SBERT

Introduction

- SentenceTransformers is a Python framework for state-of-the-art sentence, text and image embeddings.
- This framework can be used to compute sentence / text embeddings for more than 100 languages.
- These embeddings can then be compared e.g. with **cosine-similarity** to find sentences with a similar meaning. This can be useful for **semantic textual similar, semantic search**

Previous Challenges

- **Finding the two most similar sentences:**

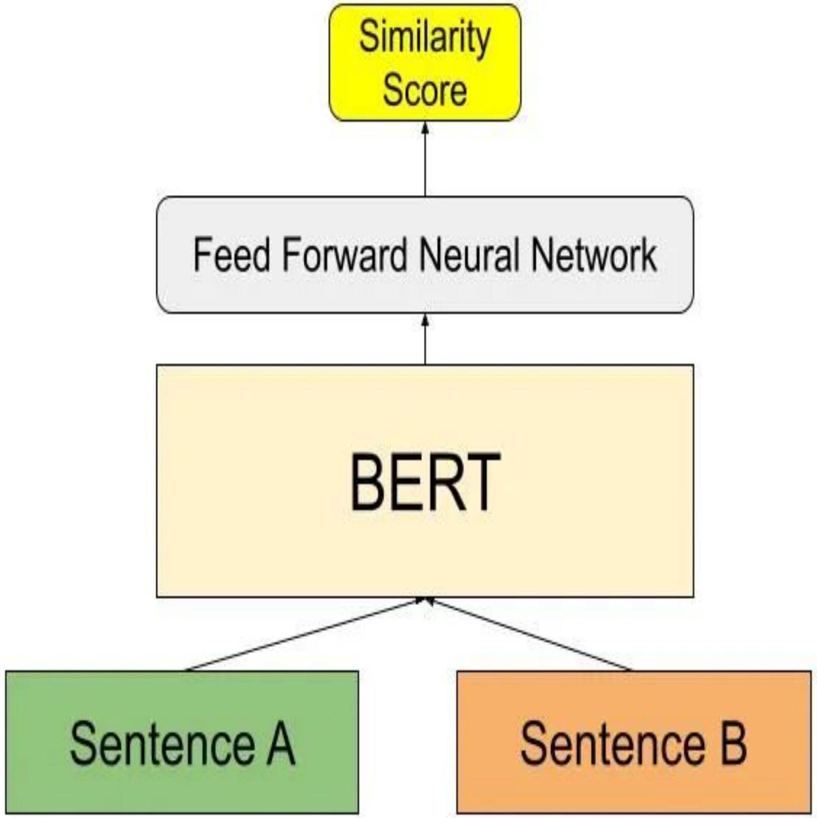
  In a dataset of n. This would require us to feed each unique pair through BERT to finds its similarity score and then compare it to all other scores. For n sentences would that result in **$n(n - 1)/2$.**

  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.**

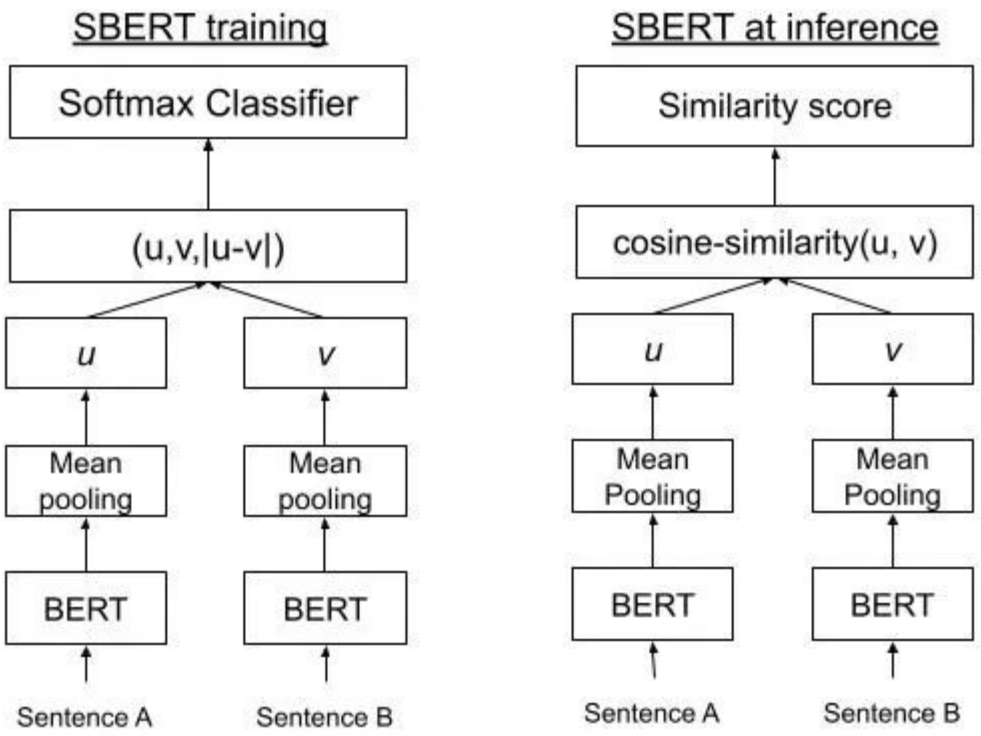
- **Performing semantic search:**
  This task entails finding the most similar sentence in a dataset given a query. Ideally, this would be done by comparing the query to all existing sentences.

  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.**

The BERT cross-encoder consists of a standard BERT model that takes in as input the two sentences, A and B, separated by a [SEP] token. On top of the BERT is a feedforward layer that outputs a **similarity score.**

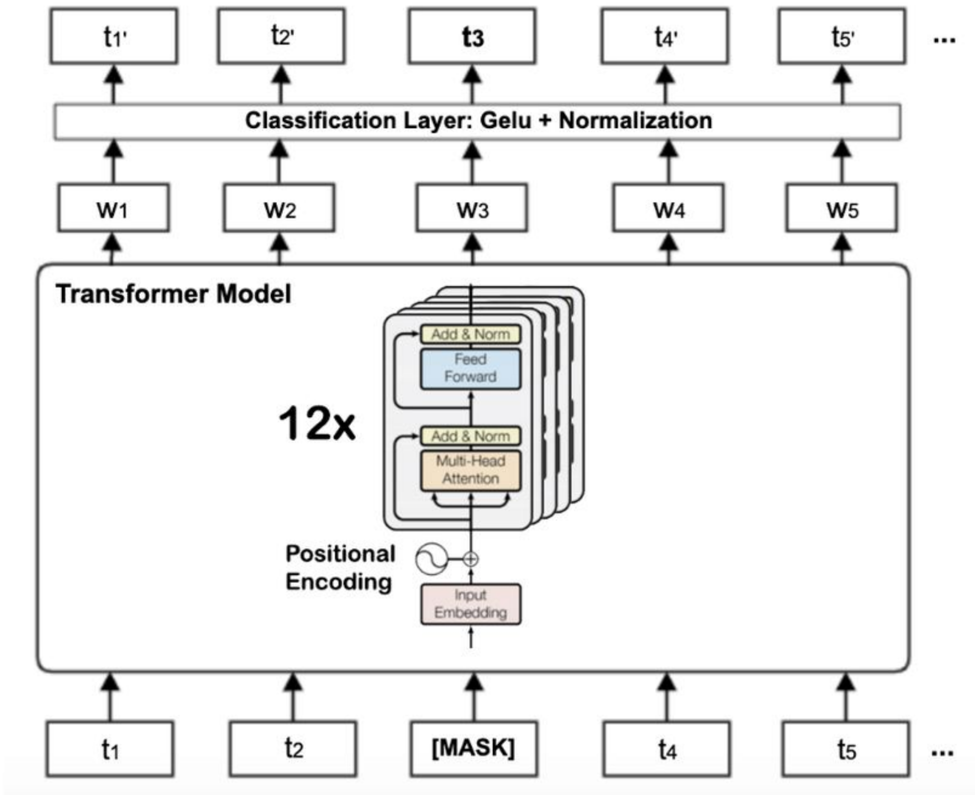


Architecture



- **What are Siamese Networks?** two (or more) ***identical subnetworks/models***, share/have the same parameters/weights. Parameter updating is mirrored across both sub-models.
- **Unlike BERT, SBERT** uses a **siamese architecture**, where it contains 2 BERT architectures that are essentially identical and share the same weights, and SBERT processes 2 sentences as pairs during training.

BERT


t1'
t2'
t3
t4'
t5'
...
Classification Layer: Gelu + Normalization
W1
W2
W3
W4
W5
Transformer Model
12x
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
t1
t2
[MASK]
t4
t5
...

Semantic Search

- **Symmetric semantic search :** query and the entries in corpus are of about the same length and have the same amount of content.
  Ex:

  "How to learn Python online?"

  "How to learn Python on the web?"

- **Asymmetric semantic search :** usually have a **short query** (like a question or some keywords) and we want to find a longer paragraph answering the query.
  Ex:

  query : *"What is Python"*

  and we want to find the paragraph

  paragraph : *"Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy ...".*

Models for Semantic Search

The **all-*** models where trained on all available training data (***more than 1 billion training pairs***) and are designed as **general purpose** models. The **all-mpnet-base-v2** model provides the best quality, while **all-MiniLM-L6-v2** is 5 times faster and still offers good quality.

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB

all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB

all-mpnet-base-v2

Description:	All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs.
Base Model:	microsoft/mpnet-base
Max Sequence Length:	384
Dimensions:	768
Normalized Embeddings:	true
Suitable Score Functions:	dot-product (`util.dot_score`), cosine-similarity (`util.cos_sim`), euclidean distance
Size:	420 MB
Pooling:	Mean Pooling
Training Data:	1B+ training pairs. For details, see model card.
Model Card:	https://huggingface.co/sentence-transformers/all-mpnet-base-v2

util.semantic_search

This function performs a ***cosine similarity search between a list of query embeddings and a list of corpus embeddings***. It can be used for Information Retrieval / Semantic Search for corpora up to about 1 Million entries.

Parameters

- **query_embeddings** – A 2 dimensional tensor with the query embeddings.
- **corpus_embeddings** – A 2 dimensional tensor with the corpus embeddings.
- **query_chunk_size** – Process 100 queries simultaneously. Increasing that value increases the speed, but requires more memory.
- **corpus_chunk_size** – Scans the corpus 100k entries at a time. Increasing that value increases the speed, but requires more memory.
- **top_k** – Retrieve top k matching entries.
- **score_function** – Function for computing scores. By default, cosine similarity.

Returns

Returns a list with one entry for each query. Each entry is a list of dictionaries with the keys 'corpus_id' and 'score', sorted by decreasing cosine similarity scores.

Thank You