Improved BERT embeddings through Negative Rank Loss

Stanford CS224N Default Project

Torstein Ørbeck Eliassen
Department of Electrical Engineering
Stanford University
`torsteoe@stanford.edu`

Aarya Mecwan
Department of Electrical Engineering
Stanford University
`amecwan@stanford.edu`

Natalie Bishay
Department of Electrical Engineering
Stanford University
`nbishay@stanford.edu`

Abstract

In this project, we implement a multi-headed minBERT model and then analyze its performance on sentiment analysis, paraphrase detection, and semantic textual similarity. To improve the performance of our model on the multi-task problem, we implement Multiple Negatives Ranking Loss Learning[6], which maximizes the difference between two dissimilar words while also minimizing the difference between two similar words. We also add a shared layer between the paraphrase and similarity tasks to further increase complexity of the model without overfitting. We find that implementing these extensions, as well as intuitively scaling some losses, greatly increases the accuracy on all three tasks.

1 Key Information to include

- Mentor: Sauren Khosla
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Multitask learning in machine learning is generally a difficult problem where we aim to have a single model, in our case the minBERT model, perform well on multiple tasks, that are more or less related. Without adequate machine complexity and targeted training we are unable to obtain reasonable (say, better than chance) accuracy on all the tasks at hand due to their differences. Using the minBERT model embeddings and a multihead multitask classifier, we train a model that solves the following tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Sentiment analysis involves describing phrases as negative, somewhat negative, neutral, somewhat positive, or positive; paraphrase detection determines whether particular words or phrases convey the same semantic meaning; and semantic textual similarity seeks to measure the degree of semantic equivalence. These tasks are all quite different. To improve the expressivity of the embeddings, we implement the negative rank loss implemented by Henderson et al. (2017) on the output of the minBERT model, which maximizes the angle difference between two dissimilar words while also minimizing the angle difference between two similar words. Additionally, since we intuitively, and as stated as a given, know that the latter two tasks are related, we incorporated a shared layer in between them to have the model learn from them at the same time, which we found was a simple yet effective way to link the two tasks together.

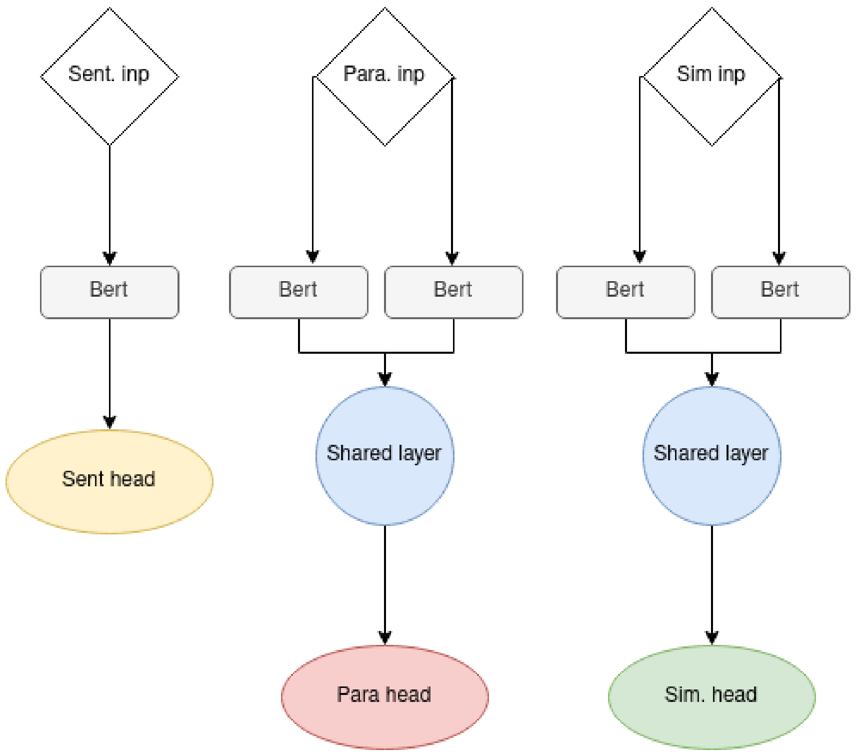


Figure 1: Model architecture

3 Related Work

For our baseline model, we use a BERT model a bidirectional encoder built on a large-scale multi-headed transformer. We referred to Devlin et al. (2018) for details on the BERT encoder and Vaswani et al. (2017) for details on the transformer. At a high level, BERT is a method of pre-training language representations. Pre-training refers to how BERT is first trained on a large source of text. We can then apply the training results to other natural language processing tasks, such as sentiment analysis, paraphrase detection, and semantic text similarity. Negative rank loss is presented in Henderson et al. (2017). The paper presents an efficient machine-learning method for natural language response suggestions. I.e. suggesting dialog answers to human-made sentences. Reimers and Gurevych (2019) use the negative rank loss to make meaningful sentence embeddings that can be applied to multiple tasks, this may imply that the idea can work well for our multitask purpose as well.

4 Approach

The task is a multi-classifier task. We are given either 1 or 2 sentences and must predict either sentiment of a single sentence, similarity of 2 sentences or do paraphrase detection of 2 sentences. We create a 3-headed model with three different heads, one that outputs the logits for predicting sentiment, one that outputs logits for predicting the paraphrase binary logits, and one that outputs the similarity score. All the models have as a baseline the MinBERT model.

4.1 Model architecture

The model architecture is given in Figure 1

The input for all three tasks is sent through the same BERT encoder. As the paraphrase task and similarity tasks both base themselves on that the model understands "likeness" between sentences, we add a layer that is shared among them to capture this functionality. This will hopefully reduce overfitting as the layer must fit both tasks, it might also improve performance as we increase the amount of data this layer can learn from. The shared layer is applied after summing the two BERT embeddings. Finally all three tasks have their own head.

4.1.1 BERT encoder

For our baseline model, we use a BERT model a bidirectional encoder built on a large-scale multi-headed transformer. We referred to Devlin et al. (2018) for details on the BERT encoder and Vaswani et al. (2017) for details on the transformer.

4.1.2 Sentiment head

The sentiment head is a linear layer with output size 5, one dimension for each class.

4.1.3 Shared layer

The shared layer inputs the summed BERT embeddings and passes it through a linear layer with hidden size the same as the hidden size of BERT.

4.1.4 Paraphrase head

The paraphrase head passes the output of the shared layer through a ReLU and then a linear layer with output size 1, a binary logit.

4.1.5 Similarity head

The similarity head passes the output of the shared layer through a ReLU and then a linear layer with output size 1, a scalar output measuring similarity. The output is finally passed through another ReLU as we know the value should be positive.

4.2 Loss functions for regular tasks

For sentiment classification, the output is a probability so we use categorical cross entropy as a loss function. The paraphrase task is a binary-decision task. The output is a probability and we therefore use Binary Cross Entropy to calculate the loss. The similarity task is a regression task, where prediction is to be done on a scale from 0 to 5. For this regression task it seems reasonable to use MSE as a loss as the outputs are continuous and the labels can be thought of as generated from Gaussian distributions.

4.3 Extension to loss functions: Custom Loss function using Negative Rank Loss

To add more expressivity to the word embeddings, we use negative rank loss on the output of BERTHenderson et al. (2017). We adapt the code from Reimers and Gurevych (2019). We add the loss to the paraphrase pairs. To implement this, we reload the paraphrase dataset with only positive datasamples. We generate an extra head to our model that simply outputs the BERT embeddings and inputs them to our custom loss function.

Let our batch of sentence pair embeddings be: $[(a_1, b_1) \quad (a_2, b_2) \quad \dots \quad (a_n, b_n)]$. Let A be the cosine similarity of each left sentence embedding to each right sentence embedding.

$$A = \begin{bmatrix} \cos(a_1, b_1) & \cos(a_1, b_2) & \dots & \cos(a_1, b_n) \\ \cos(a_2, b_1) & \ddots & & \\ \dots & & & \\ \cos(a_n, b_n) & \dots & & \cos(a_n, b_n) \end{bmatrix} \tag{1}$$

By taking the categorical cross entropy loss between A and a one hot encoded range of numbers, we are penalizing deviations between A and the identity matrix I. A scaling value is added as a tuning parameter for the loss function and set to 20 as in the original code.

Regarding implementation, the cosine similarity can easily be calculated by normalizing each embedding vector and taking the dot product between the embeddings.

4.4 Training

We choose a finetuning approach for training, by starting from the pretrained BERT weights and set all parameters as trainable. We do this since we want to train the BERT embeddings with our custom loss function, but we also deem the 3 tasks to be sufficiently different from the original BERT task to justify nudging the BERT parameters.

We choose a Round Robin approach for training the 4 different loss functions. At each epoch, we train all 4 objectives, one at a time. As the loss functions for the Sentiment and the Paraphrase task output values that are approximately 5x larger than our two other loss functions, we rescale the other tasks' learning rate by approximately 5.

Optimization is done with the Adam optimizer. We refer to Kingma and Ba (2014) for details.

5 Experiments

5.1 Data

The data we utilized is described below.

- **Paraphrase detection**: Quora Dataset[1]: This dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another.
  - train: (141,506 examples)
  - dev: (20,215 examples)
  - test: (40,431 examples)
- **Sentiment Classification**: Stanford Sentiment Treebank (SST) dataset: consists of 11,855 single sentences extracted from movie reviews.
  - train: (8,544 examples)
  - dev: (1,101 examples)
  - test: (2,210 examples)
- **Similarity task** : SemEval STS Benchmark dataset: [2]: consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).
  - train: (6,041 examples)
  - dev: (864 examples)
  - test: (1,726 examples)

5.2 Evaluation method

The evaluation metric as specified in the default final project statement is a percent accuracy on unseen data of the predicted labels vs. the labels the model outputs.

5.3 Experimental details

5.4 Pretraining and finetuning the model for single task sentiment classification

The hyperparameters of pretraining and finetuning the model are given below:

- LR : 1e-3 (pretrain) 1e-5 (finetune)
- Epochs : 10
- Batch Size : 8
- Dropout probability : 0.3

The table contains the pretrain and finetune accuracies for our baseline BERT and AdamW optimizer implementation.

[1] `https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs`
[2] `https://aclanthology.org/S131004.pdf`

Model	Score
SST Pretrain Dev Accuracy	0.371
SST Finetune Dev Accuracy	0.491
CFIMDB Pretrain Dev Accuracy	0.673
CFIMDB Finetune Dev Accuracy	0.963

5.4.1 Training multiclassifier model

Experiment 1.
We have three possibly competing objectives in this first experiment, where we do training in a epoch-based round robin fashion; we train on each model for a whole epoch at a time. This means that we technically go through the whole dataset 3 times per epoch. This is before adding the negative rank loss and a single linear head for each task.

The hyperparameters are given below:

- LR : 1e-3
- Epochs : 10 (x3)
- Batch Size : 8
- Dropout probability : 0.3

Training time was approx. 1.5 hours.

Experiment 2.
We have four possibly competing objectives, once we add in the negative rank loss. Training is done in a epoch-based round robin fashion, meaning we train on each model for a whole epoch at a time. This means that we technically go through the whole dataset 4 times per epoch.

The hyperparameters are given below, for our first experiment with negative rank loss:

- LR : 1e-3
- Epochs : 10 (x4)
- Batch Size : 8
- Dropout probability : 0.3

Training time was approximately 5 hours.

Experiment 3.
When we added the shared layer, we scaled the loss by a factor of 5 for just the losses from the sentiment analysis and the negative rank loss, determined quite loosely from looking at the figure from our first experiment in Figure 2, which is in the appendix. Essentially, we found that the negative rank loss and the sentiment analysis loss were quite low. This can be seen in the figures, and the new loss can also be seen in Figure 3. For the task where the loss is low, we scale up the learning rate. The hyperparameters are given below, for our second experiment with negative rank loss, as well as the adding of the shared layer and the aforementioned scaling:

- LR : 5e-3 for SST and Neg Rank, 1e-3 Paraphrase and STS
- Epochs : 10 (x4)
- Batch Size : 8
- Dropout probability : 0.3

Training time was approximately 5 hours.

5.5 Results

Our results are in the two tables below. We only have test set results for our third and final experiment.

Figure 2: Experiment 2 Losses

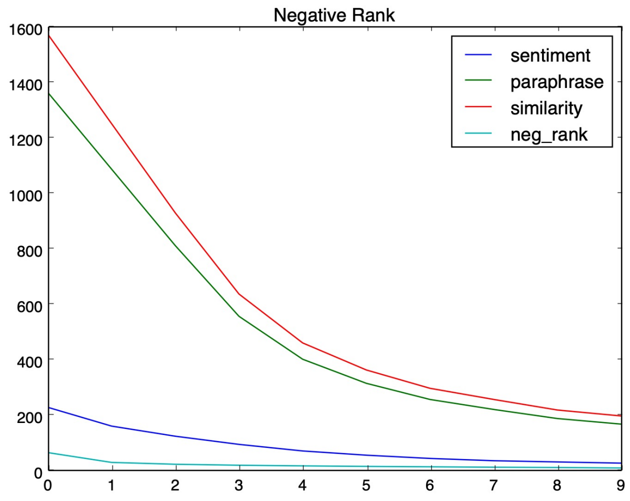
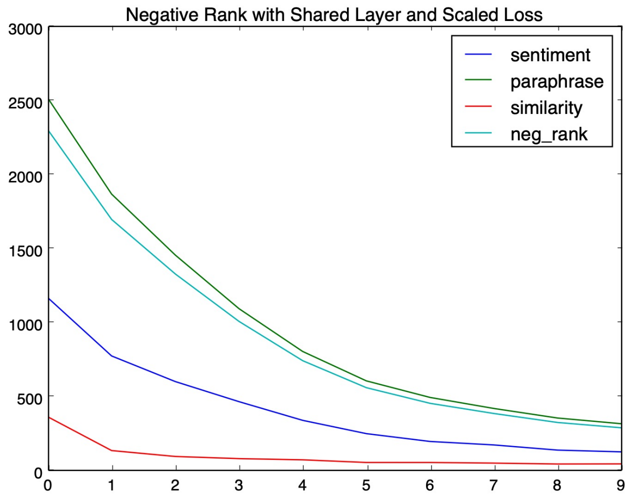


Figure 3: Experiment 3 Losses



Model	Exp. 1	Exp. 2	Exp. 3
SST dev Accuracy	0.307	0.266	0.401
Paraphrase dev Accuracy	0.652	0.767	0.800
STS dev Correlation	0.214	0.362	0.426
Overall dev score	0.391	N/A	0.542

Model	Exp. 3 Score
SST test Accuracy	0.408
Paraphrase test Accuracy	0.801
STS test Correlation	0.396
Overall test score	0.535

6 Analysis

Looking at the accuracies for all of our experiments, we can reason well about what metrics we took to improve our multitask classification. In the first experiment, we had essentially nothing that allowed for the model to learn the tasks well; since we were doing a round robin and just adding the losses together, we see that our model learns better than chance, but our dev accuracies for experiment 1 are quite low.
In experiment 2, after incorporating the negative rank loss, we see that the accuracy for paraphrase dev and STS dev both increase, but SST decreases. We think this is because the cosine loss helps link embeddings that are more similar together, and forces other embeddings to be perpendicular. Therefore, the tasks that would benefit from this are the ones that have to assess for similar meanings, which both paraphrase and STS do. However, the SST accuracy decreased. When we plotted this in Figure 2, we saw that this is likely because the SST loss was a lot smaller in magnitude compared to the others. We also observed during training that after adding the negative rank loss that training accuracy goes very high, creating some dev-train disparity.
Therefore, with all of this in mind, we designed experiment 3 by multiplying the SST loss and negative rank loss by 5 before adding it to the overall train loss – in practice we simply increased the learning rate by 5. This would prioritize the sentiment task higher. We also incorporated a shared layer between the paraphrase and STS outputs to mitigate overfitting, while increasing model complexity. We see that the dev accuracies for all 3 increase after doing this, which was very exciting, and justified that the aforementioned reasoning was valid.

7 Conclusion

Two main extensions were done in this work: negative ranking loss and shared model layers for two-sentence tasks. Negative ranking loss on the paraphrase detection sentences has a positive outcome on our model, for both the paraphrase detection task and the similarity task. This may indicate that making embeddings more orthogonal is important for multitask objectives. The shared layer not only improved the results during training, but considerably during validation; this shows how well shared layers in multi-task learning can prevent overfitting.

The results for the final model's dev and test sets were promising, and definitely improved from the preliminary round-robin approach, especially the paraphrase test accuracy of over 0.8, and far better than chance on the other two tasks, with a notable improvement from experiment to experiment.

The results are decent, but can definitely be improved. We believe that the last layers could benefit from further tuning with the BERT parameters freezed. We could also add complexity to the sentiment head, as overfitting is not our primary issue there.

Future work would consist of more hyperparameter tuning and other model architectures. In addition, doing more work on balancing the datasets would be beneficial, as well as doing batch-wise round robin versus epoch-wise round robin.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.