# Improving Semantic Meaning of BERT Sentence Embeddings

Stanford CS224N Default Project

**Timothy Yao**
Department of Computer Science
Stanford University
timyao7@stanford.edu

## Abstract

SBERT has been shown to improve the performance of BERT on downstream tasks, such as STS, by deriving semantically meaningful sentence embeddings from the BERT output. We examine whether SBERT fine-tuning is also effective in a multitask setting. Specifically, we investigate if fine-tuning the BERT model using the SBERT method can improve the performance on the three downstream tasks of sentiment analysis, paraphrase detection, and STS simultaneously. We experiment with different combinations of pooling strategies and fine-tuning methods. The results indicate that the SBERT model provides a solid foundation for improving the semantic meaning of the output sentence embeddings in the multitask domain, yielding a higher ability to be applied to other tasks and datasets compared to the standard BERT model.

## 1 Key Information

- Mentor: Kamyar Salahi
- External Collaborators (if you have any): N/A
- Sharing project: N/A

## 2 Introduction

This project aims to improve BERT [Devlin et al. (2018)] sentence embeddings through the implementation of the Sentence-BERT (SBERT) [Reimers and Gurevych (2019)] fine-tuning framework. In doing so, we investigate the effectiveness of SBERT to perform in a multitask setting, i.e., SBERT's ability to derive semantically meaningful embeddings that can be generalized for multiple downstream tasks. Specifically, we attempt to use the SBERT method for improving the performance of the BERT model on the three downstream tasks of sentiment analysis, paraphrase detection, and semantic textual similarity (STS). Then, we analyze the generalizability of this multitask BERT model by evaluating on a fourth unseen task - natural language inference (NLI).

BERT is a transformer network that, in 2018, set state-of-the-art results for NLP tasks related to sentence classification and sentence-pair classification/regression, such as sentiment analysis and paraphrase detection, respectively. For the latter, BERT works by passing in the sentence pair concatenated together to the network and obtaining a single output. While this method has been shown to be quite effective performance-wise, it is computationally impractical and a key limitation when considering other sentence-pair tasks. For instance, the task of finding the most similar pair of sentences in a corpus of $n$ sentences would take $n(n-1)/2$ inference computations. Therefore, a better procedure was required to improve the efficiency at which BERT performs sentence-pair tasks, while still leveraging and maintaining BERT's inference performance.

The conventional approach to this problem within the NLP field has been to map sentences to vector space embeddings such that semantically similar sentences are close together and dissimilar sentences are far apart. If these sentence embeddings exhibit this property, we call them *semantically meaningful*. Early approaches to derive sentence embeddings involved either averaging the BERT output (also known as the BERT embedding) or using the special classification token ([CLS]) attached to the front of every sequence. However, these approaches were shown to yield relatively poor embeddings in terms of semantic meaning [Reimers and Gurevych (2019)].

Sentence-BERT (SBERT) was a milestone method for deriving semantically meaningful sentence embeddings from BERT. SBERT's strategy is two-pronged: (1) apply a pooling operation to the BERT output to guarantee a fixed-sized sentence embedding and (2) use a siamese network that simultaneously takes two inputs to fine-tune the BERT model. In this siamese framework, a sentence embedding is generated for each sentence in a pair, which can then be fed into the downstream objective function. SBERT demonstrated state-of-the-art results not only in terms of training time and efficiency, but also in terms of performance on sentence-pair tasks. Specifically, Reimers and Gurevych (2019) demonstrate that SBERT fine-tuning for single downstream tasks consistently outperformed other methods that also attempt to construct BERT sentence embeddings.

Recently, there has been some research suggesting that multitask learning can reduce overfitting, improve performance on all downstream tasks that are trained on, and generalize models to unseen tasks [Zhang and Yang (2018)]. With its impressive capability of deriving semantically meaningful embeddings yielding high single-task performance, SBERT provides a promising starting point for developing a multitask model. We evaluate whether SBERT can significantly improve upon BERT in the multitask setting. In other words, we examine SBERT's ability to derive embeddings that are not only semantically meaningful but also generalizable in the sense that the same fine-tuned BERT model can be used for multiple downstream tasks, including tasks that the model was not directly trained for.

## 3 Related Work

BERT [Devlin et al. (2018)] demonstrates that a pre-trained transformer model can perform exceptionally well on various tasks, such as sentiment analysis and STS. One of the key limitations of BERT was that it did not directly produce semantically meaningful sentence embeddings, causing it to be inefficient when performing certain sentence-pair tasks that require searching a large space of sentences. SBERT [Reimers and Gurevych (2019)] improved upon BERT, developing a strategy by which the BERT output could be used to derive semantically meaningful sentence embeddings through a siamese network architecture. Moreover, SBERT has been shown to perform better on a wide range of sentence-pair tasks compared to other models because of this capability to derive embeddings are semantically meaningful. Some such models include those that use the BERT output directly [Qiao et al. (2019)] and those that derive sentence embeddings, such as InferSent [Conneau et al. (2017)] and Universal Sentence Encoder [Cer et al. (2018)] As such, SBERT is the basis by which we investigate an improvement to BERT for the purpose of improving multitask performance.

There are several proposed methods for multitask learning. One method is to group related tasks such that information can be shared within groups, thereby allowing a model to be trained simultaneously on multiple datasets and tasks [Kumar and Daume III (2012); Rai and Daumé III (2010)]. This method involves developing a single loss function, often involving a combination of the individual tasks' loss functions, that can train the model for all of the tasks. The main difficulty with this approach is how to share information between related tasks while separating information between unrelated tasks. Another method uses unrelated auxiliary tasks for learning multiple principal tasks [Paredes et al. (2012)]. By exploiting the prior knowledge that the auxiliary tasks are unrelated, training in this way can yield a more robust data representation that is more effective for the principal tasks to learn on. Both of these methods train on all the tasks and all the datasets in a simultaneous fashion.

The primary method for multitask learning that this project works with utilizes the underlying idea of transfer learning, where knowledge from an underlying model can be transferred to boost performance on a related task. In the multitask setting, we simply extend this notion to fine-tuning on multiple related tasks one after another so that each each task's performance is boosted by the training on the others. Unlike the previous two methods, this form of multitask learning is sequential; model updates

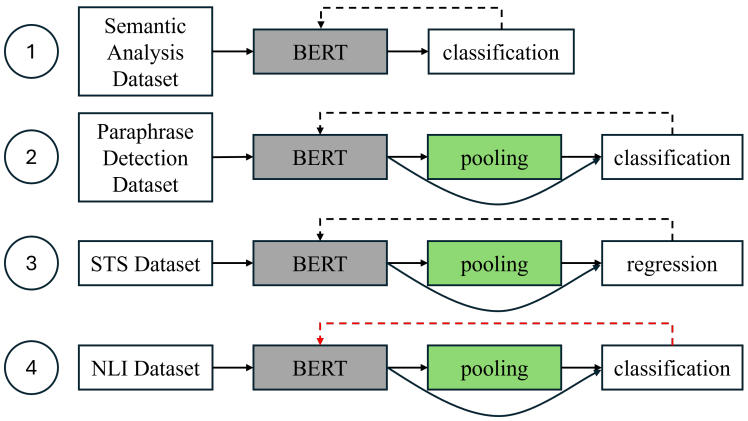


Figure 1: Sequential multitask training: we fine-tune on each task one after another. Note that the BERT model is the same across the entire sequence of steps. The pooling method across the different tasks can vary. Additionally, we also experiment without the pooling, i.e., using the original BERT formulation; for STS, using BERT implies evaluating on an outputted logit instead of a cosine similarity. We are interested in how the 3-task BERT model performs on the unseen NLI task, though we do perform an experiment training on the NLI dataset as well.

from training on one task are carried over prior to training the next, thereby learning an output that is a combination of learning on each task individually. For instance, the BERT model is pre-trained and then can be used by SBERT to fine-tune for a specific task. Likewise, the SBERT fine-tuned model for one task can in turn be used to fine-tune on another task, where some information from the model is passed along and some new information is embedded. Hence, by iteratively fine-tuning on each task, the base model can be trained to generalize for multiple tasks.

## 4 Approach

Our approach sequentially fine-tunes the BERT model on each of the three downstream tasks of sentiment analysis, paraphrase detection, and STS in that order. For sentiment analysis, a single-sentence classification task, we fine-tune using the BERT method. For paraphrase detection and STS, we experiment with different combinations of fine-tuning using the BERT and SBERT methods. Figure 1 illustrates this multitask training approach. The implementation of this training approach builds off the code given in the DFP handout, but is otherwise our own.

The SBERT approach relies on two key components. The first component is a pooling operation that is applied to the output of BERT in order to generate a fixed-sized sentence embedding. There are three pooling operations that SBERT suggests: using the `CLS`-token BERT output, taking the `MEAN` of the BERT output embeddings, and taking the `MAX`-over-time of the BERT output embeddings. The second component is using a siamese network that simultaneously takes a pair of input sentences to fine-tune the BERT model; this can be applied to the sentence-pair tasks of paraphrase detection and STS. In this siamese framework, a sentence embedding is generated for each sentence in a pair, which can then be fed into an objective function based on the downstream task. The implementation of both components is my own.

For both paraphrase detection and STS, we experiment with the three pooling strategies. Reimers and Gurevych (2019) found that the `MEAN` strategy performed best on STS and NLI in the single-task setting, though the difference was small. Conneau et al. (2017) found that the `MAX` strategy performed best on the NLI task. Neither report the best pooling strategy for the paraphrase detection task. We seek to evaluate which combination of potentially different pooling strategies performs best.

For paraphrase detection, we use the classification objective function. Let $n$ be the fixed size of the sentence embeddings after pooling the BERT output, and $k = 1$ be the number of labels. Given a pair of sentences $A$ and $B$, let $u, v \in \mathbb{R}^n$ be the fixed-size embeddings obtained from each sentence, respectively. We apply a softmax classifier on the product of a trainable weight matrix $W_t \in \mathbb{R}^{3n \times k}$ with the concatenation $(u, v, |u - v|)$ where $|u - v|$ denotes the absolute value of the element-wise

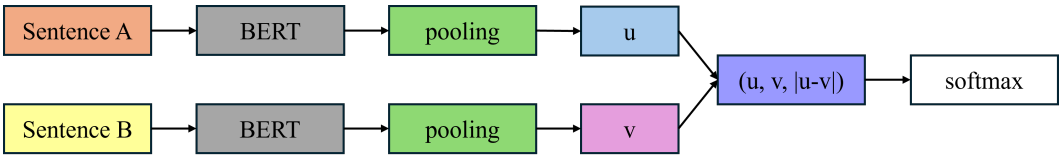


Figure 2: SBERT architecture for the classification objective function, i.e., for fine-tuning paraphrase detection or SNLI. Note that the BERT model and the pooling method are the same for both sentences. (Figure adapted from Reimers and Gurevych (2019).)

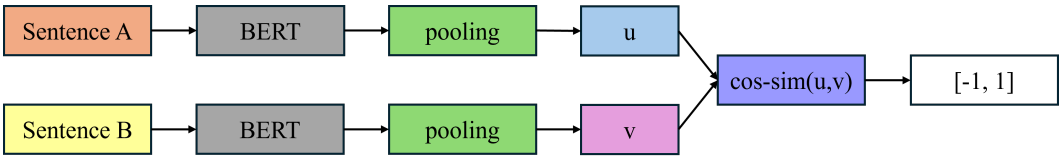


Figure 3: SBERT architecture for the regression objective function, i.e., for fine-tuning STS. Once again, note that the BERT model and the pooling method are the same for both sentences. (Figure adapted from Reimers and Gurevych (2019).)

difference between the two embeddings:

$$\text{softmax}(W_t(u, v, |u - v|)).$$

Reimers and Gurevych (2019) found that this concatenation method performs best for the NLI task under `MEAN` pooling, with the most significant component being $|u - v|$. This makes intuitive sense under the goal of producing semantically meaningful embeddings since the element-wise difference acts as a measure of how close the embeddings are in vector space. However, they do not include the best concatenation method for the paraphrase detection task. Reimers and Gurevych (2019) suggest using the concatenation $(u, v, |u - v|, u * v)$ for the NLI task, where $u * v$ is the element-wise product. (Note that under this concatenation $W_t \in \mathbb{R}^{4n \times k}$.) We investigate which concatenation strategy yields better results for paraphrase detection within the multitask setting. We optimize binary cross-entropy loss. Figure 2 depicts this pipeline.

For STS, we use the regression objective function. We optimize the mean squared error loss between the cosine similarity of the embeddings $u$ and $v$ and their true labeled similarity, appropriately normalized. Figure 3 depicts this pipeline.

For the NLI task, we are primarily interested in the performance of the BERT model fine-tuned on the previous three tasks. However, we do run an experiment where we also train on NLI to see if this can boost the performance on the main thee tasks. In this case, we use the classification objective function as formulated above, except now $k = 3$.

We evaluate our approach against the baseline BERT model developed in Part 1 of the DFP. Namely, we have four baselines: (1) BERT fine-tuning on sentiment analysis only, (2) BERT fine-tuning on paraphrase detection only, (3) BERT fine-tuning on STS only, (4) BERT fine-tuning on sentiment analysis, paraphrase detection, and STS.

## 5 Experiments

### 5.1 Data

For sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset [Socher et al. (2013)] consisting of 11,855 (8,544/1,101/2,210)[1] sentences from movie reviews each having one of five labels: negative, somewhat negative, neutral, somewhat positive, or positive.

For paraphrase detection, we use the Quora[2] dataset consisting of 404,298 (283,010/40,429/80,859) question pairs with a binary label indicating if they are paraphrases of one another.

[1]For each data set, we report (`train/dev/test`) splits.

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

For STS, we use the SemEval STS Benchmark dataset [Agirre et al. (2013)] consisting of 8,628 (6,040/863/1,725) sentence pairs with a similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). In order to perform the regression task, we scale these similarities to range from 0 to 1.

For NLI, we use the Stanford Natural Language Inference dataset [Bowman et al. (2015)] consisting of 569,033[3] (549,367/9,842/9,824) sentence pairs labeled entailment, contradiction, or semantically independent.

We recognize that the Quora dataset is orders of magnitude larger than the SST and STS datasets, which could potentially lead the multitask model to over-tune for paraphrase detection. As such, we run an experiment to detect if training only on a random subset (approximately 10%) of the Quora dataset that is comparable in size to the SST and STS datasets leads to better multitask performance.

Finally, we note that each dataset is split into `train`, `dev`, and `test` sets. The models are trained on the `train` set, validated on the `dev` set for each epoch, and a final evaluation is performed on the `test` set once training is complete.

### 5.2 Evaluation method

For sentiment analysis, paraphrase detection, and STS, we use the evaluation metrics given in the DFP. We use the classification accuracy for sentiment analysis and paraphrase detection. Similarly, we use classification accuracy for NLI since it is a classification problem that follows a similar formulation as paraphrase detection. We use Pearson correlation for STS.

Note that both BERT and SBERT output classification logits for sentiment analysis, paraphrase detection, and NLI. Hence, it is clear that using classification accuracy for either model is well-formulated. On the other hand, BERT and SBERT output different objects for STS: BERT outputs logits for a binary classification of whether a pair of sentences is unrelated or similar, whereas SBERT outputs a cosine similarity. That being said, both are measures of how similar two sentences are, i.e., they both are expected to be statistically correlated with the true similarity score. Therefore, Pearson correlation is still a valid and comparable evaluation metric regardless of whether we fine-tune using BERT or SBERT.

Finally, in order to choose the best model, we include the DFP leaderboard overall score, which is computed to be the average of SST accuracy, Quora accuracy, and (SemEval correlation + 1) / 2.

### 5.3 Experimental details

For each experiment, we trained the full model for 5 epochs using a batch size of 32 and AdamW optimizer with learning rate of 1e-5 on a T4 GPU using either Google GCP or Google Colab. The slowest experiment - fine-tuning on the entireties of all four datasets - trained for approximately 15 hours. The fastest experiment - baseline of fine-tuning on SemEval dataset only - trained for less than 5 minutes.

### 5.4 Results

In Table 1, we report the evaluation metrics for the best performing model (in terms of overall score) on the `dev` set from each experiment.

**Baselines.** First, we observe that the baselines trained for single tasks have quite effective scores for their respective tasks. This is expected since Devlin et al. (2018) demonstrated that the BERT CLS token can be decently effective at both sentence classification and sentence-pair tasks, even if they are not necessarily semantically meaningfully. Furthermore, the baseline of performing sequential multitask fine-tuning on all three datasets improves the overall score, though individual scores may suffer slightly. This makes sense as the BERT model is learning a generalized output that can perform relatively well on all three tasks simultaneously.

**Pooling Methods.** We experiment with different combinations of pooling methods between the paraphrase detection and STS tasks. There are a total of 16 possible combinations, including using standard BERT (i.e., no pooling). For the sake of time, we perform five experiments to evaluate the different pooling methods on each of the two datasets in an attempt to understand what methods

[3]This is after filtering out sentence pairs that had no consensus classification.

| Model | SST | Quora | SemEval | SNLI | Overall |
|---|---|---|---|---|---|
| SA | 0.515 | 0.570 | -0.101 | — | 0.512 |
| P | 0.201 | 0.901 | 0.577 | — | 0.630 |
| STS | 0.142 | 0.563 | 0.834 | — | 0.541 |
| SA-P-STS | 0.481 | 0.903 | 0.836 | 0.118 | 0.768 |
| SA-P-STS(C) | 0.516 | 0.891 | 0.592 | — | 0.734 |
| SA-P-STS(M) | 0.519 | 0.898 | 0.800 | 0.321 | 0.772 |
| SA-P(C)-STS(C) | 0.488 | 0.875 | 0.641 | — | 0.728 |
| SA-P(M)-STS(M) | 0.516 | 0.868 | 0.799 | 0.317 | 0.761 |
| SA-P(X)-STS(X) | 0.510 | 0.850 | 0.725 | — | 0.741 |
| SA-P[sub]-STS(M) | 0.509 | 0.849 | 0.809 | — | 0.754 |
| SA-P[sub](M)-STS(M) | 0.510 | 0.856 | 0.801 | — | 0.755 |
| SA-P(M)*-STS(M) | 0.507 | 0.894 | 0.813 | 0.330 | 0.769 |
| SA-P-STS(M)-NLI(M) | 0.492 | 0.834 | 0.754 | 0.855 | 0.730 |

Table 1: Results for each experiment. SA, P, STS, and NLI indicate model was fine-tuned on sentiment analysis, paraphrase detection, STS, and NLI, respectively. (C), (M), (X) indicate `CLS`, `MEAN`, and `MAX` pooling was used for SBERT fine-tuning; if none of these are present, then fine-tuning was done using BERT. [sub] indicates a random subset of the `train` set was used for training. SNLI evaluation was only performed on the top model and the NLI fine-tuned model. * indicates that $(u, v, |u - v|, u * v)$ concatenation was used.

work better for which datasets and in combination. Among these five experiments, we found the (paraphrase detection, STS) pairs that worked best in terms of overall performance were (BERT, `MEAN`) and (`MEAN`, `MEAN`).

**Quora Train Size.** We run experiments on the two best pooling models to see if randomly sampling a subset of the Quora `train` set at each epoch could lead to better generalizability on all three tasks and improve overall performance. Specifically, at each epoch, we only train on a random sample of 10% of the data so that the size is comparable to the SST and SemEval datasets. Interestingly, these two experiments produce results that are comparable to their full-Quora counterparts, though slightly worse. This perhaps is because training on a larger set will in general allow the model to learn more in this multitask setting even if most of the data comes from a single task.

**Product Concatenation.** Using this concatenation method does appear to improve both paraphrase detection and overall performance of the multitask model over the corresponding model that uses the standard SBERT `MEAN` pooling. Conneau et al. (2017) suggest that the element-wise product also encodes a notion of distance. Moreover, this extra product encodes more information about the relationship between the two sentence embeddings that can work well for tasks like NLI and paraphrase detection, though not on its own.

**SNLI.** We evaluate our top three models as well as the multitask BERT baseline on the SNLI dataset to examine the capability of the multitask model to generalize to unseen tasks. We also run a single experiment where we train the multitask model on the SNLI dataset. In this experiment we obtain a quite high SNLI score at the cost of the other three. Furthermore, the performance of the other four models on SNLI are quite poor. This could be due to the fact that NLI is quite a different problem compared to STS and paraphrase detection; NLI aims to learn entailment vs. contradiction, whereas STS and paraphrase detection are more related to the similarity of two sentences' meanings. However, the SBERT fine-tuned models do outperform the BERT baseline, suggesting that the SBERT embeddings are more generalizable to other tasks, even ones that are not directly related to the original problem objective.

**Final Model Prediction.** We use the `dev` set results to choose the top 3 models among all the experiments to submit to the `test` set leaderboard. We report the evaluation metrics on the `test` set for these 3 models in Table 2. The final chosen model's performance is italicized in the first row.

| Model | SST | Quora | SemEval | Overall |
|---|---|---|---|---|
| *SA-P-STS(M)* | *0.498* | *0.896* | *0.804* | *0.765* |
| SA-P(M)*-STS(M) | 0.484 | 0.894 | 0.836 | 0.765 |
| SA-P(M)-STS(M) | 0.510 | 0.865 | 0.800 | 0.758 |

Table 2: `test` set leaderboard submission results for the top 3 performing models on the `dev` set.

## 6 Analysis

Based on these sets of experiments where we explore different pooling methods as well as different tweaks to try to improve the generalizability of the model, a combination of using the original BERT fine-tuning and SBERT fine-tuning on different datasets appears to produce the best model when evaluating on the set of three downstream tasks. In terms of raw overall performance, the best model does not perform significantly better than the baseline SA-P-STS. However, when looking at the individual task scores, we observe that the baseline SA-P-STS model must sacrifice the performance on SST to fit the Quora and SemEval data better. This goes against the original aim of multitask learning to use information from each task to boost the performance on all tasks simultaneously. Indeed, the SA-P-STS model, since it uses the same BERT CLS token for all three tasks, is likely learning an embedding that fits the data it is trained the most on, i.e., Quora. Furthermore, it appears from the P model that training on the Quora dataset is linked to improvement on the SemEval dataset but not the SST dataset. This is most likely a result of the fact that paraphrase detection and STS are fundamentally similar tasks that have objectives related to developing semantic meaning in the sentence representations. Sentiment analysis is a much different problem, and so it is foreseeable that the BERT output trained primarily on the Quora and SemEval datasets do not generalize well to SST, even though it has also been trained on.

The best three models utilizing the SBERT framework, particularly the `MEAN` pooling, learn embeddings that are much better suited for the multitask problem as we see they do not sacrifice as much performance on any one individual task. Moreover, though it is not near to directly training on the NLI task performance-wise, the SA-P-STS(M) and SA-P(M)*-STS(M) models perform much better compared to SA-P-STS when evaluating on the unseen SNLI dataset. This suggest that these two models are learning embeddings that are semantically meaningful, though such a property is not fully generalizable to the NLI task which uses a contradiction category that is not captured by the STS or paraphrase detection problem formulations.

## 7 Conclusion

We present a sequential framework for fine-tuning a BERT model to perform well simultaneously on the three downstream tasks of sentiment analysis, paraphrase detection, and STS. We demonstrate that the SBERT method provides an effective multitask fine-tuning backbone for learning semantically meaningful sentence embeddings compared to the standard BERT. SBERT fine-tuning yields our best two models. Furthermore, these SBERT models do not sacrifice performance on any one individual task to gain a upper hand on another. In essence, SBERT is able to significantly improve upon the semantic meaning of the BERT sentence embeddings, yielding close-to-top performance on individual tasks simultaneously and top performance overall, as well as somewhat generalizing to unseen tasks and datasets.

## 8 Ethics Statement

One ethical consideration of this project is whether the BERT or SBERT representations of language outputted by this model effectively capture social standards. We recall the goal of producing semantically meaningful sentence embeddings is simply to pull similar sentences together and push dissimilar sentences apart. It then brings into question whether this goal is able to capture embeddings that give an improved analysis of sentence sentiment that reflects current moral or ethical values. In other words, do these semantically meaningful sentence embeddings allow the model to learn morally meaningful embeddings? There is some research that may suggest BERT and SBERT do have an improved "moral compass" [Schramowski et al. (2019)]. To improve the moral meaning of the

embeddings, one possible strategy might be to incorporate a morality tag into the output embeddings, similar to how the special CLS token is attached to the BERT output. This tag could be continuous, perhaps on some multi-dimensional hypersphere that encodes a moral compass, thereby allowing for a regression task that provides some insight as to the degree by which a sentence is moral.

Another related issue is the generalization of such moral or social standards across different languages. It is foreseeable that one could combine a NMT system and this multitask BERT model for a strategy of evaluating how well semantic meaning is preserved after translation. However, we must remember that different cultures, and so languages, have different notions of what is moral or ethical. Therefore, if the BERT or SBERT output is morally meaningful, we must be careful of how moral values are translated when utilizing these sentence embeddings. Building off the moral token idea, there could be a way to develop a learning based method for translating the moral token into different cultures based on a different moral compass. However, this would require a reasonable encoding of a moral compass, which could be doable but certainly difficult, especially given the diversity present even within a single language.

## References


Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

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

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

Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*.

Bernardino Romera Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. 2012. Exploiting unrelated tasks in multi-task learning. In *Artificial intelligence and statistics*, pages 951–959. PMLR.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*.

Piyush Rai and Hal Daumé III. 2010. Infinite predictor subspace models for multitask learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 613–620. JMLR Workshop and Conference Proceedings.

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

Patrick Schramowski, Cigdem Turan, Sophie Jentzsch, Constantin Rothkopf, and Kristian Kersting. 2019. Bert has a moral compass: Improvements of ethical and moral values of machines. *arXiv preprint arXiv:1912.05238*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Yu Zhang and Qiang Yang. 2018. An overview of multi-task learning. *National Science Review*, 5(1):30–43.