SimCSE Lessens your Need to Seek BERT's Attention

Stanford CS224N {Default} Project

Andre Klawa
Department of Computer Science
Stanford University
aklawa@stanford.edu

Abstract

The goal of this project is to apply recent research findings to finetune the sentence-embeddings of a single BERT model such that it performs well on three semantic tasks. We demonstrate a computationally efficient way to compare sentence embeddings from our model, which delivers a performance on par with BERT models finetuned for specific downstream tasks.

1 Key Information to include

- Mentor: none
- External Collaborators (if you have any): none
- Sharing project: none

2 Introduction

This report is part of the CS224N Winter 2023 default final project. The task was to implement a major part of the logic of BERT and to find and apply methods from recent research papers to finetune BERT in such a way that it would simultaneously perform well on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). Our BERT implementation is based starter code adapted from the minBERT assignment developed for Carnegie Mellon University's CS11-711 Advanced NLP class. Despite minBERT being a minimal implementation, we are using pretrained weights from Huggigface's "bert-base-uncased" for our model.

When optimizing a neural network for multiple tasks, the objectives of different tasks may conflict with one another. Optimizing for one task can deteriorate the performance of another task, a phenomenon known as catastrophic forgetting (Chen et al., 2020a). In one of our experiments we will look into what happens if we simply finetune minBERT on all three tasks. There are multi-task leaning approaches to lessen the problem. We refrain from finetuning BERT on the three given downstream tasks. Instead our approach is to employ algorithms, that make sentence-embeddings more easily distinguishable for regression functions.

SimCSE is a contrastive learning framework that is built on the premise to make dissimilar sentence-embeddings more distinguishable. Notably, SimCSE has demonstrated improved state-of-the-art performance on many benchmarks involving sentence-embeddings. We are using SimCSE as the sole set of algorithms to finetune BERT's weights.

When dealing with sentence-embeddings in BERT, it is important to understand cross-encoders and bi-encoders. There are two ways to compare sentences using BERT: cross-encoders and Bi-encoders. Cross-encoders usually outperform Bi-encoders on regression-pair tasks(Reimers and Gurevych, 2019). In a cross-encoder we feed BERT two input sentences separated by a [SEP] token. BERT can apply attention across both sentences and outputs a single CLS embedding for both sentences. This resulting CLS embedding is no longer a sentence-embedding but a vector- encoding of two sentences. For the final decision the CLS gets fed to a regression layer. Though superior in accuracy,

this process is computationally very expensive and therefore infeasible for most downstream tasks: "Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (65 hours) with BERT." (Reimers and Gurevych, 2019)

In the Bi-encoder paradigm one sentence at a time gets fed to BERT to obtain a CLS embedding for a single sentence. The resulting sentence-embeddings can then be compared with each other to determine which are similar. The comparisons can be performed via regression or via cosine-similarity. A comparison via cosine-similarity of 10,000 sentence embeddings can now be performed in seconds and not requiring GPUs, versus the aforementioned 65 hours with a cross-encoder. The sentence embeddings can be stored in a database for later reference. However, the sentence-embeddings coming out of plain BERT are not suitable for cosine-similarity comparison. For this BERT must be finetuned with a framework like SBERT or SimCSE. Regression is more accurate than cosine-similarity, but computationally more expensive.

3 Related Work

BERT has achieved state-of-the-art performance on sentence comparison tasks, but its inference costs are often prohibitive for practical applications. SBERT (Reimers and Gurevych, 2019) was developed as a framework to fine-tune BERT to create sentence-embeddings that can be compared using cosine similarity, while maintaining BERT's accuracy. In computer vision, Chen et al. (2020b) introduced SimCLR, a contrastive learning framework. SimCSE (Gao et al., 2021) builds on the idea of fine-tuning BERT to improve sentence embeddings from SBERT and applies the learning algorithm from SimCLR to further enhance its performance.

4 Approach

4.1 SimCSE

SimCSE is a contrastive learning framework developed to fine-tune BERT to improve the quality of its sentence embeddings such that more accurate comparisons through methods such as cosine similarity or regression are possible. It that has improved state-of-the-art performance on many sentence-embedding benchmarks. BERT's "learned embeddings occupy a narrow cone in the vector space" Gao et al. (2021) which makes it difficult to distinguish dissimilar sentences from each other. SimCSE pushes dissimilar embeddings further away from each other, making them easier to distinguish, while keeping similar ones close. In mathematical terms, we use alignment to measure how close similar items are

$$\ell_{align} = \mathrm{E}_{(x,x^+)\sim P_{pos}} ||f(x) - f(x^+)||^2 \qquad (1)$$

and uniformity to measure how well the embeddings are uniformly distributed

$$\ell_{uniform} = log\ \mathrm{E}_{x,y\sim P_{data}} e^{-2||f(x)-f(y)||^2} \qquad (2)$$

In contrastive learning, a model learns to predict whether two items are similar or not. It needs positive examples i.e. two items that are similar (a "positive pair"), as well as negative examples. Unsupervised SimCSE obtains two different sentence-embeddings for the same input sentence by feeding the same sentence to BERT twice; but with a different dropout mask on each run. These two sentences become "positive pairs", while all other sentences in the same batch act as negative examples. For training, Unsupervised SimCSE uses the following cross-entropy loss function:

$$\ell_i = -log\ \frac{e^{sim(h_i^{z_i}, h_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{sim(h_i^{z_i}, h_j^{z_j'})/\tau}} \qquad (3)$$

Where h_i, h_j are BERT sentence-embeddings, $sim(h_i^{z_i}, h_i^{z_i'})$ is the cosine similarity, and $z, z^{'}$ are different dropout masks. Supervised SimCSE is being trained on natural language inference (NLI) datasets. NLI datasets are made up of entailment pairs, neutral pairs, and contradiction pairs. We now have (h_i, h_i^+, h_i^-) where $h_i$is the premise, $h_i^+$is the entailment, and h_i^- is the contradiction. The loss function for Supervised SimCSE is:

$$\ell_i = -log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_j, h_j^+)/\tau} + e^{sim(h_j, h_j^-)/\tau}} \qquad (4)$$

4.2 SimCSE Algorithm Implementation

We developed a class to train a minBERT model for SimCSE. We implemented both the unsupervised and supervised SIMCSE training algorithms according to equations (3) and (4). In contrast to the original SimCSE implementation, we omitted the Masked Language Model (MLM) training task, which has been shown to significantly improve SimCSE's benchmark scores.

4.3 Neural Network Implementation

We developed a class that hosts a minBERT model and network layers for each of the three given downstream tasks, ensuring that training for one task does not affect the other tasks. Our downstream-task implementations are as follows:

- **Sentiment Analysis:** One CLS token as input, one hidden layer with a dropout rate of 0.1 leading to a softmax function.
- **Paraphrase Detection:** Two concatenated CLS tokens as input, with two hidden layers using ReLU activation.
- **STS:** Two concatenated CLS tokens as input, with three hidden layers using ReLU activation.
- **Cross-Encoder:** Two concatenated CLS tokens as input, with one hidden layer. There is one cross-encoder layer for the paraphrase task and one for STS.

Our implementation can load a Huggingface model, a SimCSE supervised model, or a SimCSE unsupervised model. Since BERT cross-encoders are known to outperform other implementations on benchmarks, we implemented a cross-encoder function for each Paraphrase Detection and STS tasks. We originally implemented the STS task using a single linear layer with an input size of two CLS embeddings and an output size of 1. This gave us a maximum Pearson score of 0.36 for supervised SimCSE. We tested our implementation on an original SimCSE model from Huggingface and got the same poor results. Once we implemented a multi-layer network to compare two CLS tokens, we obtained a Pearson score of 0.715. We suspect that there may be a numerical issue with PyTorch in this implementation. Interestingly, we found that non-SimCSE embeddings were less susceptible to this problem.

In total, we have the following models:

- **SimCSE-supervised:** minBERT finetuned on SimCSE supervised.
- **SimCSE-unsupervised:** minBERT finetuned on SimCSE unsupervised.
- **BERT-bi-encoder:** minBERT using the regular layers for paraphrase and STS.
- **BERT-cross-encoder:** minBERT using the cross-encoder layers for paraphrase and STS.
- **BERT-finetuned-single:** We finetuned three minBERT cross-encoder models on single tasks exclusively, one for each of the STS, paraphrase, and sentiment tasks.

4.4 Baselines

At the low end, BERT bi-encoder serves as a baseline. On the high end, we finetuned three BERT finetuned-single models, each for a single task, to determine what might be achievable with the given training data.

5 Experiments

5.1 Data

The training data for the downstream tasks was provided by the CS224N starter code (ber, 2023). To finetune BERT via SimCSE we use the original SimCSE training data(Gao et al., 2021). For the unsupervised part we use wiki1m_for_simcse, which consists of 10^6 sentences from English

Wikipedia. For the supervised part we use nli_for_simcse, which consists of 275,000 examples, where each example consists of a sentence, a positive example, and a negative example.

5.2 Evaluation method

We use the f1 score and Pearson score as metrics:

- The **Pearson score** is a measure of linear correlation between two sets of data. We are computing the Pearson between the predicted values and the gold labels on STS tasks.
- The **f1 score** is a measure of a test's accuracy and is computed as the harmonic mean of precision and recall.

5.3 Experimental details

5.3.1 Hardware

We trained all our models on an NVIDIA 3090 GPU with 24GB of RAM.

5.3.2 Training of SimCSE Unsupervised

We finetuned one minBERT model on the original SimCSE training data (wiki1m_for_simcse), which consists of 10^6 randomly sampled English Wikipedia sentences. To ensure consistency with the original SimCSE implementation, we used the same training parameters and limited the maximum sentence length to 32 words like the original SimCSE implementation. Failing to do so caused our GPU to run out of memory. We trained minBERT for one epoch using a batch size of 64, a learning rate of 3e-5, and a dropout probability of 0.1, which is a critical component of the unsupervised SimCSE approach. The training took 2604 seconds. We must mention that our SimCSE algorithm implementations do not include any safeguards against overfitting. The original implementation employs Senteval as a safeguard to obtain the best model. As such, our minBERT model may suffer from overfitting.

5.3.3 Training of SimCSE Supervised

Similarly to the unsupervised part, we finetuned one minBERT model with supervised SimCSE using the original training data and original training parameters. In this part we use nli_for_simcse, which consists of 275,000 examples, where each example consists of a sentence, a positive example, and a negative example. We trained the model for 3 epochs with a batch size of 128, dropout of 0.1, and learning rate of 5e-5. The training took 2645 seconds. As in the unsupervised case, there was no protection against overfitting.

5.3.4 Training of Downstream Models

We trained all our models with the same configuration, at 10 epochs for the sentiment task, 20 epochs for STS, and 5 epochs for the paraphrase task. We used a learning rate of 1e-3 and a batch size of 64. To avoid overfitting model checkpoints were only saved if accuracy of the dev set improved. Note that we never finetuned minBERT. The BERT cross-encoder-finetuned models, which were each finetuned for one task each were trained at a learning rate of 1e-5. We used the same task specific number of epochs as for the downstream models.

5.4 Results

We report the accuracy for the paraphrase and the sentiment analysis tasks, while we use the Pearson score for the STS task.

5.5

On the official CS224N "Test Set Leaderboard" we achieved the following scores with a SimCSE-supervised model:

- **SST test Accuracy:** 0.500

Table 1: Evaluation metrics for various models

Model	Paraphrase Acc.	Sentiment Acc.	STS Corr.
SimCSE-unsupervised	0.809	0.449	0.696
SimCSE-supervised	0.819	0.479	0.715
BERT-bi-encoder	0.771	0.467	0.524
BERT-cross-encoder	0.722	0.468	0.561
BERT-finetuned-single	0.884	0.528	0.869

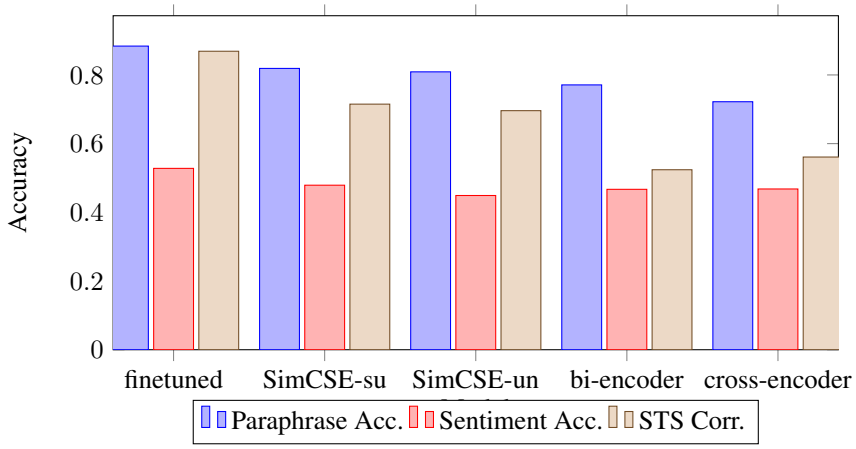

Figure 1: Accuracy comparison of different models on paraphrase, sentiment, and STS tasks.

- **Paraphrase test Accuracy:** 0.814
- **STS test Correlation:** 0.700
- **Overall test score:** 0.671

6 Analysis

6.1 SentEval

SentEval (Conneau and Kiela, 2018) is a widely used toolkit for evaluating the quality of sentence embeddings. Similar to our implementation for the three given downstream tasks, SentEval applies a regressor to sentence embeddings and provides a comprehensive set of tasks for evaluation, including sentiment analysis, STS, and paraphrase detection.

6.2 Results from the SimCSE Paper for Comparison

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC
BERT-[CLS]embedding	78.68	84.85	94.21	88.23	84.13	91.40	71.13
SimCSE-BERTbase-unsupervised	81.18	86.46	94.45	88.88	85.50	89.80	74.43
SimCSE-BERTbase-supervised	82.69	89.25	94.81	89.59	87.31	88.40	73.51

Table 2: Performance comparison of models on SentEval tasks. The data was condesed from table E.1 in Gao et al. (2021)

To get a feel for what to expect from our experiments we reduced table E.1 from the original SimCSE paper (Gao et al., 2021) to models that are equivalent to ours. To produce the original table Gao et al. (2021) trained regression classifiers for their sentence-embeddings, following the SentEval protocol, which is similar to our downstream regression approach. The tests given in the table are all part of

the SentEval battery. BERT-[CLS]embedding is comparable to our BERT-bi-encoder model. MR, CR, and SST are sentiment analysis tasks, while MRPC is a paraphrase detection task.

Your report should include *qualitative evaluation*. That is, try to understand your system (e.g. how it works, when it succeeds and when it fails) by inspecting key characteristics or outputs of your model.

6.3 Sentiment Analysis

Table 2 shows the SentEval tasks MR, CR, and SST, which test sentiment evaluation. We can see that the SimCSE models only marginally outperformed vanilla BERT on these tasks. In our experiments, unsurprisingly, SimCSE supervised performed the best with an accuracy of 0.479, slightly better than plain BERT at 0.467. However, SimCSE unsupervised performed the worst at 0.449. As mentioned earlier, our implementation of the SimCSE algorithms did not have protection against overfitting like the original implementation. Therefore, we suspect that our implementation to perform worse than the original. If the margins are small, as in this case, minor defects can quickly become noticeable. SimCSE was the best performer at 0.479. However, there is still a noticeable gap to the models finetuned for a single task, scoring at 0.528.

6.4 Paraphrase Detection

On the paraphrase detection task we should expect SimCSE supervised to outperform all others. Paraphrases can use synonyms and thus have little lexical overlap with their original sentences. Supervised SimCSE is trained on triplets consisting of a premise, an entailment, and a hard negative. The data is obtained from human-produced NLI datasets, which consist of a given sentence (premise), a statement that is true given the premise (entailment), a statement that might be true given the premise (neutral), and a statement that is false given the premise (contradiction). SimCSE uses the NLI contradiction as the hard negative, giving it a chance to learn that two sentences with great lexical overlap may not be the same. This could benefit paraphrase detection, as two sentences that differ only in word order may have different meanings. If we had had more time, we would have tested supervised SimCSE and plain minBERT on the HANS dataset to explore this further and to investigate whether SimCSE introduces any unwanted heuristics. As expected, SimCSE supervised performed best with a score of 0.819, surprisingly closely followed by SimCSE unsupervised at 0.809. Interestingly the BERT cross-encoder performed worse at 0.722 than our vanilla BERT bi-encoder at 0.771, considering that a cross-encoder may apply attention across both input sentences. As described in the STS discussion, we double checked our cross-encoder implementation. Paraphrase detection is the category in which the gap between SimCSE supervised, with a score of 0.819, and finetuned model's score of 0.884 is the smallest. A contributing factor may be that the NLI data used to train SimCSE is similar in nature to the paraphrase detection training data.

6.5 Semantic Textual Similarity (STS)

In theory, the cross-encoder should beat all other implementations, since it can leverage attention across both input sentences(Reimers and Gurevych, 2019). Our single task cross-encoder, finetuned on STS obtained a Pearson score of 0.869, which agrees with Reimers and Gurevych (2019) results. If we do not finetune minBERT on the STS task, supervised SimCSE is a clear winner with a score of 0.715, followed by unsupervised SimCSE at 0.696, while the cross-encoder scores a mere 0.561. The huge performance gap between 0.869 for the finetuned cross-encoder and 0.561 on the non-finetuned cross-encoder model, makes us question if attention across two sentences is an important factor as Reimers and Gurevych (2019) suspects.

6.6 Finetuning BERT on Downstream Tasks and Catastrophic Forgetting

We finetuned a single BERT-cross-encoder model on all tasks to investigate the possibility of encountering catastrophic forgetting. In Table 3, we compare this model to the models that were finetuned on a single task and to the non-finetuned BERT-cross-encoder. The paraphrase task had by far the largest dataset and was trained last. Therefore, it is not surprising that the accuracy for this task is the same as that of the single task models. The performance on the STS task is far below that of the single task models but still close to that of the BERT-cross-encoder, which has a score of 0.561. The sentiment task, on the other hand, had a relatively small dataset and was trained first. Still

Model	Paraphrase Acc.	Sentiment Acc.	STS Corr.
BERT-cross-encoder	0.722	0.468	0.561
Finetuned on single tasks	0.884	0.528	0.869
Finetuned on all tasks	0.884	0.135	0.404

Table 3: Finetuned on single vs. all tasks

the drop in performance to 0.135 is dramatic compared to the non-finetuned BERT-cross-encoder, which performs at 0.468. Without proof we suspect that the performance loss was due to training the paraphrase tasks. Based on this belief, it comes as a surprise that finetuning BERT on a small dataset for the paraphrase tasks with 141,498 examples can have such a profound impact on BERT itself.

7 Conclusion

In conclusion, we successfully implemented our own BERT model and utilized SimCSE, a set of contrastive algorithms to improve BERT's sentence-embeddings to perform well on three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). SimCSE performed almost on par with fine-tuning BERT on a single downstream tasks. When fine-tuning BERT on multiple downstream tasks we encountered significant catastrophic forgetting. It was surprising to observe the considerable impact a small dataset of 100K examples had on BERT when fine-tuning. Our experiments also raised interesting questions regarding the effectiveness of cross-encoders on benchmarks, specifically whether their attention apllied to two input sentences at once or their pretraining on NLI datasets led to their superior performance. Further research examining SimCSE with HANS to identify any unwanted heuristics in its sentence-embeddings would also be valuable.

References

2023. Cs 224n: Default final project: minbert and downstream tasks. `https://web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf`. Accessed on March 18, 2023.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020a. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020b. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.