

# Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers and Iryna Gurevych

Ubiquitous Knowledge Processing Lab (UKP-TUDA)  
Department of Computer Science, Technische Universität Darmstadt

[www.ukp.tu-darmstadt.de](http://www.ukp.tu-darmstadt.de)

## Abstract

BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) has set a new state-of-the-art performance on sentence-pair regression tasks like semantic textual similarity (STS). However, it requires that both sentences are fed into the network, which causes a massive computational overhead: Finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (~65 hours) with BERT. The construction of BERT makes it unsuitable for semantic similarity search as well as for unsupervised tasks like clustering.

In this publication, we present Sentence-BERT (SBERT), a modification of the pretrained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT, while maintaining the accuracy from BERT.

We evaluate SBERT and SRoBERTa on common STS tasks and transfer learning tasks, where it outperforms other state-of-the-art sentence embeddings methods.<sup>1</sup>

## 1 Introduction

In this publication, we present Sentence-BERT (SBERT), a modification of the BERT network using siamese and triplet networks that is able to derive semantically meaningful sentence embeddings<sup>2</sup>. This enables BERT to be used for certain new tasks, which up-to-now were not applicable for BERT. These tasks include large-scale seman-

tic similarity comparison, clustering, and information retrieval via semantic search.

BERT set new state-of-the-art performance on various sentence classification and sentence-pair regression tasks. BERT uses a cross-encoder: Two sentences are passed to the transformer network and the target value is predicted. However, this setup is unsuitable for various pair regression tasks due to too many possible combinations. Finding in a collection of  $n = 10\,000$  sentences the pair with the highest similarity requires with BERT  $n \cdot (n-1)/2 = 49\,995\,000$  inference computations. On a modern V100 GPU, this requires about 65 hours. Similar, finding which of the over 40 million existent questions of Quora is the most similar for a new question could be modeled as a pair-wise comparison with BERT, however, answering a single query would require over 50 hours.

A common method to address clustering and semantic search is to map each sentence to a vector space such that semantically similar sentences are close. Researchers have started to input individual sentences into BERT and to derive fixed-size sentence embeddings. The most commonly used approach is to average the BERT output layer (known as BERT embeddings) or by using the output of the first token (the [CLS] token). As we will show, this common practice yields rather bad sentence embeddings, often worse than averaging GloVe embeddings (Pennington et al., 2014).

To alleviate this issue, we developed SBERT. The siamese network architecture enables that fixed-sized vectors for input sentences can be derived. Using a similarity measure like cosine-similarity or Manhattan / Euclidean distance, semantically similar sentences can be found. These similarity measures can be performed extremely efficient on modern hardware, allowing SBERT to be used for semantic similarity search as well as for clustering. The complexity for finding the

<sup>1</sup>Code available: <https://github.com/UKPLab/sentence-transformers>

<sup>2</sup>With *semantically meaningful* we mean that semantically similar sentences are close in vector space.

most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours with BERT to the computation of 10,000 sentence embeddings (~5 seconds with SBERT) and computing cosine-similarity (~0.01 seconds). By using optimized index structures, finding the most similar Quora question can be reduced from 50 hours to a few milliseconds (Johnson et al., 2017).

We fine-tune SBERT on NLI data, which creates sentence embeddings that significantly outperform other state-of-the-art sentence embedding methods like InferSent (Conneau et al., 2017) and Universal Sentence Encoder (Cer et al., 2018). On seven Semantic Textual Similarity (STS) tasks, SBERT achieves an improvement of 11.7 points compared to InferSent and 5.5 points compared to Universal Sentence Encoder. On SentEval (Conneau and Kiela, 2018), an evaluation toolkit for sentence embeddings, we achieve an improvement of 2.1 and 2.6 points, respectively.

SBERT can be adapted to a specific task. It sets new state-of-the-art performance on a challenging argument similarity dataset (Misra et al., 2016) and on a triplet dataset to distinguish sentences from different sections of a Wikipedia article (Dor et al., 2018).

The paper is structured in the following way: Section 3 presents SBERT, section 4 evaluates SBERT on common STS tasks and on the challenging Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Section 5 evaluates SBERT on SentEval. In section 6, we perform an ablation study to test some design aspect of SBERT. In section 7, we compare the computational efficiency of SBERT sentence embeddings in contrast to other state-of-the-art sentence embedding methods.

## 2 Related Work

We first introduce BERT, then, we discuss state-of-the-art sentence embedding methods.

BERT (Devlin et al., 2018) is a pre-trained transformer network (Vaswani et al., 2017), which set for various NLP tasks new state-of-the-art results, including question answering, sentence classification, and sentence-pair regression. The input for BERT for sentence-pair regression consists of the two sentences, separated by a special [SEP] token. Multi-head attention over 12 (base-model) or 24 layers (large-model) is applied and the output is passed to a simple regression function to derive the final label. Using this setup, BERT set a

new state-of-the-art performance on the Semantic Textual Similarity (STS) benchmark (Cer et al., 2017). RoBERTa (Liu et al., 2019) showed, that the performance of BERT can further improved by small adaptations to the pre-training process. We also tested XLNet (Yang et al., 2019), but it led in general to worse results than BERT.

A large disadvantage of the BERT network structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT. To bypass this limitations, researchers passed single sentences through BERT and then derive a fixed sized vector by either averaging the outputs (similar to average word embeddings) or by using the output of the special CLS token (for example: May et al. (2019); Zhang et al. (2019); Qiao et al. (2019)). These two options are also provided by the popular bert-as-a-service-repository<sup>3</sup>. Up to our knowledge, there is so far no evaluation if these methods lead to useful sentence embeddings.

Sentence embeddings are a well studied area with dozens of proposed methods. Skip-Thought (Kiros et al., 2015) trains an encoder-decoder architecture to predict the surrounding sentences. InferSent (Conneau et al., 2017) uses labeled data of the Stanford Natural Language Inference dataset (Bowman et al., 2015) and the Multi-Genre NLI dataset (Williams et al., 2018) to train a siamese BiLSTM network with max-pooling over the output. Conneau et al. showed, that InferSent consistently outperforms unsupervised methods like SkipThought. Universal Sentence Encoder (Cer et al., 2018) trains a transformer network and augments unsupervised learning with training on SNLI. Hill et al. (2016) showed, that the task on which sentence embeddings are trained significantly impacts their quality. Previous work (Conneau et al., 2017; Cer et al., 2018) found that the SNLI datasets are suitable for training sentence embeddings. Yang et al. (2018) presented a method to train on conversations from Reddit using siamese DAN and siamese transformer networks, which yielded good results on the STS benchmark dataset.

Humeau et al. (2019) addresses the run-time overhead of the cross-encoder from BERT and present a method (poly-encoders) to compute a score between  $m$  context vectors and pre-

---

<sup>3</sup><https://github.com/hanxiao/bert-as-service/>

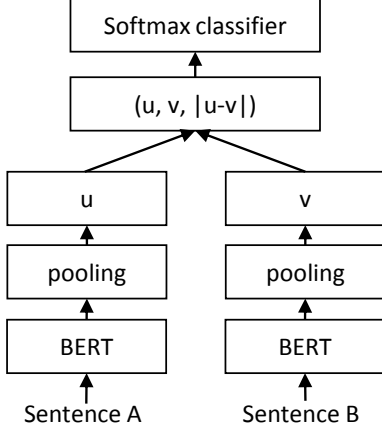


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

computed candidate embeddings using attention. This idea works for finding the highest scoring sentence in a larger collection. However, poly-encoders have the drawback that the score function is not symmetric and the computational overhead is too large for use-cases like clustering, which would require  $O(n^2)$  score computations.

Previous neural sentence embedding methods started the training from a random initialization. In this publication, we use the pre-trained BERT and RoBERTa network and only fine-tune it to yield useful sentence embeddings. This reduces significantly the needed training time: SBERT can be tuned in less than 20 minutes, while yielding better results than comparable sentence embedding methods.

### 3 Model

SBERT adds a pooling operation to the output of BERT / RoBERTa to derive a fixed sized sentence embedding. We experiment with three pooling strategies: Using the output of the CLS-token, computing the mean of all output vectors (MEAN-strategy), and computing a max-over-time of the output vectors (MAX-strategy). The default configuration is MEAN.

In order to fine-tune BERT / RoBERTa, we create siamese and triplet networks (Schroff et al., 2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

The network structure depends on the available

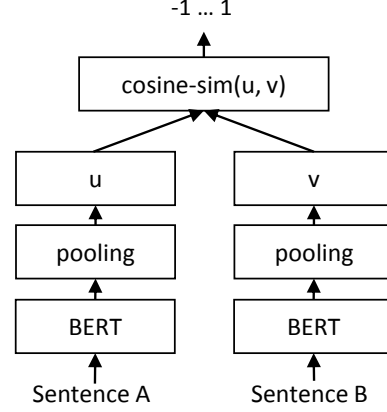


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

training data. We experiment with the following structures and objective functions.

**Classification Objective Function.** We concatenate the sentence embeddings  $u$  and  $v$  with the element-wise difference  $|u - v|$  and multiply it with the trainable weight  $W_t \in \mathbb{R}^{3n \times k}$ :

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

where  $n$  is the dimension of the sentence embeddings and  $k$  the number of labels. We optimize cross-entropy loss. This structure is depicted in Figure 1.

**Regression Objective Function.** The cosine-similarity between the two sentence embeddings  $u$  and  $v$  is computed (Figure 2). We use mean-squared-error loss as the objective function.

**Triplet Objective Function.** Given an anchor sentence  $a$ , a positive sentence  $p$ , and a negative sentence  $n$ , triplet loss tunes the network such that the distance between  $a$  and  $p$  is smaller than the distance between  $a$  and  $n$ . Mathematically, we minimize the following loss function:

$$\max(\|s_a - s_p\| - \|s_a - s_n\| + \epsilon, 0)$$

with  $s_x$  the sentence embedding for  $a/n/p$ ,  $\|\cdot\|$  a distance metric and margin  $\epsilon$ . Margin  $\epsilon$  ensures that  $s_p$  is at least  $\epsilon$  closer to  $s_a$  than  $s_n$ . As metric we use Euclidean distance and we set  $\epsilon = 1$  in our experiments.

#### 3.1 Training Details

We train SBERT on the combination of the SNLI (Bowman et al., 2015) and the Multi-Genre NLI

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	<b>76.69</b>	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	<b>78.46</b>	<b>74.90</b>	80.99	76.25	<b>79.23</b>	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	<b>74.53</b>	77.00	73.18	<b>81.85</b>	<b>76.82</b>	79.10	74.29	<b>76.68</b>

Table 1: Spearman rank correlation  $\rho$  between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as  $\rho \times 100$ . STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

(Williams et al., 2018) dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels *contradiction*, *entailment*, and *neutral*. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. We fine-tune SBERT with a 3-way softmax-classifier objective function for one epoch. We used a batch-size of 16, Adam optimizer with learning rate  $2e-5$ , and a linear learning rate warm-up over 10% of the training data. Our default pooling strategy is MEAN.

## 4 Evaluation - Semantic Textual Similarity

We evaluate the performance of SBERT for common Semantic Textual Similarity (STS) tasks. State-of-the-art methods often learn a (complex) regression function that maps sentence embeddings to a similarity score. However, these regression functions work pair-wise and due to the combinatorial explosion those are often not scalable if the collection of sentences reaches a certain size. Instead, we always use cosine-similarity to compare the similarity between two sentence embeddings. We ran our experiments also with negative Manhattan and negative Euclidean distances as similarity measures, but the results for all approaches remained roughly the same.

### 4.1 Unsupervised STS

We evaluate the performance of SBERT for STS without using any STS specific training data. We use the STS tasks 2012 - 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), the STS benchmark (Cer et al., 2017), and the SICK-Relatedness dataset (Marelli et al., 2014). These datasets provide labels between 0 and 5 on the semantic relatedness of sentence pairs. We showed in (Reimers et al., 2016) that Pearson correlation is badly suited for

STS. Instead, we compute the Spearman’s rank correlation between the cosine-similarity of the sentence embeddings and the gold labels. The setup for the other sentence embedding methods is equivalent, the similarity is computed by cosine-similarity. The results are depicted in Table 1.

The results shows that directly using the output of BERT leads to rather poor performances. Averaging the BERT embeddings achieves an average correlation of only 54.81, and using the CLS-token output only achieves an average correlation of 29.19. Both are worse than computing average GloVe embeddings.

Using the described siamese network structure and fine-tuning mechanism substantially improves the correlation, outperforming both InferSent and Universal Sentence Encoder substantially. The only dataset where SBERT performs worse than Universal Sentence Encoder is SICK-R. Universal Sentence Encoder was trained on various datasets, including news, question-answer pages and discussion forums, which appears to be more suitable to the data of SICK-R. In contrast, SBERT was pre-trained only on Wikipedia (via BERT) and on NLI data.

While RoBERTa was able to improve the performance for several supervised tasks, we only observe minor difference between SBERT and SRoBERTa for generating sentence embeddings.

### 4.2 Supervised STS

The STS benchmark (STSb) (Cer et al., 2017) provides is a popular dataset to evaluate supervised STS systems. The data includes 8,628 sentence pairs from the three categories *captions*, *news*, and *forums*. It is divided into train (5,749), dev (1,500) and test (1,379). BERT set a new state-of-the-art performance on this dataset by passing both sentences to the network and using a simple regres-



sion method for the output.

Model	Spearman
<i>Not trained for STS</i>	
Avg. GloVe embeddings	58.02
Avg. BERT embeddings	46.35
InferSent - GloVe	68.03
Universal Sentence Encoder	74.92
SBERT-NLI-base	77.03
SBERT-NLI-large	79.23
<i>Trained on STS benchmark dataset</i>	
BERT-STSB-base	84.30 $\pm$ 0.76
SBERT-STSB-base	84.67 $\pm$ 0.19
SRoBERTa-STSB-base	<b>84.92</b> $\pm$ 0.34
BERT-STSB-large	<b>85.64</b> $\pm$ 0.81
SBERT-STSB-large	84.45 $\pm$ 0.43
SRoBERTa-STSB-large	85.02 $\pm$ 0.76
<i>Trained on NLI data + STS benchmark data</i>	
BERT-NLI-STSB-base	<b>88.33</b> $\pm$ 0.19
SBERT-NLI-STSB-base	85.35 $\pm$ 0.17
SRoBERTa-NLI-STSB-base	84.79 $\pm$ 0.38
BERT-NLI-STSB-large	<b>88.77</b> $\pm$ 0.46
SBERT-NLI-STSB-large	86.10 $\pm$ 0.13
SRoBERTa-NLI-STSB-large	86.15 $\pm$ 0.35

Table 2: Evaluation on the STS benchmark test set. BERT systems were trained with 10 random seeds and 4 epochs. SBERT was fine-tuned on the STSB dataset, SBERT-NLI was pretrained on the NLI datasets, then fine-tuned on the STSB dataset.

We use the training set to fine-tune SBERT using the regression objective function. At prediction time, we compute the cosine-similarity between the sentence embeddings. All systems are trained with 10 random seeds to counter variances (Reimers and Gurevych, 2018).

The results are depicted in Table 2. We experimented with two setups: Only training on STSB, and first training on NLI, then training on STSB. We observe that the later strategy leads to a slight improvement of 1-2 points. This two-step approach had an especially large impact for the BERT cross-encoder, which improved the performance by 3-4 points. We do not observe a significant difference between BERT and RoBERTa.

### 4.3 Argument Facet Similarity

We evaluate SBERT on the Argument Facet Similarity (AFS) corpus by Misra et al. (2016). The AFS corpus annotated 6,000 sentential argument pairs from social media dialogs on three controversial topics: *gun control*, *gay marriage*, and *death penalty*. The data was annotated on a scale from 0 (“different topic”) to 5 (“completely equivalent”). The similarity notion in the AFS corpus is fairly different to the similarity notion in the STS datasets from SemEval. STS data is usually

descriptive, while AFS data are argumentative excerpts from dialogs. To be considered similar, arguments must not only make similar claims, but also provide a similar reasoning. Further, the lexical gap between the sentences in AFS is much larger. Hence, simple unsupervised methods as well as state-of-the-art STS systems perform badly on this dataset (Reimers et al., 2019).

We evaluate SBERT on this dataset in two scenarios: 1) As proposed by Misra et al., we evaluate SBERT using 10-fold cross-validation. A drawback of this evaluation setup is that it is not clear how well approaches generalize to different topics. Hence, 2) we evaluate SBERT in a cross-topic setup. Two topics serve for training and the approach is evaluated on the left-out topic. We repeat this for all three topics and average the results.

SBERT is fine-tuned using the Regression Objective Function. The similarity score is computed using cosine-similarity based on the sentence embeddings. We also provide the Pearson correlation  $r$  to make the results comparable to Misra et al. However, we showed (Reimers et al., 2016) that Pearson correlation has some serious drawbacks and should be avoided for comparing STS systems. The results are depicted in Table 3.

Unsupervised methods like tf-idf, average GloVe embeddings or InferSent perform rather badly on this dataset with low scores. Training SBERT in the 10-fold cross-validation setup gives a performance that is nearly on-par with BERT.

However, in the cross-topic evaluation, we observe a performance drop of SBERT by about 7 points Spearman correlation. To be considered similar, arguments should address the same claims and provide the same reasoning. BERT is able to use attention to compare directly both sentences (e.g. word-by-word comparison), while SBERT must map individual sentences from an unseen topic to a vector space such that arguments with similar claims and reasons are close. This is a much more challenging task, which appears to require more than just two topics for training to work on-par with BERT.

### 4.4 Wikipedia Sections Distinction

Dor et al. (2018) use Wikipedia to create a thematically fine-grained train, dev and test set for sentence embeddings methods. Wikipedia articles are separated into distinct sections focusing on certain aspects. Dor et al. assume that sen-

Model	$r$	$\rho$
<i>Unsupervised methods</i>		
tf-idf	46.77	42.95
Avg. GloVe embeddings	32.40	34.00
InferSent - GloVe	27.08	26.63
<i>10-fold Cross-Validation</i>		
SVR (Misra et al., 2016)	63.33	-
BERT-AFS-base	77.20	74.84
SBERT-AFS-base	76.57	74.13
BERT-AFS-large	78.68	76.38
SBERT-AFS-large	77.85	75.93
<i>Cross-Topic Evaluation</i>		
BERT-AFS-base	58.49	57.23
SBERT-AFS-base	52.34	50.65
BERT-AFS-large	62.02	60.34
SBERT-AFS-large	53.82	53.10

Table 3: Average Pearson correlation  $r$  and average Spearman’s rank correlation  $\rho$  on the Argument Facet Similarity (AFS) corpus (Misra et al., 2016). Misra et al. proposes 10-fold cross-validation. We additionally evaluate in a cross-topic scenario: Methods are trained on two topics, and are evaluated on the third topic.

tences in the same section are thematically closer than sentences in different sections. They use this to create a large dataset of weakly labeled sentence triplets: The anchor and the positive example come from the same section, while the negative example comes from a different section of the same article. For example, from the Alice Arnold article: Anchor: *Arnold joined the BBC Radio Drama Company in 1988.*, positive: *Arnold gained media attention in May 2012.*, negative: *Balding and Arnold are keen amateur golfers.*

We use the dataset from Dor et al. We use the Triplet Objective, train SBERT for one epoch on the about 1.8 Million training triplets and evaluate it on the 222,957 test triplets. Test triplets are from a distinct set of Wikipedia articles. As evaluation metric, we use accuracy: Is the positive example closer to the anchor than the negative example?

Results are presented in Table 4. Dor et al. fine-tuned a BiLSTM architecture with triplet loss to derive sentence embeddings for this dataset. As the table shows, SBERT clearly outperforms the BiLSTM approach by Dor et al.

## 5 Evaluation - SentEval

SentEval (Conneau and Kiela, 2018) is a popular toolkit to evaluate the quality of sentence embeddings. Sentence embeddings are used as features for a logistic regression classifier. The logistic regression classifier is trained on various tasks in a 10-fold cross-validation setup and the prediction accuracy is computed for the test-fold.

Model	Accuracy
mean-vectors	0.65
skip-thoughts-CS	0.62
Dor et al.	0.74
SBERT-WikiSec-base	0.8042
SBERT-WikiSec-large	<b>0.8078</b>
SROBERTa-WikiSec-base	0.7945
SROBERTa-WikiSec-large	0.7973

Table 4: Evaluation on the Wikipedia section triplets dataset (Dor et al., 2018). SBERT trained with triplet loss for one epoch.

The purpose of SBERT sentence embeddings are not to be used for transfer learning for other tasks. Here, we think fine-tuning BERT as described by Devlin et al. (2018) for new tasks is the more suitable method, as it updates all layers of the BERT network. However, SentEval can still give an impression on the quality of our sentence embeddings for various tasks.

We compare the SBERT sentence embeddings to other sentence embeddings methods on the following seven SentEval transfer tasks:

- **MR**: Sentiment prediction for movie reviews snippets on a five star scale (Pang and Lee, 2005).
- **CR**: Sentiment prediction of customer product reviews (Hu and Liu, 2004).
- **SUBJ**: Subjectivity prediction of sentences from movie reviews and plot summaries (Pang and Lee, 2004).
- **MPQA**: Phrase level opinion polarity classification from newswire (Wiebe et al., 2005).
- **SST**: Stanford Sentiment Treebank with binary labels (Socher et al., 2013).
- **TREC**: Fine grained question-type classification from TREC (Li and Roth, 2002).
- **MRPC**: Microsoft Research Paraphrase Corpus from parallel news sources (Dolan et al., 2004).

The results can be found in Table 5. SBERT is able to achieve the best performance in 5 out of 7 tasks. The average performance increases by about 2 percentage points compared to InferSent as well as the Universal Sentence Encoder. Even though transfer learning is not the purpose of SBERT, it outperforms other state-of-the-art sentence embeddings methods on this task.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg.
Avg. GloVe embeddings	77.25	78.30	91.17	87.85	80.18	83.0	72.87	81.52
Avg. fast-text embeddings	77.96	79.23	91.68	87.81	82.15	83.6	74.49	82.42
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
InferSent - GloVe	81.57	86.54	92.50	<b>90.38</b>	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	<b>93.2</b>	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	<b>76.00</b>	87.41
SBERT-NLI-large	<b>84.88</b>	<b>90.07</b>	<b>94.52</b>	90.33	<b>90.66</b>	87.4	75.94	<b>87.69</b>

Table 5: Evaluation of SBERT sentence embeddings using the SentEval toolkit. SentEval evaluates sentence embeddings on different sentence classification tasks by training a logistic regression classifier using the sentence embeddings as features. Scores are based on a 10-fold cross-validation.

It appears that the sentence embeddings from SBERT capture well sentiment information: We observe large improvements for all sentiment tasks (MR, CR, and SST) from SentEval in comparison to InferSent and Universal Sentence Encoder.

The only dataset where SBERT is significantly worse than Universal Sentence Encoder is the TREC dataset. Universal Sentence Encoder was pre-trained on question-answering data, which appears to be beneficial for the question-type classification task of the TREC dataset.

Average BERT embeddings or using the CLS-token output from a BERT network achieved bad results for various STS tasks (Table 1), worse than average GloVe embeddings. However, for SentEval, average BERT embeddings and the BERT CLS-token output achieves decent results (Table 5), outperforming average GloVe embeddings. The reason for this are the different setups. For the STS tasks, we used cosine-similarity to estimate the similarities between sentence embeddings. Cosine-similarity treats all dimensions equally. In contrast, SentEval fits a logistic regression classifier to the sentence embeddings. This allows that certain dimensions can have higher or lower impact on the classification result.

We conclude that average BERT embeddings / CLS-token output from BERT return sentence embeddings that are infeasible to be used with cosine-similarity or with Manhattan / Euclidean distance. For transfer learning, they yield slightly worse results than InferSent or Universal Sentence Encoder. However, using the described fine-tuning setup with a siamese network structure on NLI datasets yields sentence embeddings that achieve a new state-of-the-art for the SentEval toolkit.

## 6 Ablation Study

We have demonstrated strong empirical results for the quality of SBERT sentence embeddings. In

this section, we perform an ablation study of different aspects of SBERT in order to get a better understanding of their relative importance.

We evaluated different pooling strategies (MEAN, MAX, and CLS). For the classification objective function, we evaluate different concatenation methods. For each possible configuration, we train SBERT with 10 different random seeds and average the performances.

The objective function (classification vs. regression) depends on the annotated dataset. For the classification objective function, we train SBERT-base on the SNLI and the Multi-NLI dataset. For the regression objective function, we train on the training set of the STS benchmark dataset. Performances are measured on the development split of the STS benchmark dataset. Results are shown in Table 6.

	NLI	STSb
<i>Pooling Strategy</i>		
MEAN	<b>80.78</b>	<b>87.44</b>
MAX	79.07	69.92
CLS	79.80	86.62
<i>Concatenation</i>		
$(u, v)$	66.04	-
$( u - v )$	69.78	-
$(u * v)$	70.54	-
$( u - v , u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v,  u - v )$	<b>80.78</b>	-
$(u, v,  u - v , u * v)$	80.44	-

Table 6: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman’s rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy.

When trained with the classification objective function on NLI data, the pooling strategy has a rather minor impact. The impact of the concatenation mode is much larger. InferSent (Conneau

et al., 2017) and Universal Sentence Encoder (Cer et al., 2018) both use  $(u, v, |u - v|, u * v)$  as input for a softmax classifier. However, in our architecture, adding the element-wise  $u * v$  decreased the performance.

The most important component is the element-wise difference  $|u - v|$ . Note, that the concatenation mode is only relevant for training the softmax classifier. At inference, when predicting similarities for the STS benchmark dataset, only the sentence embeddings  $u$  and  $v$  are used in combination with cosine-similarity. The element-wise difference measures the distance between the dimensions of the two sentence embeddings, ensuring that similar pairs are closer and dissimilar pairs are further apart.

When trained with the regression objective function, we observe that the pooling strategy has a large impact. There, the MAX strategy perform significantly worse than MEAN or CLS-token strategy. This is in contrast to (Conneau et al., 2017), who found it beneficial for the BiLSTM-layer of InferSent to use MAX instead of MEAN pooling.

## 7 Computational Efficiency

Sentence embeddings need potentially be computed for Millions of sentences, hence, a high computation speed is desired. In this section, we compare SBERT to average GloVe embeddings, InferSent (Conneau et al., 2017), and Universal Sentence Encoder (Cer et al., 2018).

For our comparison we use the sentences from the STS benchmark (Cer et al., 2017). We compute average GloVe embeddings using a simple for-loop with python dictionary lookups and NumPy. InferSent<sup>4</sup> is based on PyTorch. For Universal Sentence Encoder, we use the TensorFlow Hub version<sup>5</sup>, which is based on TensorFlow. SBERT is based on PyTorch. For improved computation of sentence embeddings, we implemented a smart batching strategy: Sentences with similar lengths are grouped together and are only padded to the longest element in a mini-batch. This drastically reduces computational overhead from padding tokens.

Performances were measured on a server with Intel i7-5820K CPU @ 3.30GHz, Nvidia Tesla

V100 GPU, CUDA 9.2 and cuDNN. The results are depicted in Table 7.

Model	CPU	GPU
Avg. GloVe embeddings	6469	-
InferSent	137	1876
Universal Sentence Encoder	67	1318
SBERT-base	44	1378
SBERT-base - smart batching	83	2042

Table 7: Computation speed (sentences per second) of sentence embedding methods. Higher is better.

On CPU, InferSent is about 65% faster than SBERT. This is due to the much simpler network architecture. InferSent uses a single BiLSTM layer, while BERT uses 12 stacked transformer layers. However, an advantage of transformer networks is the computational efficiency on GPUs. There, SBERT with smart batching is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. Smart batching achieves a speed-up of 89% on CPU and 48% on GPU. Average GloVe embeddings is obviously by a large margin the fastest method to compute sentence embeddings.

## 8 Conclusion

We showed that BERT out-of-the-box maps sentences to a vector space that is rather unsuitable to be used with common similarity measures like cosine-similarity. The performance for seven STS tasks was below the performance of average GloVe embeddings.

To overcome this shortcoming, we presented Sentence-BERT (SBERT). SBERT fine-tunes BERT in a siamese / triplet network architecture. We evaluated the quality on various common benchmarks, where it could achieve a significant improvement over state-of-the-art sentence embeddings methods. Replacing BERT with RoBERTa did not yield a significant improvement in our experiments.

SBERT is computationally efficient. On a GPU, it is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder. SBERT can be used for tasks which are computationally not feasible to be modeled with BERT. For example, clustering of 10,000 sentences with hierarchical clustering requires with BERT about 65 hours, as around 50 Million sentence combinations must be computed. With SBERT, we were able to reduce the effort to about 5 seconds.

<sup>4</sup><https://github.com/facebookresearch/InferSent>

<sup>5</sup><https://tfhub.dev/google/universal-sentence-encoder-large/3>