Enhanced Sentence we Embeddings with SimCSE

Stanford CS224N Default Project

Christo Hristov
Department of Computer Science
Stanford University
`christoh@stanford.edu`

Brendan McLaughlin
Department of Computer Science
Stanford University
`mclaughlin@stanford.edu`

Will Healy
Department of Computer Science
Stanford University
`whealy@stanford.edu`

Abstract

NLP model performance can be severely reduced by incomplete or unrepresentative embeddings. Looking to develop more comprehensive sentence embeddings, this paper explores the results of applying unsupervised SimCSE as described by Gao et al. (2022) to the BERT model proposed by Devlin et al. (2019). When evaluated on sentiment analysis and semantic textual similarity, we found that the SimCSE variant of BERT yields %0.036 and %0.001 lower results respectively than the BERT model. However, in paraphrase detection, the unsupervised SimCSE model produces a %0.003 improvement in accuracy, demonstrating SimCSE's potential to improve sentence embeddings for certain tasks in an accessible and unsupervised manner. The SimCSE variant yields 0.460 accuracy for sentiment analysis, 0.780 accuracy for paraphrase detection, and 0.337 Pearson's correlation for semantic textual similarity. This paper then proposes improvements to the original unsupervised SimCSE method described, investigating the usage of a Gaussian dropout on top of the normal dropout implemented in the BERT model. The usage of Gaussian dropout yields a 0.431 accuracy in sentiment analysis, 0.776 accuracy in paraphrase detection, and 0.329 Pearson's correlation in semantic textual similarity, demonstrating %0.029, %0.004, and %0.008 decrease respectively over the normal dropout unsupervised SimCSE.

1 Key Information to include

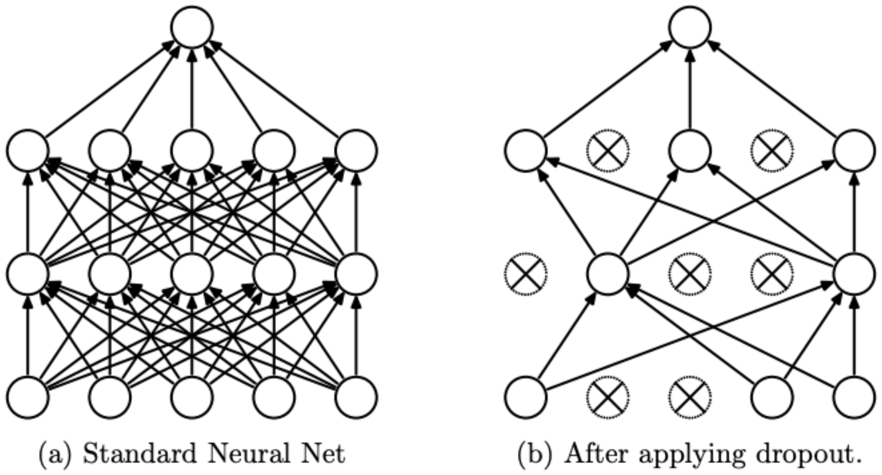
- Mentor: Hamza El Boudali
- Contributions:
  Christo: Implemented Gaussian Dropout adjustment to unsupervised SimCSE and helped summarize design decisions and implementations in the final paper.
  Will: Implemented SimCSE loss functions and integrated with multitask training to allow for simultaneous training of unsupervised SimCSE and multitask. Also assisted in the write-up.
  Brendan: Implemented missing code on base-BERT model and multiple parts of our classifier including the multitask training loop and the integration of contrastive learning into our training loop. Also assisted in the write-up

2 Introduction

A model's ability to capture the semantic meaning of a text depends on the accuracy of the embeddings it can produce for an input. More comprehensive embeddings can lead to better performances in a variety of NLP tasks. Thus, we investigate how to further improve general sentence embeddings derived from a pre-trained transformer encoder. For this paper, we decided to use BERT as the underlying encoder model, though our findings could be applied on top of any transformer-based encoder model Devlin et al. (2019). Sun et al. (2020) proposes further pre-training on data specific to the tasks of interest. While such pre-training leads to increased performance in the targeted tasks, we worry that training on such specific data will not lead to universal improvements in sentence embeddings for general tasks. Further suggestions include implementing a multiple negatives ranking loss learning objective to further train the model Henderson et al. (2017). Using a labeled data set of similar sentences and differing sentences, this loss function aims to minimize the embedding distance between similar sentences and maximize the embedding distance between differing sentences Henderson et al. (2017). While the resulting embeddings are generalizable, this finetuning objective relies on labeled data, of which we have less. Thus, we choose to focus on unsupervised SimCSE, a contrastive learning objective that does not require labeled data Gao et al. (2022).

2.1 Unsupervised SimCSE

Unsupervised SimCSE, as proposed by Gao et al. (2022), is a finetuning technique that utilizes a contrastive learning objective function. Contrastive learning functions aim to minimize the distance between similar data points (positive pairs), while maximizing the distance between differing data points (negative pairs). This idea can be applied to NLP, where positive pairs consist of embeddings of semantically similar sentences, while negative pairs are embeddings of semantically different sentences. To create positive pairs without labeled data indicating two sentences are similar, we simply run the same sentence through the encoder twice Gao et al. (2022). Due to the random dropout included in the design of BERT, the two encodings of the sentence have a different dropout applied and thus vary slightly. The dropout used was first introduced in Srivastava et al. (2014) and can be visualized below.



Negative pairs are formed with the target sentence embedding and every other sentence embedding in the batch Gao et al. (2022). By teaching to pull similar embeddings closer together and push different embeddings further apart, we hypothesize that SimCSE creates better sentence embeddings that are generalizable to all tasks. We apply Unsupervised SimCSE to a pre-trained BERT model by constructing a training function that simultaneously does contrastive loss using the loss function proposed by SimCSE and finetunes BERT on the tasks of sentiment classification, paraphrase detection, and semantic textual similarity. This approach yields accuracies for each task of 0.460, 0.780, .337 respectively. This is a %0.036 and %0.001 decrease in sentiment classification and semantic textual similarity accuracy, but a %0.003 increase in paraphrase detection accuracy relative to the accuracies obtained by the baseline pre-trained BERT model without SimCSE after the same simultaneous multi-task finetuning. These results show the model's slightly improved ability to interpret semantic similarities in text through additional unsupervised SimCSE finetuning.

2.2 Gaussian Dropout

We then expand upon the unsupervised SimCSE method proposed by Gao et al. (2022) by applying Gaussian dropout to the already embedded sentences that have undergone standard dropout within the BERT model, aiming to introduce further noise when deriving positive pair embeddings. Gaussian dropout differs from standard dropout in that, instead of setting certain activations to zero with a given probability, it assigns these activations to a Gaussian random variable with a mean of 0 and a chosen standard deviation. We conjecture that relying solely on the standard dropout inherent in the BERT model for creating positive pairs results in embeddings that are too similar to each other. Learning to push these similar embeddings closer together provides minimal new semantic understanding to the model. This is why we observe minimal improvements with the standard unsupervised SimCSE approach compared to using the baseline BERT model alone. By applying additional Gaussian dropout to the derived sentence embeddings, after they have already experienced the standard dropout, we aim to introduce more variability into the positive pair embeddings. Introducing this variance in representations through Gaussian dropout is expected to allow the model to explore a wider range of features and dimensions, thereby improving its ability to capture nuanced aspects of semantics and context. We opt for an additional Gaussian dropout layer over a standard one because a standard dropout layer could be too aggressive, creating overly dissimilar positive pairs. A Gaussian layer adds more nuanced, local variability than a binary standard dropout layer, potentially allowing the model to achieve a more comprehensive understanding.

When simultaneously finetuned on the tasks of sentiment classification, paraphrase detection, and semantic textual similarity, the pre-trained BERT model with Gaussian unsupervised SimCSE yields accuracies of 0.431, 0.776, and 0.329 respectively. This is a %.029, %.004, and %.008 decrease relative to the accuracies obtained by the baseline pre-trained BERT model with standard SimCSE after the same task finetuning.

3 Related Work

For the baseline BERT model that we finetune with both the unsupervised SimCSE and the other three tasks, we elect to use the 12 Encoder Transformer Layer BERT model Devlin et al. (2019). This is not only the model defined for the default final project, but also the model upon which the original unsupervised SimCSE was built upon Gao et al. (2022). When experimenting with unsupervised SimCSE, researchers gained insights into various model optimizations that we directly adopt to improve our own model's performance Gao et al. (2022). Such optimizations include dropout probability, data augmentation techniques to create positive pairs, learning rate, and batch size. However, in the original SimCSE paper, researchers only evaluated the finetuned unsupervised SimCSE BERT model with the semantic textual similarity task on the STS12, STS13, STS14, STS15, STS-B, and SICK-R datasets Gao et al. (2022). STS tasks lend themselves directly to the unsupervised SimCSE training framework, which is optimized to identify similar sentences and distinguish between differing ones. Thus, to test the true generalizability of the SimCSE sentence embeddings, we test these embeddings on three tasks: sentiment classification, paraphrase detection, and semantic textual similarity. While the STS task and paraphrase detection are still measuring text similarity, the sentiment classification task is fully different. By comparing results from the unsupervised SimCSE variations to the baseline BERT model in all three tasks, we develop further intuitions about resulting SimCSE embeddings and their ability to be generalized to different tasks that was not shown in the original exploration.

4 Approach

The baseline BERT model we will build upon is pre-trained using masked language modeling and next-sentence prediction on Wikipedia articles. This pre-trained model is then simultaneously finetuned on the tasks of sentiment analysis, paraphrase detection, and semantic textual similarity. For each task, a different additional linear layer is added to contextualize the results into the necessary number of logits. A dropout layer is also added after the BERT model but before the task-specific linear layer. This finetuned model is evaluated on the three tasks, to judge the embeddings developed without SimCSE.

4.1 Unsupervised SimCSE

We build upon the pretrained minBERT model Devlin et al. (2019). To clearly identify the impact SimCSE has on model embeddings, we make no model changes to min BERT, instead deciding to train the existing model with the SimCSE loss function. This means we exclude the MLP layer added on top of the [CLS] token, that is used in the original SimCSE exploration (Gao et al., 2022). We use the following loss function, also used in (Gao et al., 2022).

$$\ell_i = -\log \frac{e^{\mathrm{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{\mathrm{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'})/\tau}},$$

In this loss function, N refers to the batch size. $\mathrm{sim}(h_1, h_2)$ calculates cosine similarity $\frac{\mathbf{h_1} \cdot \mathbf{h_2}}{\|\mathbf{h_1}\| \|\mathbf{h_2}\|}$. For a batch of sentences, we calculate the loss for one sentence, l_i in the following manner. We derive $h_i^{z_i}$ and $h_i^{z_i'}$, by taking the same sentence x_i and running it through the BERT model twice. Due to the random dropout, z, applied by the model, we will get back two different hidden embeddings of the [CLS] token, $h_i^{z_i}$ and $h_i^{z_i'}$. We choose to use the [CLS] token embeddings rather than the average sentence embeddings in the loss function, because the original paper shows that using just the [CLS] token yields better results Gao et al. (2022). We opt to use the same dropout probability of $p = 0.1$ as shown to be optimal in the original paper (Gao et al., 2022). The similarity in the positive pair makes up the numerator that we want to maximize. We then compare negative pairs in the denominator. A negative pair is formed with one of the [CLS] embeddings of our target sentence that we already derived, $h_i^{z_i}$ and the [CLS] embedding of any other sentence in the batch, $h_j^{z_j'}$. Other sentences in the batch are assumed to differ relatively enough from the target sentence to be treated as negative pairs. As we derive this new embedding by running BERT, a new random dropout mask is used, z_j'. We compare our target sentence to every other sentence in the batch, thus leading to the summation over the batch in the denominator. The total loss of the batch is calculated as the average of these individual losses. The baseline BERT model simultaneously finetuned on all three tasks using the same additional three linear layers present in the baseline training and the added unsupervised SimCSE objective without any additional linear regressors.

4.2 Guassian Dropout Addition

The same objective function is used again. Initial positive pair sentence embeddings are derived by again running the same sentence through BERT twice. However, this time, these outputted [CLS] token embeddings are independently inputted through a Gaussian dropout layer to create the positive examples $h_i^{z_i}$ and $h_i^{z_i'}$. One of these positive examples is maintained to then construct the negative pairs. The additional Gaussian dropout is applied on top of the [CLS] token embeddings for the other sentences in the batch to create the negative examples, $h_j^{z_j'}$. After the unsupervised SimCSE finetuning is complete, the additional Gaussian dropout layer is removed for the finetuning of the model on the other three tasks. The linear layers specific to these three tasks are also reintroduced for their finetuing. This Gaussian dropout will use a dropout probability of $p = 0.1$, a mean of 0, and a standard deviation of 0.1.

5 Experiments

5.1 Data

For the finetuning of the pre-trained BERT model, we are using the datasets provided in the default project code. For sentiment analysis, we finetune our model using the Stanford Sentiment Treebank (SST). This dataset consists of thousands of unique prases labeled as negative, somewhat negative, neutral, somewhat positive, and positive (cite SST). To translate the BERT [CLS] embedding into one of these five rankings, an additional linear layer is included to project the [CLS] embedding into a vector of 5 logits. For paraphrase detection, we finetune our model using the Quora Question Pair Dataset (cite Quora). This dataset consists of sets of two questions and a label yes/no indicating whether the two questions are paraphrasing each other. To translate the BERT [CLS] embeddings into yes or no, the embeddings of the two sentences are concatenated together and then projected to a single logit. For semantic textual similarity, we finetune our model on the SemEval STS Benchmark dataset (cite STS). Within this dataset, two sentences are given a label between 0 and 5 indicating their degree of similarity. To translate the BERT [CLS] token embedding into a single similarity value, the embedding of each sentence in a pair is concatenated together and then projected through a linear layer to a single logit. We perform unsupervised SimCSE with both standard and Gaussian dropout using the same dataset from the original paper Gao et al. (2022). The paper randomly sampled 10^6 sentences from the WikiText-103 dataset [1]. This dataset consists of preprocessed sentences from a broad range of Wikipedia articles. As the SimCSE finetuning is unsupervised, no additional labeling is needed, allowing us to parse the Wikipedia data by determined batch size.

5.2 Evaluation method

We evaluate the baseline pre-trained BERT model, the unsupervised SimCSE standard dropout model, and the unsupervised SimCSE Gaussian dropout model on the same three tasks upon which each model is finetuned. Sentiment classification is evaluated by standard accuracy (proportion of sentiments predicted correctly over total predictions. Paraphrase detection is binary, so is also evaluated based on direct accuracy. As semantic similarity is evaluated on a scale of 1 to 5, the accuracy of the model is computed using the Pearson correlation of the true similarity value and the predicted value.

5.3 Experimental details

The finetuning of sentiment classification, paraphrase detection, and semantic textual similarity over each dataset occurs with a dropout probability across all layers of 0.1, a batch size of 8, 3 epochs, and a learning rate of 1e-5. As unsupervised SimCSE is simultaneously finetuned along with these three tasks, it is also trained for three epochs. The Gaussian unsupervised SimCSE is also finetuned along with the other three tasks so similarly is trained for three epochs. We elect to use the same specifications as the multitask classification model to directly compare the effect of the additional Gaussian layer. Thus we use a batch size of 8, a learning rate of 1e-5, and a dropout probability of 0.1.

5.4 Results

We compare the results of three models: a baseline multitask model, one enhanced with unsupervised SimCSE, and another further tweaked with Gaussian dropout. Our aim was to see how they performed across different tasks: Semantic Textual Similarity (STS), sentiment analysis, and paraphrasing. As shown by the bar graph, the baseline multitask model significantly outperformed the other two models on sentiment analysis. The baseline multitask model also was the highest performer on the STS dataset, but only slightly beating out unsupervised SimCSE with standard dropout. The unsupervised SimCSE had the highest accuracy on paraphrasing. The unsupervised SimCSE model with Gaussian dropout performed the worst on all three tasks.

6 Analysis

The baseline multitask model turned out to be pretty adept at picking up on the subtleties of sentences and performed well on all three tasks. Introducing SimCSE, despite being an unsupervised method, boosted the model's paraphrasing skills by helping it get better at spotting sentences that are different but mean the same thing. However, adding Gaussian dropout introduced too much noise to the positive pairs.

[1] https://paperswithcode.com/dataset/wikitext-103

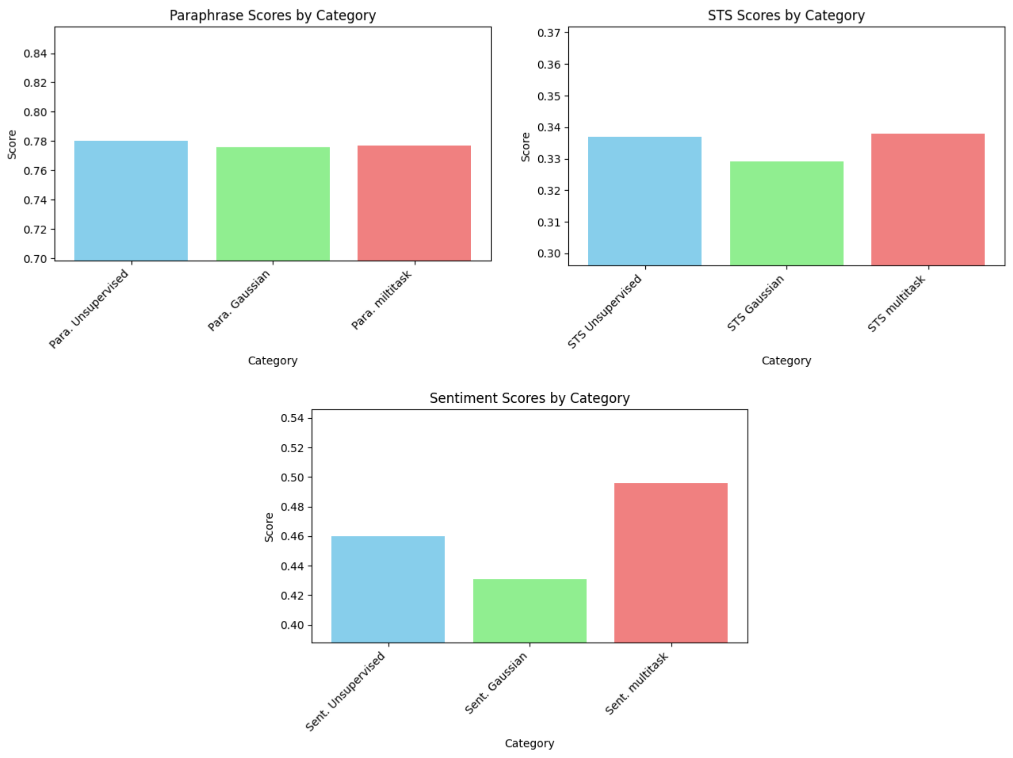


Figure 1: Comparison between regular SimCSE and SimCSE with Gaussian dropout

In the original SimCSE paper, their model was evaluated on standardized STS datasets and showed SOTA performance. Thus, we had originally hypothesized that the unsupervised SimCSE would do best on the STS task (as compared to the base multitask model). The improvement instead being on the paraphrase detection could potentially reflect our model's unique configuration or the specific nature of the datasets we used. The original SimCSE paper highlighted its strengths in capturing nuanced semantic similarities, which should theoretically benefit tasks like STS. However, our findings suggest that the context in which SimCSE was deployed alongside our multitask framework may have shifted its strengths more toward paraphrase detection. This could be due to the multitask model's architecture inherently supporting a broader understanding of language, which, when combined with SimCSE's unsupervised learning approach, enhances its ability to identify paraphrases, a task that demands a deep understanding of semantic equivalence rather than just similarity.

The introduction of Gaussian dropout caused the model to significantly worsen. Designed to inject a more complex form of noise into the training process, Gaussian dropout was supposed to encourage the model to become more robust by having it recognize the semantic similarity between more dissimilar positive pairs. However, this additional complexity backfired. Instead of making the model more versatile, it seemed to overload it with too much noise, making our positive pairs too distant to be useful. The failure of this additional noise is highlighted by the model's low scores on all three tasks.

This tells us that there's a fine line between adding complexity to improve a model and overdoing it to the point where performance starts to dip. The Gaussian dropout, in our case, crossed that line, suggesting that not all innovations lead to better results, especially in the delicate balance of language processing.

Going forward, it's clear that researchers must tread carefully when tweaking model architectures, especially with techniques like dropout. Researchers must find a sweet spot when using unsupervised contrastive learning by incorporating enough noise to generate sufficiently distant positive pairs but not so much that it begins to recognize dissimilar sentences as being similar. For anyone building NLP models, our study serves as a reminder to keep it simple, effective, and, most importantly, focused on the task at hand.

7 Conclusion

While our exploration of the integration of SimCSE-style contrastive learning with simultaneous multitask fine-tuning to improve sentence embeddings of BERT did not yield definitively superior results to the same approach without contrastive learning, it did lean toward the outcome that we hypothesized: it demonstrated slightly elevated performance on tasks that involved comparing two sentences. Additionally, not only did our approach contain elements of originality, by combining two of the default project extensions in an attempt to produce better generalized sentence embeddings than either of the extensions alone, but we also conducted our investigation in a thorough and controlled manner. We trained a baseline implementation and two variants of contrastive learning that were implemented directly atop the baseline featuring well-contained and intentional changes.

Because the SimCSE contrastive loss function had a runtime of $O(k^2)$ where k is the length of a batch our contrastive approaches proved to be computationally expensive, significantly slowing down our training process. For this reason, time was a limiting factor in the quality of our results since we had to limit our training iterations and number of epochs per training session. As a result, we had fewer opportunities to analyze how hyperparameters such as Gaussian standard deviation, learning rate, and batch size were affecting the accuracies of our data. We were not able to train our approach variations with a larger number of epochs. Each variation, baseline pre-trained BERT model, baseline pre-trained BERT model with unsupervised SimCSE, and the BERT model finetuned with Gaussian dropout unsupervised SimCSE, were all trained with only 3 epochs. Had we had more time, we would also change our implementation such that the effects of contrastive learning are amplified compared to the multitask fine-tuning. This would bring more certainty to the table regarding the impact of our approach variations since they were rather subtle in the present study.

Moving forward, the next steps would include experimenting with the aforementioned experiments and implementing adjustments to inform a new exploration direction. Possible directions for further exploration include reducing overfitting to the training data by tweaking and/or adding more regularization techniques the model, focusing solely on comparative tasks like STS and Paraphrase detection, and optimizing the dropout used in contrastive learning with multitask funetuning. Additionally, we would like to explore supervised SimCSE and the benefits it has on universal sentence embeddings, not just embeddings for semantic textual similarity tasks. Within supervised SimCSE, there are other hyperparameters fixed within the original paper Gao et al. (2022) that we would experiment with changing.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

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.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?