Techniques for Extracting Meaningful BERT Sentence Embeddings for Downstream Tasks

Stanford CS224N: Natural Language Processing with Deep Learning

Jacob Mejia
Department of Computer Science
Stanford University
jamejia@stanford.edu

Matt Harvill
Department of Computer Science
Stanford University
mharvill@stanford.edu

Michael Xue
Department of Computer Science
Stanford University
mikexue7@stanford.edu

Abstract

In this project, we first implement key components of the BERT transformer model to gain a better understanding of the architecture. We then focus the majority of our effort on finetuning and building on top of the base BERT model in order to extract richer sentence embeddings and succeed at multiple downstream tasks. Our tasks of interest include sentiment analysis, paraphrase detection, and semantic textual similarity. We find that the combination of using Jaccard similarity for sentence comparison tasks, weighing the losses of the three tasks, sharing network weights across paraphrase and textual similarity tasks, and representing sentences by the average of their token embeddings gives us optimal performance on our tasks of interest. We also test with other methods that don't improve performance across tasks and discuss the implications.

1 Key Information

- Mentor: David Huang
- Sharing project: N/A

2 Introduction

The emergence of the transformer architecture has significantly increased the capabilities of the field of natural language processing and become the new state-of-the-art with regard to language model architecture. Transformers have several advantages in comparison to traditional NLP models such as RNNs and LSTMs. First, transformers have the ability to capture long range dependencies between words and thus capture contextual information efficiently, a limitation of its counterparts. Second, transformers can learn representations that are more reusable and transferable across different tasks, making them more efficient for transfer learning.

One of the most important and difficult obstacles in natural language processing is developing sentence embeddings capable of performing well on a variety tasks. Learning robust sentence embeddings will result in machines better understanding human language and a model capable of performing multitask learning efficiently. The difficulty in achieving this goal lies in optimizing a finetuning process capable of learning generalizable sentence embeddings across different tasks. We aim to tackle this issue and present proven strategies for achieving this goal.

In this project, we implement a transformer based architecture, minBERT, and then use techniques that result in better learned sentence embeddings for higher performance on downstream NLP tasks. More specifically, we improve upon pretrained BERT embeddings for our downstream tasks of interest by architecting a finetuning pipeline with the goal of learning more robust embeddings. We accomplish this goal by experimenting with many strategies. The most salient strategies include representing sentences by the average of their token embeddings as opposed to using the CLS token embedding, building separate feed-forward networks for sentiment classification and sentence similarity/paraphrase detection, and weighing the sentiment classification related losses higher with respect to the other tasks. The details of each of these techniques and more are outlined in the **Approach** section.

3 Related Work

Since BERT embeddings can be obtained without training, they are often used for many downstream tasks. Unsuprisingly, it is common for them to be used as a starting point for a range of multitask learning objectives, including Polarity and Subjectivity Detection (Satapathy et al., 2022) as an example. Due to the large range of these applications, the study of multitask learning built on BERT embeddings is important and impactful to many parties; this prompted the creation of GLUE (Wang et al., 2019), a common benchmark of NLP tasks including semantic textual similarity, paraphrase detection, sentence similarity, and more.

Many researchers have studied methods used to finetune BERT for improved multitask performance, including Zhang et al. (2021), who showed that not all BERT layers are beneficial to transferring learning across tasks. In Reimers and Gurevych (2019), it was also discovered that using the CLS token as the sentence embedding was not as effective as using an average of the sentence's token embeddings or GloVe embeddings. In our attempt to finetune a BERT model to succeed at sentiment analysis, paraphrase detection, and semantic textual similarity tasks, we incorporated ideas from these papers and more to produce our well-rounded multitask model.

4 Approach

4.1 Baseline

Our baseline model includes up to a single linear layer for each of the three tasks on top of BERT. For sentiment classification, we transform BERT embeddings to output five logits followed by a softmax to produce probabilities for the five classes. For paraphrase detection, we pass the difference of each sentence's BERT embeddings through our task-specific linear layer and transform it into a probability of paraphrase with a sigmoid activation. Lastly, for our semantic textual similarity task we use a linear transform of the cosine similarity of our sentences to output sentence similarities ranging from 0-5.

4.2 Main Strategies

After the base model implementation, we focus on strategies to improve our pretrained BERT sentence embeddings. Our first improvement to our baseline model includes implementing feed-forward networks with skip connections, dropout, batchnorm, and ReLU activations to replace our intitial linear layers. We then find that the following strategies improve our embeddings and generalize well across our tasks of interest, placing us in the top quartile of leaderboard submissions.

4.2.1 Average Token Embeddings vs CLS Token Embedding

Through initial testing, we find that representing our sentences with the average of their tokens' BERT embeddings leads to better performance on our tasks compared to using the CLS tokens of each sentence, inspired partially by Reimers and Gurevych (2019). We conclude that using the average of token embeddings captures important information about the sentence for our given tasks that is not captured in the CLS token embedding. We compare the results of each approach in **Experiments** and prove that we achieve stronger results using the average.

4.2.2 Jaccard Similarity for Paraphrase Detection and Semantic Textual Similarity

Computing the Jaccard Similarity of two sentences is simple—all we have to do is compute the number of tokens that overlap divided by the total number of unique tokens shared between the sentences. We decide to add the Jaccard Similarity of inputs for Paraphrase Detection and Semantic Textual Similarity and find that it improves performance, inspired by work from Jeyaraj and Kasthurirathna (2021). We append this feature to the BERT embedding difference of our two input sentences and feed this new embedding through to our feed-forward network. This is visualized in Figure 1 below by the Yellow *Jaccard Similarity* segment.

4.2.3 Sharing Weights for Similar Tasks

Another effective strategy is sharing weights between the paraphrase detection and sentence similarity networks. We recognize that paraphrase detection and sentence similarity are inherently very similar tasks, and decide to use the same feed-forward network for each, inspired partially by Reimers and Gurevych (2019). Thus, we pass as input batches of BERT embeddings for the similarity task to the paraphrase detection network. The backpropagation updates for the network are then influenced by the loss function of both tasks. The result is a robust network that can perform well on both tasks simultaneously and learn the underlying similarities between each of the tasks. The only difference is that we scale the output from 0-1 (used directly for paraphase detection) to 0-5 for semantic textual similarity.

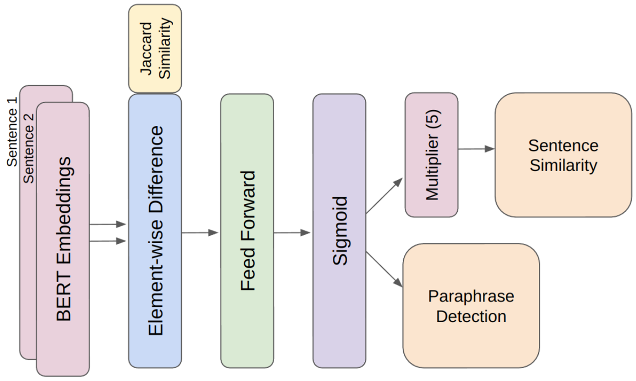

Figure 1: Shared Network for Paraphrase Detection and Semantic Textual Similarity

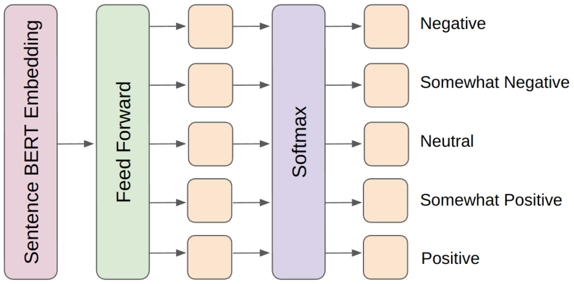

Figure 2: Sentiment Classification Specific Network on top of BERT

4.2.4 Weighted Loss Function

We experiment with using a weighted loss in conjunction with shared weights as explained above. Here, we assign half of the loss to the sentiment task, and a quarter of the loss to both the paraphrase and similarity tasks. This ensures that updates made to the paraphrase and similarity networks are not disproportionately larger than those to the sentiment network. This also gives us superior performance compared to leaving the loss magnitudes untouched.

4.2.5 Layer-Wise Learning Rate Decay

Another strategy that we test is updating only the last few layers of BERT in finetuning, since not all layers necessarily impact learning for downstream tasks Zhang et al. (2021). We find that while this speeds up training, it does not perform as well as training on the entire BERT model. However, our final model uses a small learning rate of $\alpha = 10^{-5}$ for BERT parameters and a larger learning rate for our layers on top of BERT ($\alpha = 10^{-3}$), which optimizes task-specific learning without overwriting important semantic information already learned in the pretrained BERT model.

4.2.6 Random Sampling Training

In our approach to training, we do not train on all data points in each epoch because of our imbalance in available data (the Quora dataset for paraphrase detection is much larger than our other datasets). Instead, for each epoch we train on a random set of 4096 samples for each task, and sum up the losses before making an update to all parameters. This is similar to training on all data points in expectation and we find that it performs similarly to training on all data points in a single epoch, while significantly speeding up training to allow for more epochs and ultimately better performance given a time constraint. It also adjusts the weights of our model parameters in a more balanced way since updates include loss from each of the tasks instead of sequential updates for a single task at a time.

4.2.7 Extra Data

There are disparities in the number of training examples for each respective task which we highlight in **Data**. The paraphrase detection task contains roughly 130,000 more training examples relative to the other tasks, thus we gather more data online for the sentiment and similarity tasks, each of which is further described in **Data**. The purpose in gathering more data is to improve the generalizability of these tasks by training the model on more examples so that predictions on unseen data were more accurate. We find that training on more data from our new data sets does not seem to improve our performance noticeably and lengthens the time required for training.

4.2.8 SMART

We implement Adversarial Regularization according to Jiang et al. (2020), but find that it does not aid much in our training since we do not have issues with overfitting to the dev set. Although we do not use this method in any of our final results, we adapt code from namisan (2023) while working on this project and bootstrapping our SMART implementation.

5 Experiments

5.1 Data

We use a variety of data sets for each of the downstream tasks of interest. For the sentiment classification task, we use the Stanford Sentiment Treebank and Amazon Review data sets which contain 8,544 training sentence examples from movie reviews of varying sentiment rated on a 0 to 4 discrete scale and 18,000 training reviews of different Amazon products rated on a 0 to 4 discrete scale, respectively (GeeksforGeeks, 2021). For the paraphrase detection task, we use the Quora data set which has 141,506 training labeled question pairs indicating paraphrase presence. Finally, for the sentence similarity task, we use the SemEval STS Benchmark data set consisting of 6,041 training sentence pairs of varying similarity on a continuous scale from 0 (unrelated) to 5 (equivalent meaning). We also use the Sentences Involving Compositional Knowledge (SICK) (4,439 training samples) Face (2021) data set which has identical structure to the SemEval STS Benchmark. More data was used for the sentiment and similarity tasks to provide more examples for tasks that had less data samples.

5.2 Evaluation Method

For sentiment analysis and paraphrase detection, we compute the accuracy of our predicted labels compared to the true labels, with labels $y_i \in \{0, 1\}$ for paraphrase detection and labels $y_i \in \{0, 1, 2, 3, 4\}$ for sentiment analysis. For semantic textual similarity, we compute the Pearson correlation which measures the strength of the linear association between our predictions x_i and the true predictions y_i.

5.3 Experimental Details

We test a variety of model configurations for our experiments. These include:

- Using the CLS embedding vs. average output embedding from BERT (denoted CLS)
- Sharing weights for the paraphrase and similarity tasks (SW)
- Taking a weighted average of losses across tasks vs. a simple average (WL)
- Using extra data for the sentiment and similarity tasks (ED)

We train our model on each possible configuration of these settings ($2^4 = 16$ possible combinations) for 25 epochs. We fix our learning rate at $\alpha = 10^{-5}$ for BERT parameters and $\alpha = 10^{-3}$ for all parameters post-BERT.

5.4 Results

After running our multitask classifier across these 16 configurations, we find that the best-performing model uses the average output embedding, shares weights among the paraphrase and similarity tasks, uses a weighted average loss, and excludes extra data sets for the sentiment and similarity tasks. We also include our baseline model results for comparison.

Using extra data takes longer to train since processing large Amazon reviews is more time-intensive, which is why we break the tie for our best model

Model	SST	PARA	STS	Highest Average Dev Acc.
Milestone model, pretrained	0.206	0.381	0.108	0.232
Milestone model, finetuned	0.433	0.377	0.406	0.405
!CLS, !SW, !WL, !ED	0.493	0.826	0.579	0.633
!CLS, !SW, !WL, ED	0.516	0.835	0.715	0.689
!CLS, !SW, WL, !ED	0.510	0.834	0.575	0.640
!CLS, !SW, WL, ED	0.518	0.840	0.709	0.689
!CLS, SW, !WL, !ED	0.510	0.807	0.852	0.723
!CLS, SW, !WL, ED	0.521	0.808	0.847	0.725
!CLS, SW, WL, !ED	**0.527**	**0.795**	**0.854**	**0.725**
!CLS, SW, WL, ED	0.500	0.808	0.853	0.720
CLS, !SW, !WL, !ED	0.520	0.787	0.651	0.653
CLS, !SW, !WL, ED	0.511	0.790	0.565	0.622
CLS, !SW, WL, !ED	0.509	0.799	0.526	0.611
CLS, !SW, WL, ED	0.500	0.789	0.567	0.619
CLS, SW, !WL, !ED	0.502	0.765	0.816	0.695
CLS, SW, !WL, ED	0.495	0.759	0.828	0.694
CLS, SW, WL, !ED	0.504	0.773	0.823	0.700
CLS, SW, WL, ED	0.505	0.755	0.831	0.697

After selecting this model, our accuracies and correlation scores on the test leaderboard are as follows:

- **SST accuracy: 0.510**
- **Paraphrase accuracy: 0.809**
- **STS correlation: 0.860**
- **Overall score: 0.727**

These results show that using the average BERT embeddings performs higher than using the CLS embedding. Further, we see that across configurations, using extra data does not impact performance significantly. The most salient factor in boosted performance is the use of shared weights for the paraphrase and semantic similarity tasks. This was quite a surprising finding for us; initially, we thought it would be beneficial to train two separate feed-forward networks to learn the two different tasks. Upon intuiting that the two tasks are actually quite similar (both try to see how close in "meaning" two sentences are), we decided to investigate training a single feed-forward network for both tasks, which would allow for gradients to be added in the update step (and thereby train faster). Here we can see that our intuition was correct, as models using shared weights, holding all else equal, performed several percentage points higher on average.

To verify our model was not overfitting over a training period of 25 epochs, we also plotted the training and dev loss over time. We confirm that for 25 epochs, the dev loss does not begin to increase (on average), and thus our model is sound for evaluation on the test set.

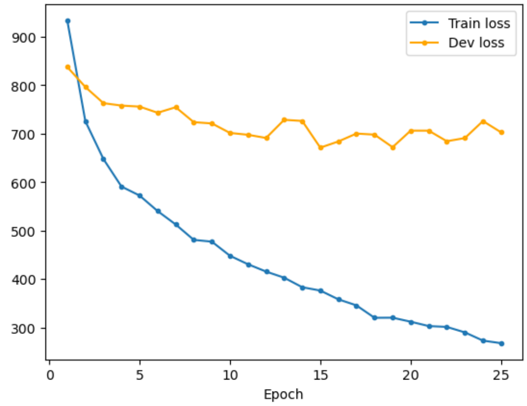

Figure 3: Training and dev losses over time for chosen model

6 Analysis

We take a look at confusion matrices and scatterplots for respective tasks below (dev set).

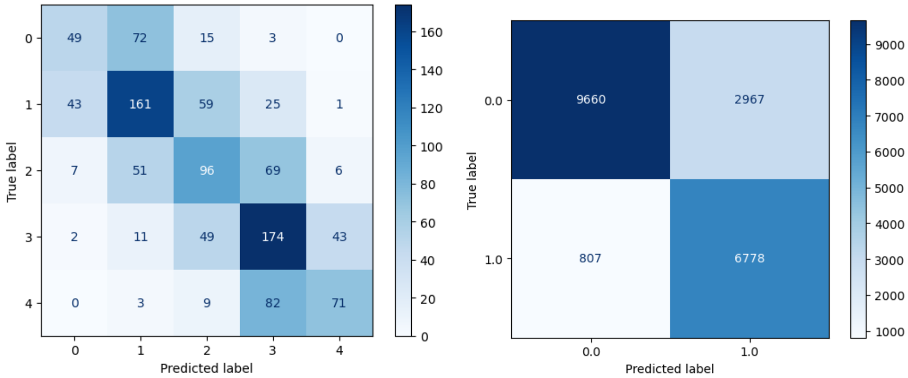

Figure 4: Sentiment Classification (left) and Paraphrase confusion matrices (right)

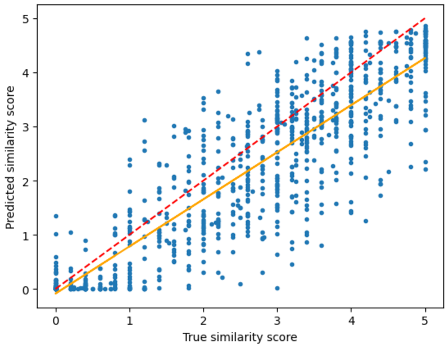

Figure 5: Semantic Textual Similarity Correlation

For the sentiment analysis task, we see that the most accurately predicted categories are 1 and 3. This is likely because these two categories have the most data points compared to all other categories, and thereby the networks are more optimized to categorize reviews with those labels. For the paraphrase detection task, we see that there is a false positive rate of about 0.235 and a false negative rate of about 0.106. Thus, it seems that the model is more likely to say that two sentences are paraphrases of one another when they are in fact not. This is interesting considering the fact that there are more examples of pairs of sentences that are not paraphrases of one another. Finally, for the semantic similarity task, we see that the line of best fit is under the perfect prediction line. This means that on average, our model predicts that sentences are less similar than they actually are.

We provide a few examples for each task in which our best model succeeds and where it fails, highlighting both its strengths and weaknesses.

6.1 Sentiment Analysis

Correct Prediction: A warm, funny, engaging film.
Sentiment Class: 4
Analysis: From this sentence, we can infer that our model can understand the polarity of certain words and can effectively categorize the strength of how positive they are.

Correct Prediction: It's fascinating to see how Bettany and McDowell play off each other.
Sentiment Class: 3
Analysis: From this example, we observe that our model can identify relatively neutral sentences well and picks up on the fact that a positive word ("fascinating") in a neutral sentence is classified as positive.

Incorrect Prediction: The film's hackneyed message is not helped by the thin characterizations, nonexistent plot and pretentious visual style.
Sentiment Class: True: 0. Prediction: 1.
Analysis: As shown by our confusion matrix, the model struggles with predicting negative sentiment examples. Given that the word "hackneyed" is an infrequently used word, we might conclude that the resulting embeddings do not contextualize its true meaning.

6.2 Paraphrase Detection

Correct Prediction: Sentence 1: Is wowecoin legal in India? Sentence 2: Is Antarvasna legal in India?
Paraphrase Class: 0
Analysis: From this pair of sentences, we observe that the model can correctly distinguish between two different nouns even if the sentences are nearly identical.

Correct Prediction: Sentence 1: Does reporting fake names on Quora do anything? Sentence 2: Is it worth it to report fake names on Quora?
Paraphrase Class: 1
Analysis: From this pair of sentences, we observe that asking the same question with the words scrambled still results in a correct prediction.

Incorrect Prediction: Sentence 1: What can you get as a customer of Star Alliance? Sentence 2: What are some ways to register with Star Alliance?
Paraphrase Class: True: 0. Predicted: 1.
Analysis: This example shows that our model may struggle when the verbs are not exactly synonyms with each other even if the sentences are very similar.

6.3 Semantic Textual Similarity

Since the labels for this task are continuous, we consider predictions within 0.05 to be correct.

Correct Prediction: Sentence 1: A cat is drinking milk. Sentence 2: A kitten is drinking milk.
Similarity Score: True: 4.0. Prediction: 4.03
Analysis: We see that our model is able to detect similarities among nouns that may be different but still represent very similar ideas.

Correct Prediction: Sentence 1: A chimpanzee is hurting a woman. Sentence 2: A man is driving a car.
Similarity Score: True: 0.0. Prediction: 0.001.
Analysis: This example showcases the strength our model has in detecting very dissimilar sentences that clearly are not related.

Incorrect Prediction: Sentence 1: Fire in Russian psychiatric hospital kills 38. Sentence 2: 38 feared dead in Russian psychiatric hospital fire.
Similarity Score: True: 4.2. Prediction: 2.71.
Analysis: We suspect that this error arises because our model has a hard time associating "feared dead" with "killed". This may be due to the fact that the model struggles to draw similarities between similar nouns and noun phrases.

7 Conclusion

This paper has showcased effective strategies for extracting meaningful sentence embeddings from pretrained BERT embeddings. We have shown that strategies such as weight sharing, weighted loss, and representing sentences by the average of their token embeddings leads to strong performance for downstream NLP tasks like sentiment analysis, paraphrase detection, and semantic textual similarity. We also showed that incorporating other techniques such as random sampling in training, adding Jaccard Similarity for sentence similarity related tasks, and using different learning rates across layers of our model can improve performance. While some of our methods like SMART (Jiang et al., 2020) and adding more training data didn’t improve our performance, they still provided important insights into multitask learning from BERT embeddings. Overall, we were proud of our model because it had a relatively simple architecture and performed well on the test set.

References

Hugging Face. 2021. Sick dataset. Accessed on March 20, 2023.

GeeksforGeeks. 2021. Amazon product reviews sentiment analysis in python. Accessed on March 20, 2023.

Manuela Nayantara Jeyaraj and Dharshana Kasthurirathna. 2021. MNet-sim: A multi-layered semantic similarity network to evaluate sentence similarity. *International Journal of Engineering Trends and Technology*, 69(7):181–189.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

namisan. 2023. mt-dnn.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Ranjan Satapathy, Shweta Pardeshi, and Erik Cambria. 2022. Polarity and subjectivity detection with multitask learning and bert embedding.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning.