Sentence-BERT

Data Circles Journal Club 7-27-22

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

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 for **semantic similarity search** as well as for unsupervised tasks like **clustering**.

Intro & Related Work

SBERT = Sentence-BERT

BERT = Bidirectional Encoder Representations from Transformers (2018)

RoBERTa = Robust BERT (2019)

GloVe = Global Vectors (2014)

InferSent: Sentence Embedding GloVe + BiLSTM

STS = Semantic Textual Similarity

SNLI = Stanford Natural Language Inference

MNLI = Multi-Genre Natural Language Inference

NLP = Natural Language Processing

BERT / RoBERTa

Bidirectional Encoder Representations from Transformers

Many-to-1

- Sentiment analysis
- Classification

Masked (Cloze) Language Modeling

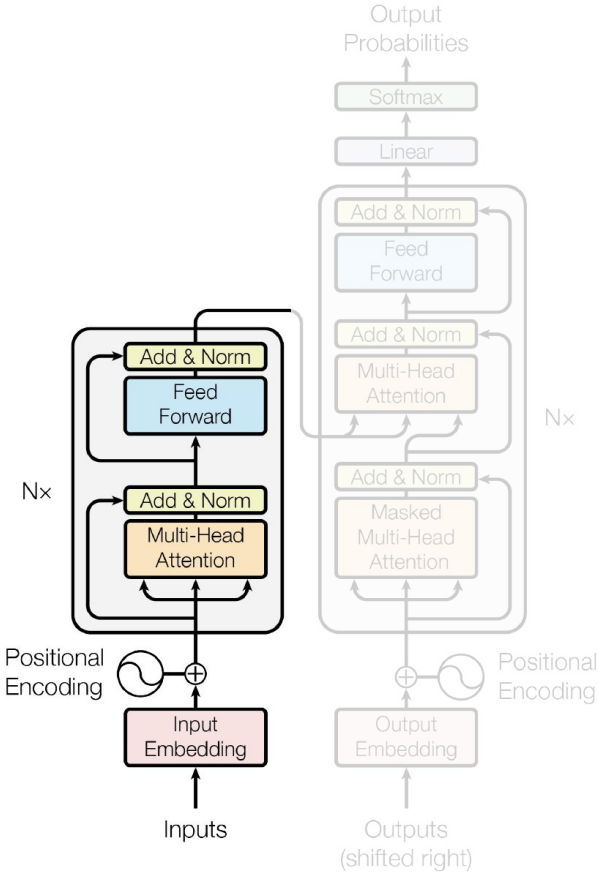
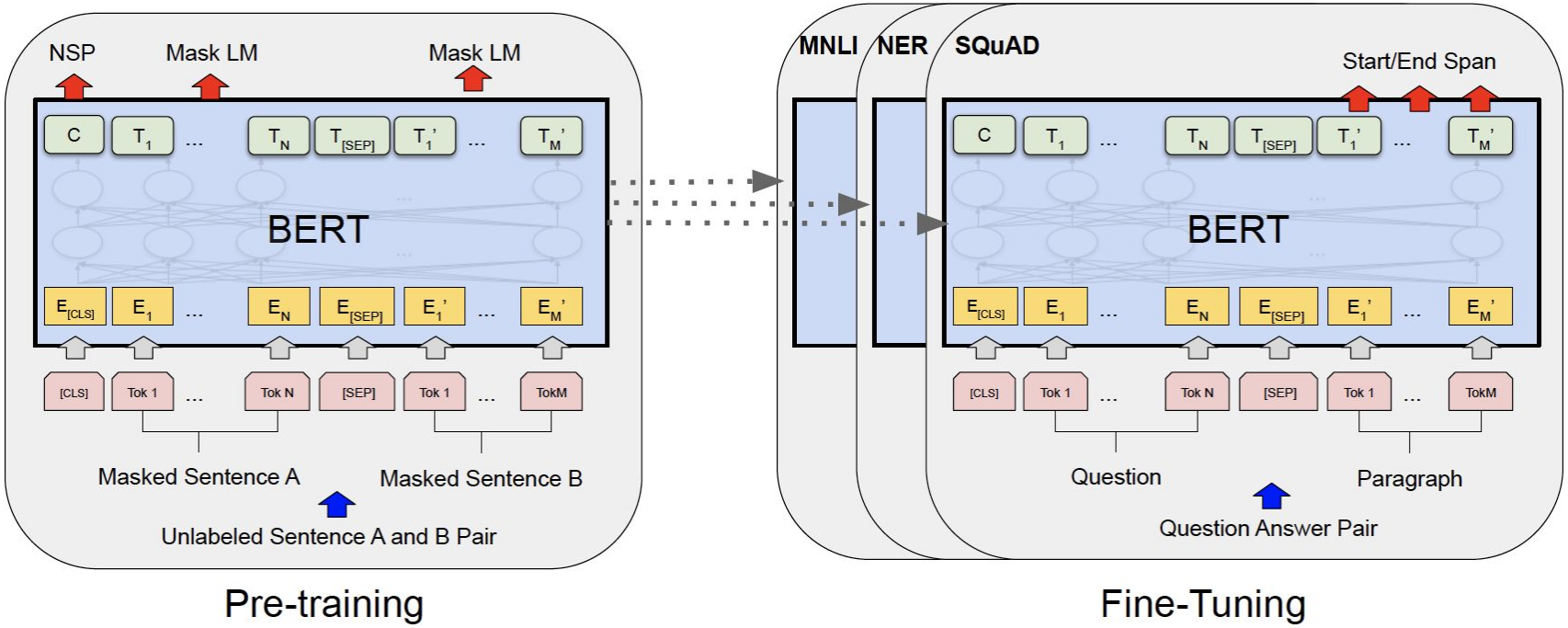


Figure 1: The Transformer - model architecture.

BERT / RoBERTa



BERT / RoBERTa

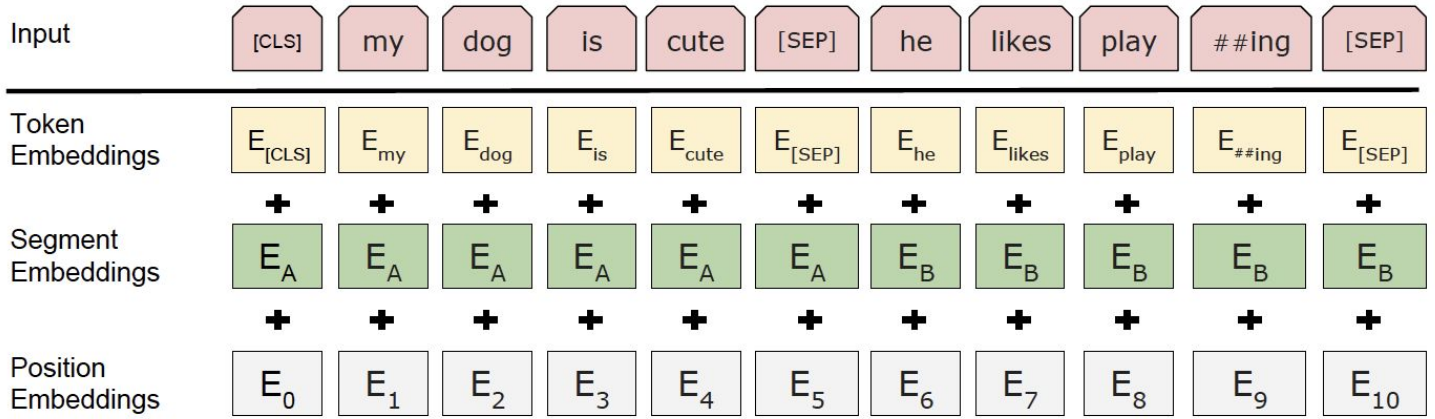


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

SBERT Model

3 Structures & Objective Functions

- Classification
- Regression
- Triplet

SBERT Model

Classification Objection Function

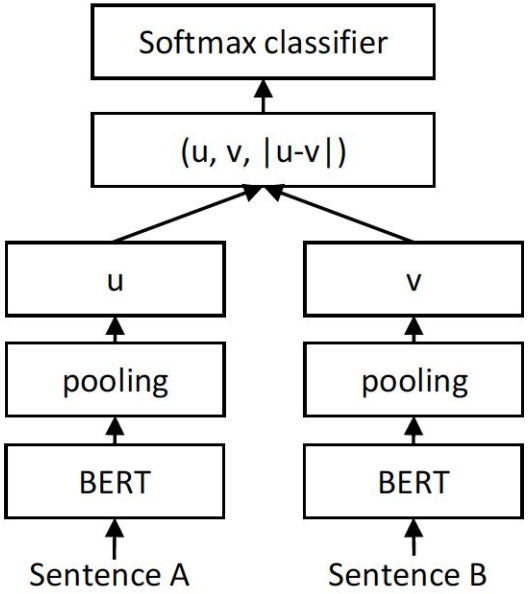


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

SBERT Model

Regression Objective Function

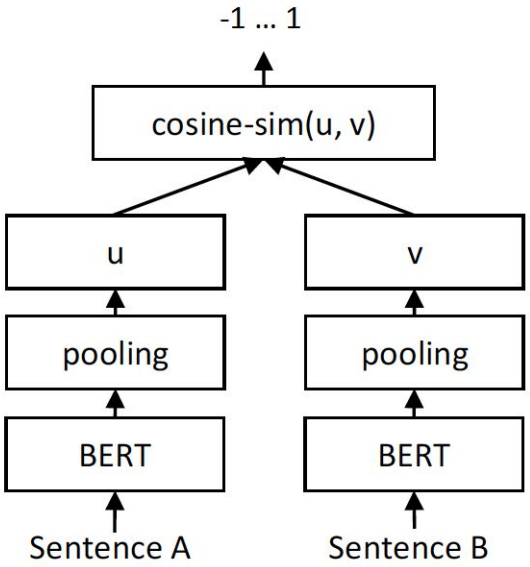


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

SBERT Model

Triplet Objective Function

- *a* anchor
- *p* positive sentence
- *n* negative sentence
- $max(\| s_a - s_p \| - \| s_a - s_n \| + \epsilon, 0)$

Evaluation: Semantic Textual Similarity

4 Methods

- Unsupervised STS
- Supervised STS
- Argument Facet Similarity (AFS)
- Wikipedia Sections

Dataset Preview

Subset: mnli

Split: train

premise (string)	hypothesis (string)	label (class label)	idx (int)
Conceptually cream skimming has two basic dimensions - product and geography.	Product and geography are what make cream skimming work.	1 (neutral)	0
you know during the season and i guess at at your level uh you lose them to the next level if if…	You lose the things to the following level if the people recall.	0 (entailment)	1
One of our number will carry out your instructions minutely.	A member of my team will execute your orders with immense precision.	0 (entailment)	2
How do you know? All this is their information again.	This information belongs to them.	0 (entailment)	3
yeah i tell you what though if you go price some of those tennis shoes i can see why now you know…	The tennis shoes have a range of prices.	1 (neutral)	4
my walkman broke so i'm upset now i just have to turn the stereo up real loud	I'm upset that my walkman broke and now I have to turn the stereo up really loud.	0 (entailment)	5
But a few Christian mosaics survive above the apse is the Virgin with the infant Jesus, with…	Most of the Christian mosaics were destroyed by Muslims.	1 (neutral)	6

Evaluation: Semantic Textual Similarity

Unsupervised STS

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	**76.69**	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	**78.46**	**74.90**	80.99	76.25	**79.23**	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	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ 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.

Dataset Preview

Subset: stsb

Split: train

sentence1 (string)	sentence2 (string)	label (float)	idx (int)
A plane is taking off.	An air plane is taking off.	5	0
A man is playing a large flute.	A man is playing a flute.	3.8	1
A man is spreading shreded cheese on a pizza.	A man is spreading shredded cheese on an uncooked pizza.	3.8	2
Three men are playing chess.	Two men are playing chess.	2.6	3
A man is playing the cello.	A man seated is playing the cello.	4.25	4
Some men are fighting.	Two men are fighting.	4.25	5
A man is smoking.	A man is skating.	0.5	6
The man is playing the piano.	The man is playing the guitar.	1.6	7
A man is playing on a guitar and singing.	A woman is playing an acoustic guitar and singing.	2.2	8
A person is throwing a cat on to the ceiling.	A person throws a cat on the ceiling.	5	9

Evaluation

Supervised STS benchmark

Model	Spearman
Not trained for STS	
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
Trained on STS benchmark dataset	
BERT-STSb-base	84.30 ± 0.76
SBERT-STSb-base	84.67 ± 0.19
SRoBERTa-STSb-base	**84.92** ± 0.34
BERT-STSb-large	**85.64** ± 0.81
SBERT-STSb-large	84.45 ± 0.43
SRoBERTa-STSb-large	85.02 ± 0.76
Trained on NLI data + STS benchmark data	
BERT-NLI-STSb-base	**88.33** ± 0.19
SBERT-NLI-STSb-base	85.35 ± 0.17
SRoBERTa-NLI-STSb-base	84.79 ± 0.38
BERT-NLI-STSb-large	**88.77** ± 0.46
SBERT-NLI-STSb-large	86.10 ± 0.13
SRoBERTa-NLI-STSb-large	86.15 ± 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.

Evaluation

Argument Facet Similarity (AFS)

3 controversial topics:
gun control, gay marriage, death penalty
Different vs equivalent claims + reasoning

Model	r	ρ
Unsupervised methods		
tf-idf	46.77	42.95
Avg. GloVe embeddings	32.40	34.00
InferSent - GloVe	27.08	26.63
10-fold Cross-Validation		
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
Cross-Topic Evaluation		
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 ρ 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.

Evaluation

Wikipedia Section Distinction

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:

a: Arnold joined the BBC Radio Drama Company in 1988
p: Arnold gained media attention in May 2012.
n: Balding and Arnold are keen amateur golfers.

Model	Accuracy
mean-vectors	0.65
skip-thoughts-CS	0.62
Dor et al.	0.74
SBERT-WikiSec-base	0.8042
SBERT-WikiSec-large	**0.8078**
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.

Evaluation: SentEval

Toolkit to evaluate quality of sentence embeddings

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	**90.38**	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	**93.2**	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	**76.00**	87.41
SBERT-NLI-large	**84.88**	**90.07**	**94.52**	90.33	**90.66**	87.4	75.94	**87.69**

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.

Ablation Study

Pooling strategies: MEAN, MAX, CLS

10 different random seeds, average performance

Classification Objective trained on SNLI + MNLI

Regression Objective trained on STSb

	NLI	STSb
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(u, v)	66.04	-
$(\|u - v\|)$	69.78	-
$(u * v)$	70.54	-
$(\|u - v\|, u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v, \|u - v\|)$	**80.78**	-
$(u, v, \|u - v\|, u * v)$	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.

Computation Efficiency

Server:
Intel i7-5820K CPU @ 3.30GHz, Nvidia Tesla V100 GPU, CUDA 9.2 and cuDNN

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.

Example: Paraphrase Mining

```
!pip install sentence-transformers
from sentence_transformers import SentenceTransformer, util

df = pd.DataFrame(requests.get("https://stampy.ai/w/api.php").json())
checkpoint = "paraphrases-multi-qa-mpn"
#@param ['distilbert-base-nli-stsb-quora-ranking', 'multi-qa-mpnet-base-dot-v1', 'all-MiniLM-L6-v2']
model = SentenceTransformer(checkpoint)

# Single list of sentences - possible tens of thousands of sentences
sentences = df["fulltext"].values.tolist()
paraphrases = util.paraphrase_mining(model, sentences)

for paraphrase in paraphrases[0:100]:
   score, i, j = paraphrase
   print(f"{df['fulltext'][i]}\n{df['fulltext'][j]}\nscore:{score:.2f}\n")
```

https://sbert.net/

Example: Duplicate Questions

Question1	Question2	Score
Who helped create Stampy?	Who created Stampy?	0.98
Is humanity doomed?	How doomed is humanity?	0.95
What is a canonical question on Stampy's Wiki?	What is a canonical version of a question on Stampy's Wiki?	0.93
Why can't we just "put the AI in a box" so it can't influence the outside world?	Couldn't we keep the AI in a box and never give it the ability to manipulate the external world?	0.92
How might a superintelligence technologically manipulate humans?	How might a superintelligence socially manipulate humans?	0.92
Why is AI Safety important?	Why is safety important for smarter-than-human AI?	0.91
Can we tell an AI just to figure out what we want, then do that?	Can we just tell an AI to do what we want?	0.90
What is AI Safety?	Why is AI Safety important?	0.90

Example: Transformer Setup

```
!pip install datasets transformers[sentencepiece]
!pip install faiss-gpu
from transformers import AutoTokenizer, AutoModel

df = pd.DataFrame(requests.get("https://stampy.ai/w/api.php").json())
checkpoint = "paraphrases-multi-qa-mpn"
#@param ['distilbert-base-nli-stsb-quora-ranking', 'multi-qa-mpnet-base-dot-v1', 'all-MiniLM-L6-v2']

# load pretrained tokenizer and model
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModel.from_pretrained(checkpoint)
model.to(device)
dataset = Dataset.from_pandas(df)

# embed entire set of stampy questions then pkl to file
embeddings_dataset = dataset.map( lambda x: {"embeddings":
     get_embeddings(x["text"]).detach().cpu().numpy()[0]})
embeddings_dataset.add_faiss_index(column="embeddings")
```

Example: Semantic Search

```
question_embedding = get_embeddings([question]).cpu().detach().numpy()

scores, samples = embeddings_dataset.get_nearest_examples("embeddings", question_embedding, k=6)

samples_df = pd.DataFrame.from_dict(samples)
samples_df["scores"] = scores
samples_df.sort_values("scores", ascending=True, inplace=True)

for _, row in samples_df.iterrows():
   print(f"({row.scores:.2f})\t{row.fulltext}")
```

Sentence-BERT (SBERT) *fine-tunes BERT in a* ***siamese / triplet*** *network architecture. We evaluated the quality on various common* ***semantic textual search*** *benchmarks, where it could achieve a significant improvement over state-of-the-art* ***sentence embeddings*** *methods. SBERT is* ***computationally efficient.***

Discussion

Personal experiences?

Potential applications?

Questions?

Key takeaways?