Sentence-BERT-inspired Improvements to minBERT

Stanford CS224N Default Project

Raj Pabari
Department of Computer Science
Stanford University
rajpabari@stanford.edu

Abstract

Introduced in Devlin et al. (2018), Bidirectional Encoding Representations from Transformers (BERT) is a popular transformer-based model that generates contextual word embeddings that have empirircally performed well on many downstream tasks. These downstream tasks include sentiment analysis, paraphrase detection, and semantic textual similarity. However, the vanilla BERT model has some shortcomings which have been explored and improved in the literature. This project seeks to learn if specifically the Sentence-BERT extensions of the vanilla minBERT model yield the improvements that were demonstrated in the follow-on paper of Reimers and Gurevych (2019). Specifically, we implemented alternative pooling strategies and cosine similarity finetuning to see if we observe an improvement on the aforementioned downstream tasks. We conclude that different architecture choices are better suited for different tasks, and surprisingly achieved better performance on some of the tasks by using approaches different than those proposed by Sentence-BERT. These results suggest that, while one-size-fits-all architectures such as Sentence-BERT may achieve decent performance as a baseline, feature engineering and hyperparameter tuning can supercharge this performance in practice.

1 Key Information to include

Mentor: None, External Collaborators: None, Sharing project: No

2 Introduction

Due to their ability to effectively embed natural language into latent vector spaces, transformers (Vaswani et al., 2017) have seen success on many downstream natural language processing tasks and have risen greatly in popularity as a result. In this project, we will focus on three of these tasks:

- Sentiment Analysis ("Sentiment"): Given a sentence, classify its sentiment, for instance on an integer scale from 0 to 4 where 0 is "Negative," 2 is "Neutral," and 4 is "Positive."
- Semantic Textual Similarity ("STS"): Given two sentences, determine their degree of similarity, for instance on a scale from 0 to 5 where 0 is "Not at all Related," 2.5 is "Somewhat Related," and 5 is "Same Meaning."
- Paraphrase Classification ("Paraphrase"): Given two questions, determine (yes/no) if the questions are paraphrases of each other.

Despite the great power of transformers, the diverse natures of these tasks demand different architecture choices to achieve the best performance. However, these architecture choices are not obvious, and thus a rich literature has formed on this topic. The goal of this paper is to continue this literature of extending BERT to improve its performance in application. Inspired by the work of Reimers and Gurevych (2019) as a starting point, we experiment with various important architecture choices so as to better understand what works best for these downstream tasks.

3 Related Work

There have been myriad applications of the transformer and attention architectures since their introduction in Vaswani et al. (2017). One of the main such applications came in the form of Bidirectional Encoding Representations from Transfomers (BERT) in Devlin et al. (2018), which set a new state of the art for NLP tasks such as the aforementioned. Despite being quite successful, one drawback of BERT was that when it was queried to get embeddings of individual sentences, these embeddings were worse than GloVe or InferSent with respect to clustering or computation speed for instance (Pennington et al., 2014; Conneau et al., 2017). This was cause for concern, because despite the promise of attention-based embedding models, they were struggling to outperform classical NLP techniques on downstream tasks.

The motivation for the Sentence-BERT (SBERT) method of Reimers and Gurevych (2019) is to take advantage of other techniques such as Siamese and triplet networks to adapt BERT such that it produces better-clustered sentence embeddings while still performing well on downstream sentence classification tasks. Siamese and triplet networks are relevant for tasks involving multiple input sentences. Siamese networks produce embeddings with models that have tied weights, and triplet networks pair sentences with both positive and negative companion sentences to both "push" and "pull" embeddings to the right place. Despite not being the first paper to try such approaches, SBERT has received significant prominence since its release, in part due to the popular `sentence-transformers`[1] library that was released along with the paper. Thus, due to its popularity, the findings of the SBERT paper warrant reproduction and independent experimentation.

4 Approach

Our goal for this project is to understand how much of an improvement on downstream tasks we will see by extending the vanilla minBERT model using some of the methods of the Sentence-BERT paper. Specifically, we seek to learn if adding the pooling and cosine similarity layers at inference time yields the improvement that was demonstrated in Reimers and Gurevych (2019). Given the empirical success of Sentence-BERT, there is good reason to believe that the proposed changes will in fact yield an improvement on the downstream tasks. In this section, we detail the final architecture that was submitted to the test leaderboard for each of the three downstream tasks. These architectures were chosen as the result of the experiments in §5.

4.1 Sentiment Analysis Architecture

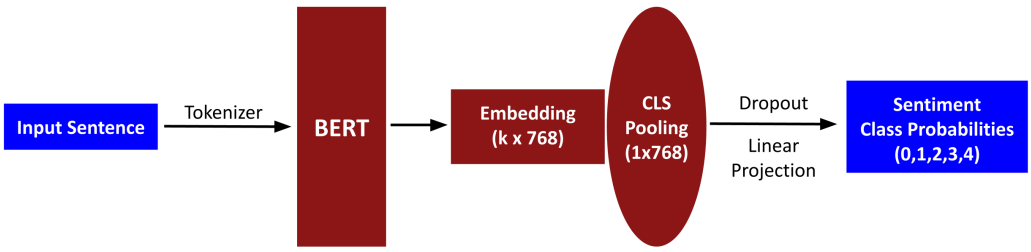

Figure 1: Final Architecture for Sentiment Analysis Task

Because it has only one input sentence, the sentiment analysis task has the simplest architecture. Given a tokenized input sentence, we feed it through the BERT model to receive its embedding in the 768-dimensional latent space (k depends on the length of the sentence). From there, we employ the CLS pooling strategy, which takes the embedding of the special `[CLS]` token as the embedding of the entire sentence. In the BERT architecture, this special token contains information about all the other tokens in the sentence, so taking the embedding of this token to be the embedding of the

[1] https://github.com/UKPLab/sentence-transformers/tree/master

whole sentence is sensible. From there, dropout is applied as a regularization technique, and then the pooled embedding is passed through a linear projection layer with weight matrix $W \in \mathbb{R}^{768\times5}$. The output of this is unnormalized probabilities that the input sentence belongs to each of the 5 sentiment classes. To compute the loss, the sigmoid function is used to normalize, then the cross entropy loss between the normalized probabilities and the true output labels is computed. Finally, in order to get a prediction from the model, an $\arg\max$ is taken over the normalized probabilities.

4.2 Semantic Textual Similarity Architecture

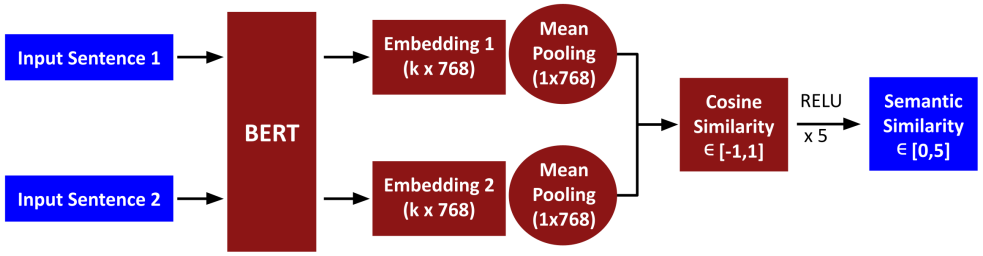

Figure 2: Final Architecture for Semantic Textual Similarity Task

For the semantic textual similarity task, we now have two tokenized input sentences. Inspired by the Siamese network structure of Reimers and Gurevych (2019), we feed them separately through the same BERT model to receive two separate embeddings. This time, we employ a mean pooling strategy, taking the embedding to be the mean across all of the tokens. From here, we take the cue of SBERT and compute a cosine-similarity score as our similarity the two pooled embeddings. Recalling, however, that cosine similarity is in the range $[-1, 1]$, and that the input data has semantic similarity scores in the range $[0, 5]$, we normalize the output of the cosine similarity by taking a `RELU` (transforming the score to the range $[0, 1]$) and multiplying by 5 (transforming the score to the range $[0, 5]$). Note that the cosine similarity between the vast majority of input sentences is positive (eg. in $[0, 1] \subset [-1, 1]$), so the `RELU` has very little distortion effect and is mostly present to ensure that we always output predictions in the correct range.

4.3 Paraphrase Detection Architecture

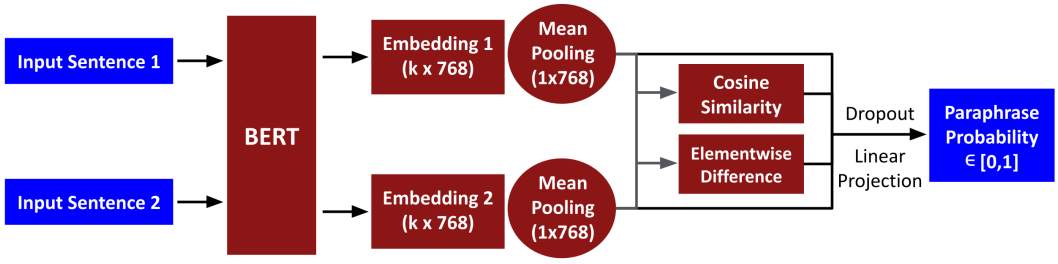

Figure 3: Final Architecture for Paraphrase Detection Task

The paraphrase detection task should be seen as a merge of the two preceding tasks, because it is a classification task like the sentiment task but it is also comparing the similarity between two input sentences like the STS task. As such, the architecture is a merge of the two preceding architectures. We use a linear projection layer with dropout in order to predict the unnormalized probability that the sentences are paraphrases of each other (this time, it is binary classification, so only one output is necessary). However, the input to the linear projection layer is a concatenation of both input

embeddings (call these $e_1, e_2 \in \mathbb{R}^{1\times 768}$), their cosine similarity $c \in \mathbb{R}^{1\times 1}$, and their absolute elementwise difference $|e_1 - e_2| \in \mathbb{R}^{1\times 768}$. Thus, the linear projection layer has weight matrix of dimension $W \in \mathbb{R}^{(3(768)+1)\times 1}$. Note that the cosine similarity between vectors e_1, e_2 is given by

$$\text{Cosine-Similarity}(e_1, e_2) = c = \frac{\langle e_1, e_2 \rangle}{\|e_1\|\|e_2\|} \in \mathbb{R}$$

The inspiration for the elementwise difference input feature comes again from the classification architecture of Reimers and Gurevych (2019) for classification tasks with two input sentences; the inspiration for the other input features comes from the preceding tasks. Finally, to output a prediction, we normalize the output of the linear layer by taking a sigmoid (this yields a probability $\in [0, 1]$), and round this to the nearest integer in order to get a prediction $\in \{0, 1\}$.

5 Experiments

5.1 Data

For the Sentiment Analysis task, we have two datasets of highly polar movie reviews – the Stanford Sentiment Treebank[2] (SST) and the CFIMDB dataset[3]. These are single input sentences, paired with output labels on an integer scale from 0 to 4 indicating the sentiment (0 - negative, 4 - positive). For the paraphrase detection task, we have the Quora Question Pairs[4] dataset, which contains pairs of input sentences and a binary classification label for whether or not they are paraphrases of each other. Finally, for the Semantic Textual Similarity dataset, we have that STS dataset from the shared task of the 2013 Conference on Lexical and Computational Semantics (Agirre et al., 2013). This dataset contains pairs of input sentences, and floating-point evaluations on a scale of 0 to 5 indicating how similar they are in meaning (0 - least similar, 5 - most similar).

5.2 Evaluation method

Given that we have a small number of classes for the sentiment analysis and paraphrase tasks (five and two respectively), we can compute a simple accuracy score as our evaluation method. Because of the continuous nature of the semantic textual similarity labels, however, an accuracy score does not make sense. Rather, we measure the Pearson correlation of the predicted labels with the true labels, and we seek a higher correlation.

5.3 Experimental details

For this project, we utilized four NVIDIA V100 GPUs on AWS EC2, as well as eight Graviton2 processors on AWS EC2 for CPU-intensive tasks such as evaluation, data preprocessing, and tokenization. For all experiments, unless otherwise specified we are fintetuning the base model `bert-base-uncased`[5] from Google, with a learning rate of 1×10^{-5}, dropout probability of 0.3, and a BERT hidden-layer size of 768. In the experiments, we report the maximum accuracy scores / Pearson correlation on the dev set from 10 epochs for the sentiment and STS tasks. For the paraphrase task, because the size of the dataset is prohibitively large (on the order of 100 times larger than the others), we report the maximum accuracy on the dev set after 3 epochs in experiments. After the experiments, for the final submission to the test leaderboard, we train each model on the best architecture that we found, for more epochs (until convergence). Note that each of the models is being finetuned for each of the tasks independently of the other tasks.

5.4 Baselines

The first part of this project involved implementing minBERT[6], specifically focusing on the functionality of the BERT model itself and the Adam optimizer for training. For the sentiment task,

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

[3]https://ai.stanford.edu/ amaas/data/sentiment/

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

[5]https://huggingface.co/google-bert/bert-base-uncased

[6]https://github.com/timothydai/minbert-default-final-project

given that our minBERT implementation is essentially the same as the final architecture, our baseline was already able to achieve performance comparable to the baselines provided in the default project handout. To replicate the provided "baseline" data from the handout, we performed a sweep over 10 different random seeds and evaluated our "minBERT" model on the sentiment task; the accuracy on the dev set is below:

Model	Accuracy	Stdev
Pretraining Baseline; SST dataset	0.390	0.007
Pretrained minBERT; SST dataset	**0.398**	0.003
Pretraining & Finetuning Baseline; SST dataset	0.515	0.004
Pretrained & Finetuned minBERT; SST dataset	**0.520**	0.005
Pretraining Baseline; CFIMDB dataset	**0.780**	0.002
Pretrained minBERT; CFIMDB dataset	0.759	0.013
Pretraining & Finetuning Baseline; CFIMDB dataset	0.966	0.007
Pretrained & Finetuned minBERT; CFIMDB dataset	**0.976**	0.006

Table 1: Sentiment dev accuracy of vanilla minBERT relative to provided baselines

Given the closeness of our vanilla minBERT model in performance to the provided baselines, this provides a sanity check that we have properly implemented BERT. Thus, given that our implementation is bug-free, we proceeded with naive implementations for the other downstream tasks for use as a baseline. The baseline architecture looks much like that detailed in §4.1, except for two-input-sentence tasks, there are two input sentences, so the linear projection layer receives a concatenation of their pooled embeddings, and has weight matrix of dimension $W \in \mathbb{R}^{2(768)\times 1}$. There are glaring issues with this architecture for the paraphrase and STS tasks; these issues will be investigated in detail in subsequent experiments. However for now, our naive implementation achieves the following on the tasks outlined in §1 (note henceforth we use only the SST dataset for the sentiment task):

Downstream Task	Score
Sentiment	0.520
STS	0.378
Paraphrase	0.763

Table 2: Performance of our baseline minBERT implementation on downstream tasks

5.5 Input Features

We seek now to improve the performance of our model on the STS and paraphrase tasks. In Reimers and Gurevych (2019), they suggest two extensions to the naive minBERT architectures implemented in the previous section. The first of these is, for tasks with two input sentences, to use the cosine similarity between the sentences as the prediction, rather than a linear projection of the pooled embeddings. The other suggestion is to include an elementwise difference input feature for classification tasks (recall STS is a regression task, so this applies to paraphrase) as outlined in §4.3.

We acknowledge that the paraphrase baseline performance is already quite good, so we consider some extensions of the proposed architectures from SBERT. We experiment with the following:

- Baseline: linear projection of embeddings of both sentences
- Proj + Cos-Sim: linear projection of embeddings of both sentences, concatenated with the cosine similarity between the embeddings
- Proj + Cos-Sim + Diff: linear projection of embeddings of both sentences, concatenated with the cosine similarity between the embeddings and the elementwise difference (more details in §4.3)
- Cos-Sim: use cosine similarity directly as the model's prediction, without linear projection (more details in §4.2)

The performance of each of these architectures is detailed below (with mean pooling, per results from next section). Despite STS being a regression task, we try all of the methods anyways –

Downstream Task	Baseline	Proj + Cos-Sim	Proj + Cos-Sim + Diff	Cos-Sim
STS	0.378	0.391	0.389	**0.799**
Paraphrase	0.763	0.758	**0.853**	0.619

Table 3: Performance of our baseline minBERT implementation on downstream tasks

The fact that the cosine similarity is the best performing by far for the regression task of STS is not too surprising. However, what is somewhat surprising is the fact that the projection + cosine similarity + elementwise difference architecture was the best performing for the paraphrase task. This is surprising because it is an extension of the architectures proposed in Reimers and Gurevych (2019), and is not directly proposed in the paper. The fact that this merged architecture that we came up with outperforms the tried-and-true architecture from SBERT is somewhat surprising.

5.6 Pooling

We recognize that all of the model architectures use a pooled version of the embeddings from BERT. Taking a cue from Reimers and Gurevych (2019), we conjecture that the pooling method would have a significant effect on the performance on the downstream tasks. Different pooling techniques better preserve different properties of the source embeddings. We test three different pooling strategies on each of the three tasks –

- CLS pooling: output the embedding of the special `[CLS]` token (more detail on the motivation behind this pooling strategy in §4.1)
- Mean pooling: output the mean of the embeddings of the tokens
- Max pooling: output the max of the embeddings of the tokens

We include the results from the experiment below, using the best set of input features as determined in the previous section –

Downstream Task	CLS	Mean	Max
Sentiment	**0.536**	0.505	0.526
STS	0.758	**0.799**	0.707
Paraphrase	0.820	**0.853**	0.844

Table 4: Testing different pooling strategies for each task

The only somewhat surprising result here is that Reimers and Gurevych (2019) found that mean pooling was most effective both for STS and for SNLI (a multi-class sentiment classification task), while we found that mean pooling was actually the worst-performing pooling strategy for the sentiment task. This may point to a more fundamental difference in the demands of our sentiment task and the SNLI task. Our results do otherwise corroborate the findings of Reimers and Gurevych (2019), though, demonstrating noticeable improvements from using mean pooling.

5.7 Dropout

There is an abundance of training data for the paraphrase task, but there is relatively little training data for the sentiment and STS tasks. Thus, one concern that we had was that our model may be overfitting to the training data after few epochs. Given our findings about the best set of input features, only the architectures of the sentiment task and the paraphrase task had regularization techniques in them still, these being in the form of dropout. Adding dropout to the STS architecture would be disastrous, because the cosine similarity is not resilient to having elements of the embedding vectors dropped out in the same way that would be sensible for dropping out before applying a linear projection layer. As a remark, because the paraphrase architecture also has a cosine similarity term, we compute the cosine similarity before applying dropout, then apply dropout to the concatenated vectors just before passing through the linear projection layer.

This being said, we wanted to experiment with the dropout probability on the data-constrained sentiment task to see if the dropout probability of 0.3 was truly the best choice –

Dropout Probability	Accuracy
0.2	0.526
0.3	**0.536**
0.5	0.518

Table 5: Performance on sentiment task with varying dropout probabilities

This experiment verified that boosting or decreasing the aggressiveness of our regularization did not improve the performance on sentiment task, which is not too surprising.

5.8 Performance on Classwide Leaderboard

For the final model, we use the best-performing architectures from the aforementioned experiments, and train them until convergence (as many epochs as necessary, varied per task). The finalized architectures were detailed above in §4. Upon submitting to the dev and test leaderboards, we received the following scores –

Evaluation Dataset	Sentiment	STS	Paraphrase
Dev Set	0.536	0.814	0.855
Test Set	0.527	0.802	0.855

Table 6: Scores achieved by final model on dev and test leaderboards

6 Qualitative Analysis

In this section, we take a deep dive into one common source of error on each of the tasks.

On the sentiment task, our model seems to be misled when the sentence expresses two different sentiments in one. For instance, the sentence "At a time when half the so-called real movies are little more than live-action cartoons , it 's refreshing to see a cartoon that knows what it is , and knows the form 's history ." begins by expressing a negative sentiment towards other movies that have been released at the same time, but ultimately turns this into a majorly positive for the movie that is being reviewed and thus received a rating of 4 (most positive) by the human evaluators. However, our model lost sight of the second half of the sentence and considered only the negative sentiment expressed in the first half, despite the negative sentiment being directed towards a different subject, and gave this sentence a 0 (most negative). This indicates (as we might expect), that the model does not truly understand what the sentence is saying, but rather is just pattern-matching.

On the STS task, we noticed that the model often tend to predict that sentences are more similar than they are. For instance, these two sentences received a score of 0 (not at all related) by the human evaluators, but a score of 3.22 (moderately related) from our model: "Let me share my opinion as Ukrainian citizen, from Kharkiv, eastern Ukraine." / "I am sure this is more phylosophical and ethical question than it seems." Overall, the model is very reluctant to output 0, despite many of the sentences in the dataset being not at all related. This is likely because of the nature of cosine similarity of real English sentences – in general, embeddings of real English sentences generally have fairly high cosine similarity with each other, no matter how the weights of the BERT model are changed, so upon taking the RELU and scaling we still obtain a nonzero prediction. A different architecture may help solve this over-shooting issue.

On the paraphrase task, we find that the model makes many errors on question pairs that use similar words but mean different things. For instance, the pair "What brings people to Quora?" / "What brings people back to Quora?" is labeled as not being paraphrases of each other, while our model classified the two as paraphrases. This indicates that the model is not sophisticated enough to understand which words, when added to a sentence, substantially change the semantic meaning of the question, to the point where the addition of that word would make the modified sentence not be a paraphrase of the original sentence.

7 Conclusion

In this project, we began by implementing a baseline version of minBERT, extended the architecture to improve its performance on the downstream tasks of §1 by taking inspiration from Sentence-

BERT (Reimers and Gurevych, 2019), and then implemented some of our own extensions with varying degrees of success. We determine which of the architecture choices are most effective in §5, and the finalized architectures are outlined in detail in §4. Overall, we were able to achieve good performance on the downstream tasks after a sufficient amount of finetuning and experimentation. The final architecture for the sentiment task ended up being essentially the same as our baseline implementation, for the STS task we ended up with something similar to SBERT, and for the paraphrase task we ended up with a novel architecture.

The most notable takeaway from this paper is that a one-size-fits-all approach is not optimal for transformer-based models to perform well on downstream tasks. Rather, some attention needs to be paid to the specific demands of the task, and then with proper experimentation one can likely improve over the baseline architectures. We did not expect that any of the novel architectures we proposed would outperform those of Sentence-BERT, but to our surprise, we were able to propose a set of input features that was not suggested in the Sentence-BERT paper and did achieve better performance. Ultimately, we achieved our best performance on the paraphrase task by starting with the Sentence-BERT architecture and extending it and building off of it based on our knowledge of the task; this appears to be a generally good problem-solving approach for tackling future problems with deep learning.

In future work, it may be interesting to explore training the models in tandem, or how the learnings from one downstream task could transfer to learning for other downstream tasks. We considered the tasks completely independently, but in fact they are somewhat related – for instance, the STS and paraphrase tasks both pertain to the "similarity" of the two input sentences.

References

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*, pages 32–43. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.