimization on the learning rate for 8 iterations with 15 initiation points. We got **0.0009325** as our optimal value.

Baseline Results:

Baseline Dev Set Accuracy		
Set	Pretraining only	Finetuning
SST	0.393	0.520
CFIMDB	0.788	0.967

Baseline Model Statistics	
Set	Accuracy/Correlation
Sentiment	0.318
Paraphrase	0.506
STS	0.023

Final Results:

Dev Set Accuracy		
Set	Pretraining only	Finetuning
SST	0.393	0.520
CFIMDB	0.788	0.967

Model Statistics	
Set	Test Accuracy/Correlation
Sentiment	0.531
Paraphrase	0.822
STS	0.848
Overall	**0.733**

6 Analysis

PolarBERT extends the BERT model by contrastive learning and multiple negatives ranking loss, both of which improve the model by the concept of generating positive and negative pairs. A qualitative analysis shows that we may have created, though enhanced, a polar model, which prefers extremes over medians. This was an unexpected and very interesting finding for us.

For Sentiment, over-enthusiastic in sentimentality scoring such that it predicts a more positive/negative sentimental rating.

Sentiment Task Analysis		
Positive Example	Prediction	Actual
It's a lovely film with lovely performances by Buy and Accorsi.	4	3
Entertains by providing good, lively company.	4	3
Unlike the speedy wham-bam effect of most Hollywood offerings, character development – and more importantly, character empathy – is at the heart of Italian for Beginners.	4	3
Negative Example	Prediction	Actual
Half Submarine flick, Half Ghost Story, All in one criminally neglected film.	1	2
-LRB- Lawrence bounces -RRB- all over the stage, dancing, running, sweating, mopping his face and generally displaying the wacky talent that brought him fame in the first place.	1	3
A coda in every sense, The Pinochet Case splits time between a minute-by-minute account of the British court's extradition chess game and the regime's talking-head survivors.	1	4

For Semantic Similarity, the model's cautious about rating similarities such that for extremely similar sentences, it tends to rate them less similar.

Semantic Similarity Task Analysis		
Example	Prediction	Actual
Russia, China Veto UN Resolution on Syria Russia, China veto UN resolution on Syria killings.	2.39	3.8
The couple danced in the church. A couple of slow dances.	0.65	3.2
Some guy sitting on a couch watching television. A guy is sitting on the couch watching TV.	4.2	5.0

7 Conclusion

We were able to significantly improve the baseline BERT model on the three sentence-level tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Our model improved by 3.0% when using contrastive learning. Applying the multiple negatives ranking loss resulted in no significant change in performance (decreased training and dev average accuracies by 0.2% and 0.7%.)

Unsupervised Contrastive Learning Contrastive Learning proved to be the most effective. Inputting only one sentence through the dropout layer instead of both, and using a single linear layer on the concatenated outputs instead of cosine similarity improved performance notably. It was intended to improve similarity detection and in practice, it improved the similarity detection and the paraphrase detection but slightly worsened the sentiment analysis.

Multiple Negatives Ranking Loss Implemented one way of modifying the batches to pass into the loss, but there are other potentially more effective approaches such as randomly sampling the true positive and negative pairs to use in the loss computation. The loss is effective when there are a consistent amount of positive samples in each batch, but the input involved both positive and negative samples.

Hyperparameter Optimization Given more time and computing resources, we would like to pursue additional modifications to the extensions we implemented and potentially apply bayesian optimization to other parameters beyond the learning rate.

References

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms.

Danqi Chen Tianyu Gao, Xingcheng Yao. 2021. In *SimCSE: Simple Contrastive Learning of Sentence Embeddings*.