NLP Project Final Report - Team 2

Replicating and Extending SimCSE: Simple Contrastive Learning of Sentence Embeddings

Aditya Srikanth
NYU CDS (2023)
aks9136@nyu.edu

Andre Chen
NYU CDS (2023)
alc9635@nyu.edu

Jin Ishizuka
NYU CDS (2023)
ji721@nyu.edu

1 Problem Introduction

In the following report we describe efforts to replicate and extend the paper "SimCSE: Simple Contrastive Learning of Sentence Embeddings" by Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE advances state-of-the-art universal sentence embeddings, a fundamental problem in natural language understanding where improvements translate downstream to important NLP tasks like document similarity analysis, question answering, and chatbot systems. In particular, the authors of the paper primarily sought to exceed current benchmarks on standardized semantic textual similarity (STS) tasks set by SOTA transformer-based models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).

In our two extensions of this paper, we conduct experiments in order to test two hypotheses. We describe these extensions in later sections, but list the hypotheses below (in order) for clarity:

1. SimCSE embeddings will lead to stronger performance over traditional BERT embeddings on a sentiment classification task.

2. Given concatenated SimCSE and BERT embeddings, a model will rely more on SimCSE embeddings when learning to perform sentiment classification.

2 SimCSE Overview

The original SimCSE paper proposes two contrastive learning frameworks to generate improved sentence embeddings. Both frameworks involve passing text data through a SOTA sentence encoder (e.g. BERT, RoBERTa, etc.), then updating the encoder weights to minimize a contrastive learning objective. The two frameworks differ in what constitutes positive and negative examples during contrastive learning. In the unsupervised case, contrastive loss is minimized between two embeddings of each sentence in the dataset (10^6 randomly sampled sentences from English Wikipedia), where both embeddings are generated with noise using standard dropout. In the supervised case, contrastive loss is minimized between the embeddings of entailment pairs from Natural Language Inference (NLI) datasets. Our work focused on replicating experiments from the supervised case, i.e. evaluating embeddings generated from supervised SimCSE on several of the STS datasets and validating that we could achieve similar improvements over benchmarks set by BERT and RoBERTa. Throughout this paper, we often refer to the embeddings generated from the supervised SimCSE framework as SimCSE Embeddings. We describe replication methodologies and results in the following section.

3 Replication of SimCSE Experiments

To validate experimental results for the supervised version of SimCSE on STS tasks, we replicated experiments from the original SimCSE paper using three different versions of supervised SimCSE. These versions, as described in the original paper, varied in terms of the encoder used to initialize input sentence embeddings prior to fine-tuning on entailment / contradiction pairs in the aforementioned NLI datasets. Specifically, these encoders were BERT Base Case, RoBERTa Base Case, and RoBERTa Large Case (RoBERTa Lg).

The replication process was conducted by forking the published code repository from the authors of the SimCSE Paper. We then replicated the data import to ensure that the data used was clean and independently supported. Finally, we updated the code to reflect the most up-to-date packages, making minor syntax changes to ensure code completion. We output our results using the same methods

in the SimCSE codebase.

The metrics collected are the same metrics from the aforementioned paper. In our tables, we shorten STSBenchmark to STSB and SICKRelatedness to SICKR. (Our tables are reported below as Table 1 and Table 2).

Metric	BERT	RoBERTa	RoBERTa Lg
STS12	75.30	76.53	77.46
STS13	84.67	85.21	87.27
STS14	80.19	80.95	82.36
STS15	85.40	86.03	86.66
STS16	80.82	82.57	83.93
STSB	84.25	85.83	86.70
SICKR	80.39	80.50	81.95

Table 1: STS performance results as reported in the original SimCSE paper

Metric	Bert	RoBERTa	RoBERTa Lg
STS12	75.30	76.53	77.45
STS13	84.67	85.20	87.27
STS14	80.19	80.95	82.36
STS15	85.40	86.03	86.66
STS16	80.82	82.56	83.93
STSB	84.25	85.83	86.70
SICKR	80.39	80.50	81.95

Table 2: STS performance results as obtained in our independent validation

4 Extension 1: Twitter Sentiment Analysis

As an extension to the findings in the initial SimCSE paper, we compared the performance of supervised SimCSE sentence embeddings against BERT sentence embeddings on a weakly supervised sentiment classification task. Specifically, we obtained a dataset of "sentences" by scraping 10,000 tweets using Twitter's search API and computing polarity scores (in the range of [-1,1]) for each tweet using TextBlob, a Python library that leverages Natural Language Toolkit (NLTK) for basic NLP tasks. We describe our classification task as "weakly supervised" because our class labels were assigned by binning TextBlob-generated sentiment polarity values in the range of [-1, 0) as negative sentiment, values in (0, 1] as positive sentiment, and values equal to 0 as neutral sentiment. This method provided relatively balanced class sizes of 2,200 / 3,200 / 4,600 for negative, neutral, and positive sentiment classes.

In the following sections related to extensions, we refer to a "feed-forward neural network or "sentiment classifier" interchangeably. We note here that this network was intended to serve as a comparative tool between SimCSE and BERT. As such, the focus in our extensions was not to produce the optimal network for conducting sentiment classification. However, we explored a small search space of hyperparameters when building our SimCSE and BERT sentiment classifiers to ensure our design choices were reasonable. We also implemented our sentiment classifiers in PyTorch using Cross Entropy Loss and the Adam optimizer.

In our comparison of SimCSE against BERT for sentiment classification, we used the huggingface transformers Python library to generate two sets of sentence embeddings for our tweets data: one using the pretrained supervised version of SimCSE's encoder (initialized on uncased BERT base embeddings), and one using the pretrained uncased BERT base encoder itself. We froze the embeddings from both methods and fed them as inputs into two different feed-forward neural network classifiers, which we independently trained. We additionally obtained locally optimal configurations for each of the two classifier models by conducting hyperparameter tuning over 45 different combinations of hidden size, learning rate, and number of hidden layers. Figure 1 shows validation accuracy curves computed over each training epoch at the top 10 hyperparameter configurations for BERT and SimCSE respectively, with the best configuration highlighted in blue. For SimCSE, we selected a hidden dimension of 512, learning rate of 5e-4, and 1 hidden layer as our final hyperparameter configuration. For BERT, we used a hidden dimension of 256, learning rate of 5e-4, and 2 hidden layers. Optimal hyperparameter configurations were selected based on which yielded the highest validation accuracy at any point during training.

After hyperparameter tuning, we compared sentiment classification accuracy on our final holdout set at the respective optimal configurations for BERT and SimCSE. To ensure our results were robust, we computed mean test accuracy across 6 replicates per method, where in each replicate we generated random 70% train / 30% test splits, trained SimCSE and BERT feedforward networks using their optimal hyperparameters, and computed

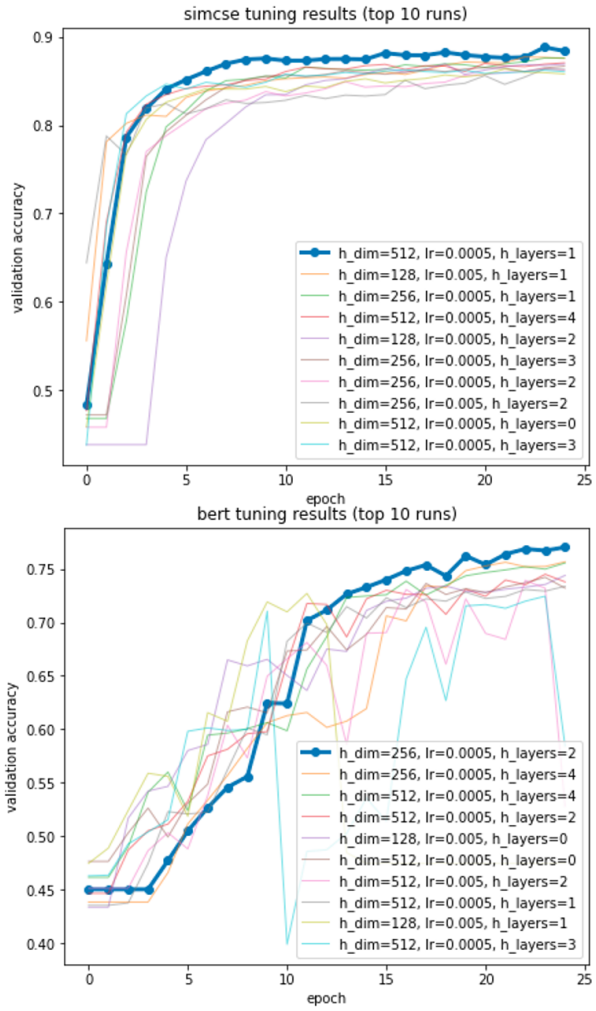

Figure 1: Validation accuracy curves obtained from training for the top 10 hyperparameter configurations for SimCSE (top) and BERT (bottom)

their respective test accuracies. We also conducted these experiments over a range of dataset sizes from 1,000 to 10,000 to determine the sensitivity of each sentiment classifier model to training size. Figure 2 shows the results of this sensitivity analysis. Our results in Figure 1 and Figure 2 highlight two key advantages our SimCSE sentiment classifier has over the baseline BERT model. The first is that we achieve higher maximum mean test accuracy with SimCSE compared to BERT. Our SimCSE sentiment classifier achieves a maximum mean test accuracy of 0.88, while its BERT-only counterpart achieves 0.74. The second advantage is that SimCSE achieves close to maximum performance with less training time and training data than BERT. We can see that the SimCSE sentiment classification model reaches near-maximum performance between epochs 5 and 10 (Figure 1) and a dataset size of roughly 4,000 (Figure 2), while the BERT classification model validation performance does not begin to level off until epoch 15 and a dataset size of 6,000.

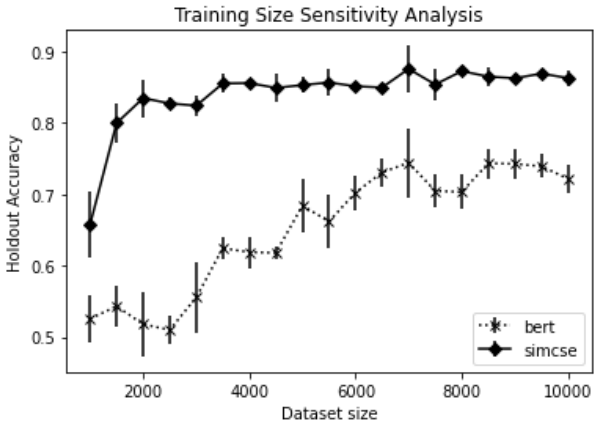

Figure 2: Sensitivity analysis of SimCSE and BERT-based sentiment classification model accuracies on training data size

5 Extension 2: Comparison of BERT and SimCSE Feature Importances

As a secondary extension to the findings in the initial SimCSE paper, we employed another method of comparing the usefulness of supervised SimCSE sentence embeddings and the usefulness of BERT sentence embeddings. Similar to Extension I, we utilize the Twitter sentiment analysis task as our means of extrinsic validation of our two embeddings. However, in this extension, instead of training and comparing two separate models, we instead train just a single model and analyze the impact of feature permutation in order to make our comparison.

We began by taking the sentence embeddings for both SimCSE and BERT (see Extension 1: Twitter Sentiment Analysis section for detail). Next, we concatenated the embeddings such that every sentence in our dataset was represented by both SimCSE and BERT embeddings in a single row. Then, we trained the same feed-forward neural network from Extension I.

The key contribution of this extension in how we evaluated this network. Instead of traditional validation, we initialized two validation datasets stemming from our original validation dataset (our concatenated SimCSE-BERT sentence embeddings). In one dataset, we permuted the sentence embedding halves belonging to SimCSE. In our second dataset, we permuted the sentence embeddings halves belonging to BERT. In both cases, the non-

permuted embeddings were still retained. To provide additional clarity, when we permuted the embeddings, we randomized the embeddings column-wise in order to make the embeddings effectively nonsensical as interpreted by the network (while still maintaining the original numeric range of data). We then ran our evaluation pipeline for each of our two validation datasets.

The intuition of this approach is that we begin by giving both the unpermuted SimCSE and BERT embeddings to our network. Then, we allow our network to learn sentiment classification using any combination of SimCSE and/or BERT features. If the network finds more value in using one embedding type (SimCSE or BERT) over another, then when we permute that embedding type in our validation, we should see a larger dropoff in accuracy.

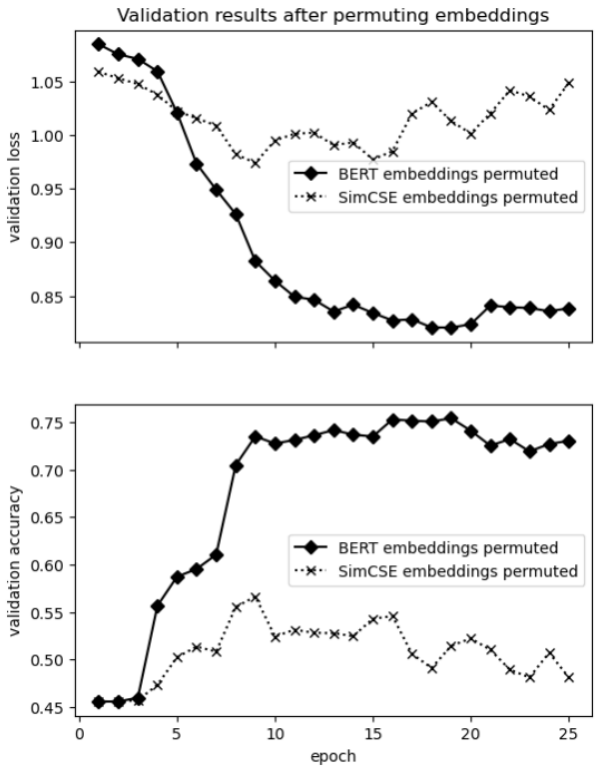

Figure 3: Validation loss and accuracy curves obtained during training after permuting BERT embeddings vs. after permuting SimCSE embeddings in the model input

6 Conclusion

After replicating the results of the original SimCSE paper, we sought to further demonstrate the utility of this contrastive learning framework in two separate experiments. In the first experiment, we trained two models on the same sentiment analysis task. One model used embeddings provided by our pretrained SimCSE encoder while the other used embeddings provided by the pretrained uncased BERT base encoder. When tasked with evaluating sentiment in a sample of tweets, we found that the SimCSE classifier was able to attain a higher mean test accuracy (0.88) when compared to the BERT classifier (0.74). Furthermore, we found that the SimCSE classifier was able to converge to near-maximal performance quicker and with less training data than the BERT classifier.

In our second experiment, we trained a single model using concatenated SimCSE and BERT embeddings. We then evaluated our model twice - first using data that included permuted (or randomized) SimCSE embeddings and again using data that included permuted BERT Embeddings. We found that after permuting the BERT embeddings, the model attained a mean test accuracy of 0.73, whereas when permuting the SimCSE embeddings, the model attained a mean test accuracy of 0.48. What this tells us is that the model likely "learned" more from the SimCSE embeddings than the BERT embeddings. This aligns with findings in our first experiment and explains why our model performance decreased the most when we stripped the interpretability and meaning from the SimCSE embeddings.

In sum, our results demonstrate how leveraging a contrastive learning framework such as SimCSE can improve sentence embeddings and ultimately improve performance in downstream NLP tasks such as sentiment analysis.

7 Group Member Contributions

- Jin Ishizuka: Wrote script to pull tweet data via the Twitter API and generate target sentiment labelings using the TextBlob library
- Andre Chen: Created tweet preprocessing pipeline and built, tuned, and evaluated sentiment classifier neural net model for BERT and SimCSE embeddings
- Aditya Srikanth: Generated dataset using API script and tweet pipeline. Conducted BERT and SimCSE embeddings permutation, trained network on permuted inputs and collected results.

References

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

Nikita Silaparasetty. 2022. Twitter sentiment analysis for data science using python in 2022.

(Gao et al., 2021) (Silaparasetty, 2022)