Robust Embeddings using Contrastive and Multiple Negatives on BERT

Stanford CS224N Default Project

Shoaib Mohammed
Department of Computer Science
Stanford University
shoaibmh@stanford.edu

Ishan Sabane
Department of Electrical Engineering
Stanford University
ishancs@stanford.edu

1 Key Information to include

- External collaborators (if you have any): N/A
- Mentor (custom project only): N/A
- Sharing project: N/A

2 Research paper summary (max 2 pages)

2.1 CSE: Contrastive Learning for NLP Fine-Tuning [1]

Background: Generating universal sentence embedding is a challenging problem in NLP. BERT which is an Encoder Based Architecture works well for generating sentence embeddings for multiple downstream tasks. One method to improve the performance on STS and paraphrase detection is to learn sentence embeddings such that the model learns by pulling semantically close neighbors together and pushing apart non-neighbor sentence embeddings [1].

Unsupervised Contrastive Learning: Consider a collection of sentences $\{x_i\}_{i=1}^m$ The unsupervised framework uses $x_i^+ = x_i$ to create a sentence pair (x_i, x_i^+). Now, a mask is used to drop random tokens from both the sentences. $h_i^z = f_\theta(x_i, z)$ where h_i^z is the masked sentence embedding and z is a random mask generated using dropout noise method with $p = 0.1$. The loss for a batch of N sentences is given as follows:

$$\mathcal{L} = -\log \frac{e^{\text{sim}(h_i^{z_i}, h_i^{z_i^*})/\tau}}{\sum_{j=1}^N e^{\text{sim}(h_i^{z_i}, h_j^{z_j^*})/\tau}}$$

All the model weights are fine-tuned using this loss. Methods such as changing the learning rate based on layer depth can be further be implemented on top of this method.

Supervised Contrastive Learning: Consider a set of paired examples $D = \{(x_i, x_i^+)\}_{i=1}^m$, where x_i and x_i^+ which are semantically related sentences. Let h_i and h_i^+ be the embedding representations of the two sentences. The sentence pairs are extend to triplets by adding negative examples. Certain datasets have negative examples which can be used directly. Hence, we convert the pair (x_i, x_i^+) to (x_i, x_i^+, x_i^-) Then the contrastive loss function for N sentence pairs is given as follows:

$$\mathcal{L} = -\log \frac{e^{\text{sim}(h_i, h_{i+})/\tau}}{\sum_{j=1}^N (e^{\text{sim}(h_i, h_j^+)/\tau} + e^{\text{sim}(h_i, h_j^-)/\tau})}$$

The above method of adding hard negative examples improves the performance for semantic textual similarity to 86.2 and is the final supervised model as used for comparitive study.

Contributions: The paper provides two learning frameworks depending on the availability of labeled dataset: Supervised and Unsupervised learning algorithms. The paper provides performance improvements for BERT base and RoBERTa over the STS dataset. The SimCSE framework outperforms previous results on semantic textual similarity tasks with an average of 76.3% for the unsupervised models and 81.6% for the supervised model using BERT base model. Other performance improvements are reported over the MLNI and SNLI datasets using supervised learning algorithm.

Limitations and Discussion The limitations in contrastive learning are due to the difficulty in generating similar sentence pairs using data augmentation methods. The unsupervised learning while improves the baseline performances, does not outperform the supervised learning in downstream tasks.

Why this Paper? The reason that we initially read the paper was because it is mentioned in the project references. However, the reason we chose this paper is because of the idea of contrastive learning which has been has proved to be efficient in computer vision. Contrastive learning aligns with alignment and uniformity which enables the learning framework to provide embeddings uniformly spread across the vector spaces while preserving semantic relatedness.

What is a wider research context of this paper? To build generalizable architectures, the extension of unsupervised contrastive objective may have a broader application in NLP. as it provides a new perspective on data augmentation. This method can be used in pretraining language models to create generalizable embeddings for sentences.

2.2 Efficient Natural Language Response Suggestion for Smart Reply [2]

Background. The paper is based on the problem of generating replies for emails. The replies are chosen from a fixed set of responses which are ranked on the basis of how good a response they are for the email.

Previously, statistical models were used to accomplish such a task. However, neural network models have replaced such systems and perform much better. More commonly, sequence-to-sequence models have been widespread in this area. The problem with `Seq2Seq` models is that they are generalized because of which replying to specific prompts can be difficult. It is also harder to rank common responses requiring extra normalization. Furthermore, they are slow and complicated to train.

In terms of motivation, the authors cite Kurzweil's work describing creating a simulation of the human neocortex (part of the brain that does most of the thinking) by having similar structured components encoding abstract ideas as sequeneces. This provides a structural hierarchy allowing to build upon the learnings of lower-level modules. Longer relationships are modeled in such a hierarchy.

Summary of contributions. The idea of using a hierarchy of deep networks built on top of n-gram representations is something that is new. This gets closer to modeling the human brain because there is a structure for each part of the deep network. The human brain has an intricate structure wherein each part of the brain focuses on a specific task. Also, the notion of having one correct response vs. $K-1$ negative responses with the aim of being able to identify why the correct response minimizes the distance is a great idea. The report precision at 1 *P@1* is 52% for a batch size of 50. Finally, the authors were able to show that this approach can outperform sequence-to-sequence models as you are better able to capture semantic information.

Limitations and discussion. I would say the approach of multiple negatives made this paper stronger. To get a specific response y from an input email x, we have to model $P(y|x)$. A set of K possible responses are used to approximate $P(y|x)$. It is important to note that the set K contains a single correct response and $K-1$ random negative responses. To elaborate upon this further, there are K input emails $\mathbf{x} = (x_1, x_2, \cdots, x_K)$ and we have their corresponding responses $\mathbf{x} = (y_1, y_2, \cdots, y_K)$. Any response y_j is a negative response for x_i if $i \neq j$. The approximate probability can be given by:

$$P_{\text{approx}}(y|x) = \frac{e^{S(x,y)}}{\sum_{k=1}^{K} e^{S(x,y_k)}}$$

In terms of flaws in the paper, it is unclear how long or short sentences are in the fixed set of responses. Is the model biased towards shorter or longer sentences? There could be problems associated with length which are not mentioned in the paper.

Why this paper? The reason that we initially read the paper was because it is mentioned in the spec. However, the reason that we chose the paper is because of the idea of using hierarchical structure.

It is interesting to see how the replies are generated. First, potential responses are pre-computed through the minimal hierarchy. At runtime the input is propagated through the hierarchical network. Then, using nearest neighbor search of the hierarchical embeddings, the best suggestions are chosen. We found the idea of trying to model structure as is in the brain interesting as this could help capture deep information of sentences.

Wider research context. We think any problem that requires comparing how good responses are can make use of *Smart Reply*. This could be with tasks involving replying to emails and messaging platforms. But it could also be applied to tasks such as STS wherein we have to evaluate the level of similarity of two sentences. A real-life application could be of translation services wherein we can perform STS to see how similar a machine translation is to a reference translation.

One application the authors mention is the problem of finding datapoints with the largest dot-product values. This problem is also called *Maximum Inner Product Search (MIPS)*. Using *hierarchical quantization for efficient search* is able to solve MIPS much faster than the traditional methods.

3 Project description

3.1 Goal

We will be implementing a Bidirectional Encoder Representations from Transformers (or BERT) model to perform three different tasks. BERT is a family of masked-language models. The first part will involve using pre-trained weights loaded into the BERT model and then performing sentence classification on the `sst` dataset and `cfimdb` dataset. Next, we will fine-tune the BERT model to perform well on multiple sentence-level tasks. Our goal is to have a BERT model that is successful in achieving the baselines described below on the individual tasks.

3.2 Task

Our model will work on three different tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS). In sentiment analysis, the goal is to evaluate the sentiment of a phrase on a scale from 0 to 4. In paraphrase detection, you have two sentences and you want to evaluate whether the second sentence is a paraphrase of the first sentence, the output is a simple yes/no (binary classification). Finally, in STS, the goal is to measure the degree of semantic evaluation between two sentences on a scale from 0 to 5.

3.3 Data

We will be using two datasets: Stanford Sentiment Treebank (SST) dataset and the CFIMDB dataset. The SST dataset contains 11,855 single sentences which were extracted from movie reviews. The dataset contains 215,514 unique phrases with a label representing values from 0 to 4. The CFIMDB dataset contains 2,434 movie reviews with a binary label for each review. We have the following splits for each dataset:

split	SST	CFIMDB	Quora	SemEval STS
train	8,544	1,701	141,506	6,041
dev	1,101	245	20,215	864
test	2,210	488	40,431	1,726

3.4 Methods

We will use a `WordPiece` tokenizer to convert input sentences into tokens which then get mapped to ids. Then, for each token we will have an embedding layer. Each embedding layer has a

dimensionality of 768. We will then implement a multi-head self attention layer which takes a query and set of key-value pairs producing an output. Finally, we will then implement a transformer layer. Additionally, we will also be implementing the `step()` function of the Adam Optimizer. This completes the implementation of the BERT model. We will then evaluate the model on the three tasks and our hope is to achieve the baseline accuracy.

Next, our goal is to build upon the model to further improve its performance by integrating the extensions from the papers discussed earlier. We plan to have additional pretraining by integrating ideas such as *fine-tuning BERT for text classification* and *cosine-similarity fine-tuning*.

3.5 Baselines

The following are task specific baselines which would be used to compare with our BERT base model implementation as well as with the fine-tuned improvements. We plan to use the BERT base model as the baseline for each of the three tasks.[3] For finetuning, we use the CSE: Contrastive Learning performance metrics for each task using the BERT base model [1].

Sentiment Analysis The given SST and CFIMDB dataset has baselines given the the default project description. We plan to compare our baseline BERT sentiment classification accuracy with the following baselines.

Dataset	Pretrained	FineTuned
SST	0.39	0.515
CFIMDB	0.78	0.966

Table 1: Accuracy score on the Development Set for Sentiment Classification

Semantic Textual Similarity The SemEval dataset includes 5 levels of similarity between two sentences. Hence we will use the default metric which is the Pearson score to compare our model performance on the leader-board and the given SemEval dataset. We plan to compare out model performance on the STS Benchmark dataset as well.

Paraphrase Detection We use the accuracy for the model to measure the performance on the Quora Dataset. Since it's a binary classification task, we compare the accuracy of our implementation with the baseline BERT model published scores.

3.6 Evaluation

We plan to evaluate our BERT implementation for sentiment classification, semantic textual similarity and paraphrase identification. For each task we have the baselines of the existing methods as well as for the fine-tuned methods. For all the tasks, accuracy score will be used as the evaluation metric. We plan to compute other metrics such as F1-score, ROC-AUC score, inference latency as well as memory usage to understand the overall performance of the model.

- For sentiment classification we plan to use the default accuracy metric to compare our model performance with the given baselines. Ideally, we will plan to use mean accuracy over the classes, F1-score and ROC-AUC scores. For sentiment classification on the binary movie review dataset, we use accuracy as well.
- For Semantic Textual similarity we use the Pearson score as the default metric along with accuracy score and cosine similarity between two sentences.
- For Paraphrase detection we use accuracy as the default metric since it is a binary classification problem.

References

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

[2] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.