SBERT: How to Use Sentence Embeddings to Solve Real-World Problems

anirban sen · Following
5 min read · Jul 29

5

Photo by Ross Joyner on Unsplash

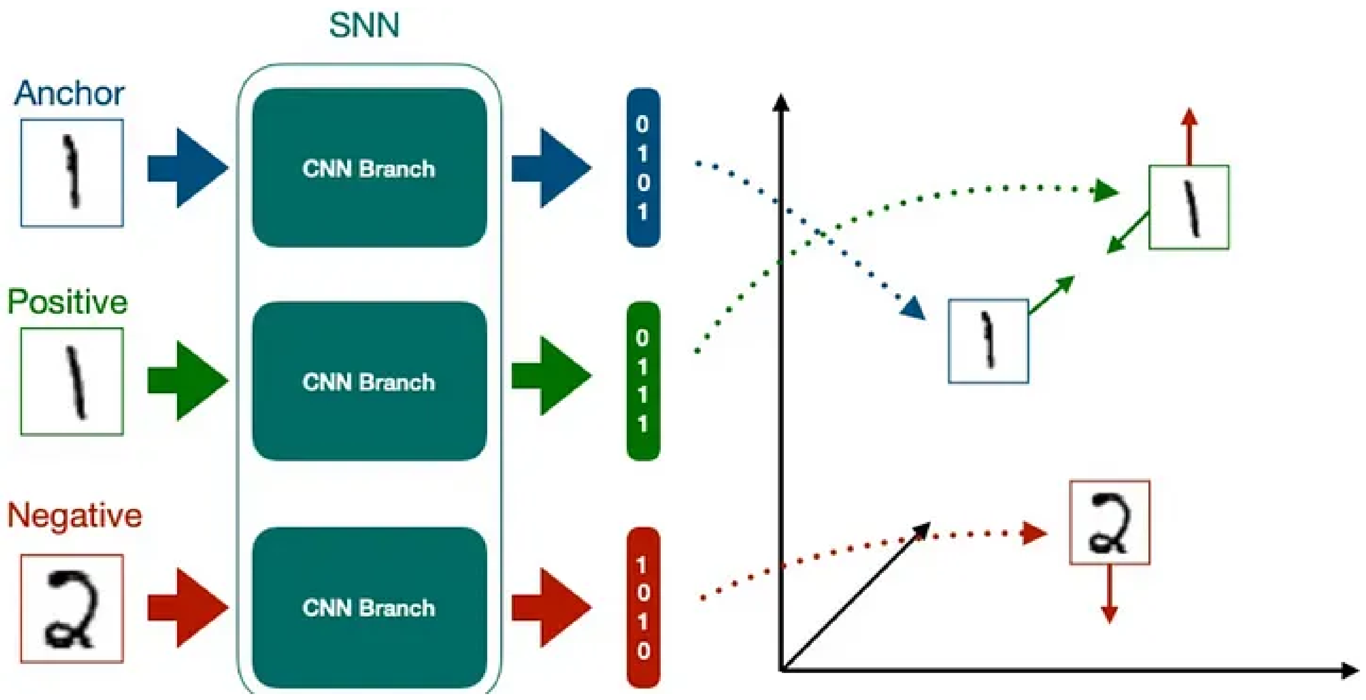
Ofcourse Transformers need no introduction (with the rise of ChatGPT i.e. Generative Pretrained Transformer for Chat). We have already read about Transformers & BERT in the text classification using BERT blog. In this one, we will learn about SentenceBERT or so called SBERT and also see how we can use it in our code.

What is SBERT?

SBERT is a framework for computing sentence embeddings using the BERT model which can be used for various downstream tasks but made computationally efficient with the use of Siamese Networks.

What is Siamese Network?

A Siamese Network is a type of network architecture that contains two or more identical subnetworks used to generate feature vectors for each input and compare them. Siamese Networks can be applied to different use cases, like detecting duplicates, finding anomalies, and face recognition. We provide three items to the model, where two of them will be similar (*anchor* and *positive* samples), and the third will be unrelated (a *negative* example.) Our goal is for the model to learn to estimate the similarity between items by minimising the distance between similar items and increasing the distance between dissimilar using something called Triplet loss function. Netflix utilises SNNs to generate user recommendations.

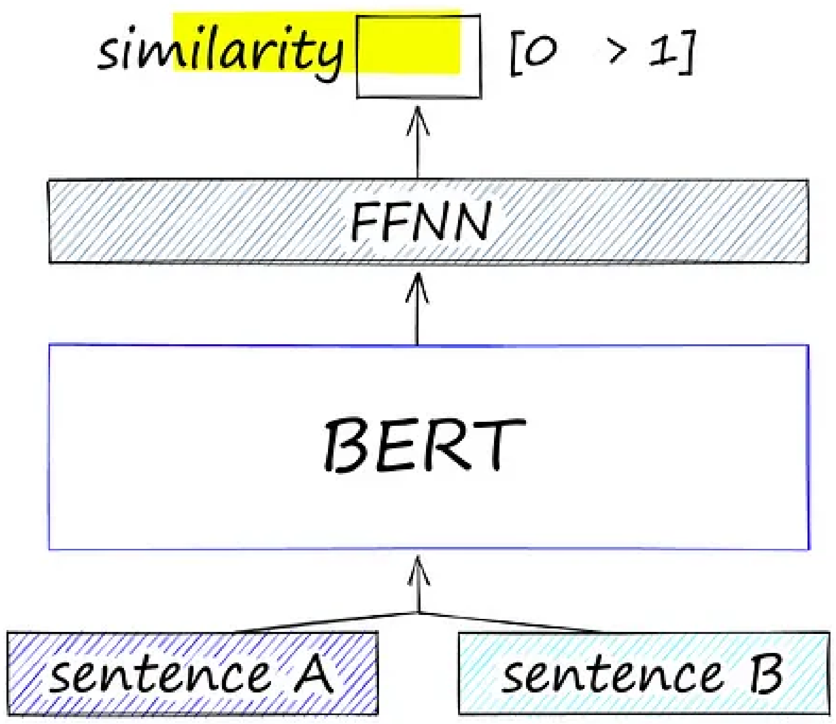


Siamese Network for Images

Triplet Loss — $L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \text{margin}, 0)$ where A is an *anchor input,* P is the positive input same as class A, N is a negative input from a different class from A, f is the embedding.

Why do we need SentenceBERT?
SBERT can be used in various usecases like Semantic textual similarity (STS), Semantic search, Text/Document Clustering, Natural language inference(NLI) etc. Some real-life use-cases are — Similar Product recommendations, Customer support, Personalized recommendations etc.

Okay but why not use normal BERT?
BERT is essentially a token embedder. So if we have to calculate similarity between 2 sentences using base BERT model — Two sentences are passed to the transformer network and the target value is predicted.



The BERT cross-encoder architecture consists of a BERT model which consumes sentences A and B. Both are processed in the same sequence, separated by a [SEP] token. All of this is followed by a feedforward NN classifier that outputs a similarity score.

https://www.pinecone.io/learn/series/nlp/sentence-embeddings/

Finding the most similar pair in a collection of 10k sentences requires about 50 million inference computations (~65 hours) with BERT.

Another alternate solution is to take out the average of the word/token embeddings and calculate similarity score between just the embeddings. The results of this solution came out to be worse than averaging GloVe embeddings.

Okay then how does it solve the problem?

SBERT uses the BERT model puts it in something called *siamese* architecture and fine-tunes it on sentence pairs. We can think of this as having two identical BERTs in parallel that share the exact same network weights. SBERT adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding (for e.g. 768 for bert-base by default). The default pooling strategy is MEAN

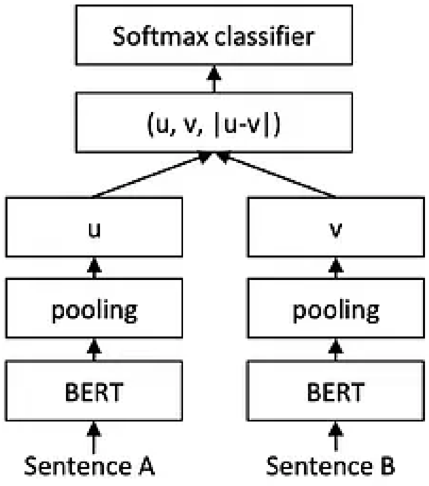


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

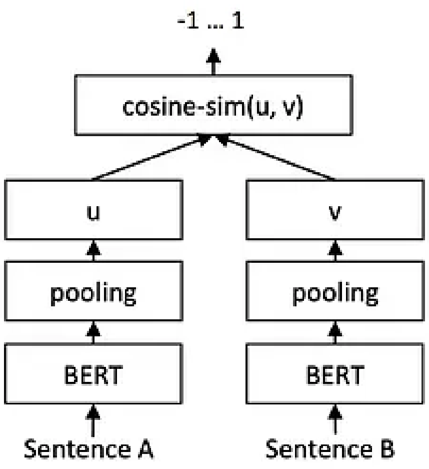


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

https://arxiv.org/pdf/1908.10084.pdf

SBERT is fine-tuned with a 3-way softmax classifier [Figure 1] (labels being contradiction, eintailment, and neutral) objective function on the combination of the SNLI and the Multi-Genre NLI dataset (~1M sentence annotated pairs combined). MultiNLI covers a range of genres of spoken and written text. **Training Params —** batch-size = 16, optimizer = Adam, learning rate = 2e−5, linear learning rate warm-up = 10% of the training data. SBERT outperformed previous methods of benchmarks like STS benchmark (STSb), Argument Facet Similarity (AFS) corpus and Wikipedia dataset from Dor et al.

Okay but how do I use it?

While inference we can use it to either

i) Get the sentence embedding of a given sentence to be used for some downstream task

ii) Get similarity score between 2 sentences using cosine similarity

```
#Install the library
! pip install -U sentence-transformers
#Import the library
from sentence_transformers import SentenceTransformer, util, InputExample, losse
#Load the model(here we use minilm)
model = SentenceTransformer('all-MiniLM-L6-v2')
#We get the embeddings by calling model.encode()
emb1 = model.encode("This is a red cat with a hat.")
emb2 = model.encode("Have you seen my red cat?")
#Get the cosine similarity score between sentences
cos_sim = util.cos_sim(emb1, emb2)
print("Cosine-Similarity:", cos_sim)
```

Okay but how do I finetune it for my usecase?

There are various ways we can finetune it on our dataset to learn semantic similarity (using different ways of inputing dataset and loss function used). The most common 2 ways are CosineSimilarityLoss and TripletLoss (both of which are very similar — For learning through CosineSimilarityLoss we have to pass input data as pairs of sentences with a label between 0 and 1 indicating their similarity. For learning through Triplet loss we use 3 sentences 1 anchor sentence, 1 positive sentence — which is similar to anchor sentence and 1 negtaive sentences — which is dissimilar to anchor sentence).

```
#Using Cosine SimilarityLoss
from torch.utils.data import DataLoader
#Define your train examples. You need more than just two examples...
#Inputs are wrapped around InputExample class which the model expects
train_examples = [InputExample(texts=['My first sentence', 'My second sentence']
    InputExample(texts=['Another pair', 'Unrelated sentence'], label=0.3)]
#Create a PyTorch dataloader and the train loss
train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=16)
train_loss = losses.CosineSimilarityLoss(model)
```

```
#Tune the model
model.fit(train_objectives=[(train_dataloader, train_loss)], epochs=1, warmup_st
```

We only change the format of the train examples and train_loss for training using Triplet loss.

```
#Using Triplet Loss
train_examples = [InputExample(texts=['My first sentence', 'My second sentence',
    InputExample(texts=['My first sentence', 'My second sentence' 'Unrelated sen
train_loss = sentence_transformers.losses.TripletLoss(model=model)
```

Hope this will help you learn about SBERT and put in to use in some real world probem for sentence similarity. Please do provide your feedback in form of responses and claps :)

References :
[1] https://arxiv.org/pdf/1908.10084.pdf
[2] https://www.pinecone.io/learn/series/nlp/sentence-embeddings/

NLP Machine Learning Deep Learning Computer Science

More from the list: "NLP"

Curated by Himanshu Birla

Jon Gi... in Towards Data ...

Characteristics of Word Embeddings

· 11 min read · Sep 4, 2021

Jon Gi... in Towards Data ...

The Word2vec Hyperparameters

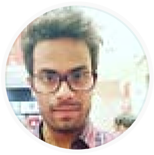
· 6 min read · Sep 3, 2021

Jon Gi... in

The Word2ve

· 15 min rea

View list



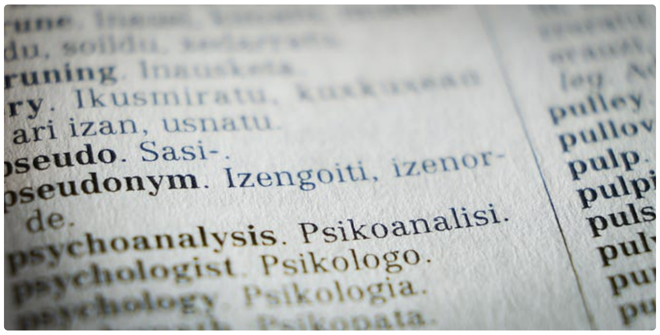
Written by anirban sen

Following

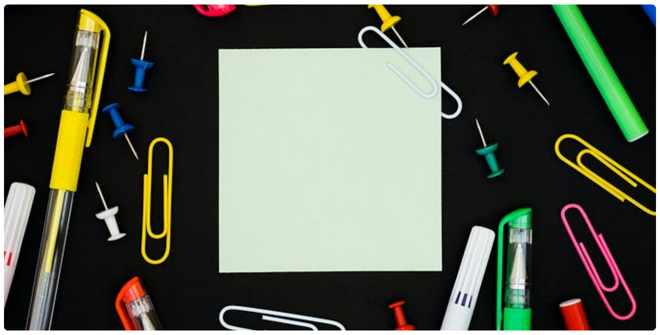
56 Followers

Data Scientist @ Amazon | www.linkedin.com/in/anirban-sen-2709/

More from anirban sen



anirban sen



anirban sen

Text Classification — From Bag-of-Words to BERT

What is Text Classification?

9 min read · Dec 30, 2020

77

Beginner's guide to one of the best Vision model — CLIP (Contrastive...

What is CLIP? Contrastive Language-Image Pre-training (CLIP for short) is a state-of-the...

6 min read · Aug 19

2 1

anirban sen

Finetuning LLMs using LoRA

Before getting to the meat of the blog i.e. Finetuning an LLM, it will be good to have a...

8 min read · 4 days ago

2

anirban sen

LangChain: The Future of LLM-powered Applications — Part...

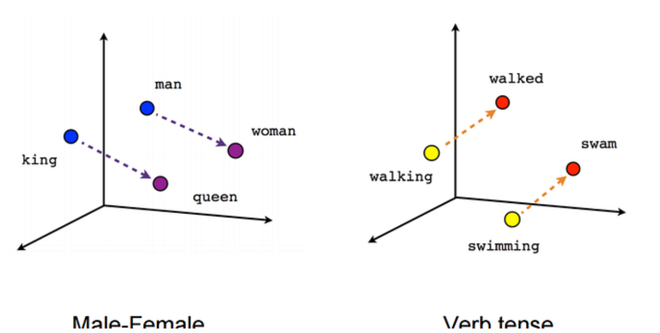
In the previous blog, we learnt about the background/basics of Langchain, In this one...

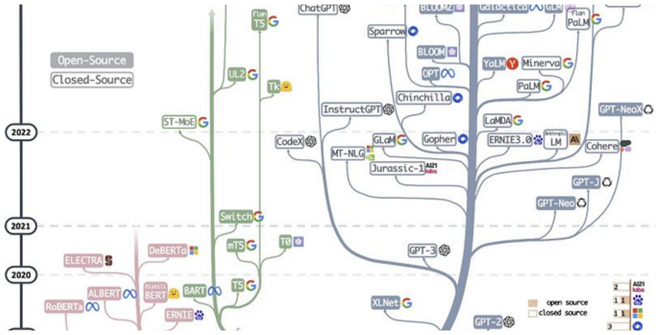
6 min read · Sep 16

1

See all from anirban sen

Recommended from Medium





Maninder Singh

Accelerate Your Text Data Analysis with Custom BERT Word…

One thing is for sure the way humans interact with each other naturally is one of the most…

4 min read · Apr 24

156

Haifeng Li

A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's…

15 min read · Sep 14

372

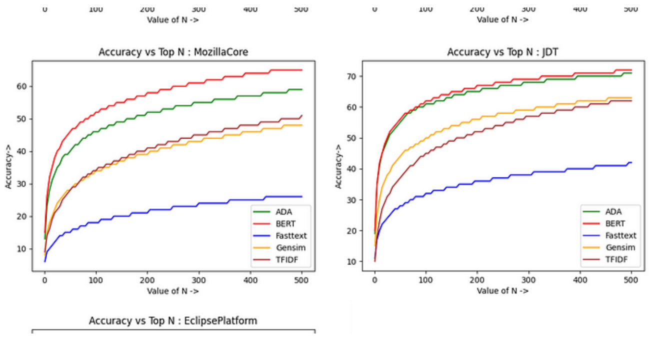
Lists

Natural Language Processing
669 stories · 283 saves

Predictive Modeling w/ Python
20 stories · 452 saves

Practical Guides to Machine Learning
10 stories · 519 saves

The New Chatbots: ChatGPT, Bard, and Beyond
13 stories · 133 saves



Alyx

Semantic Search with FAISS

HuggingFace get_neareast_example and Cosine Similarity Search

9 min read · Jul 15

71 1

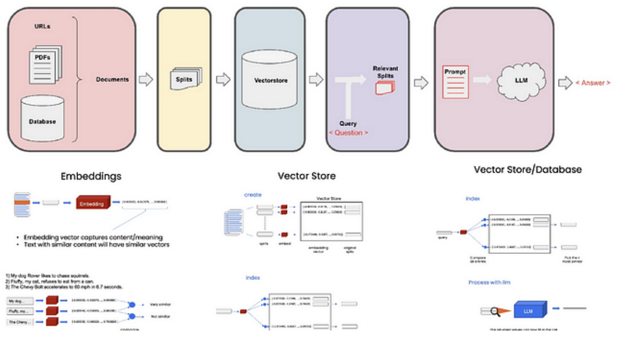
Avinash Patil

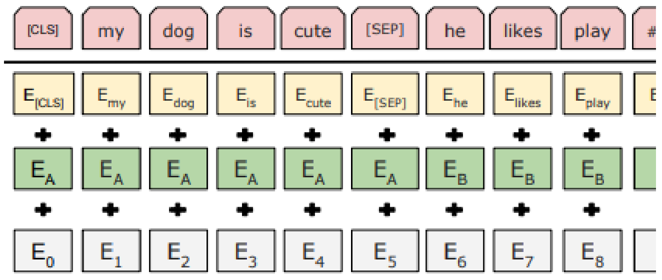
Embeddings: BERT better than ChatGPT4?

In this study, we compared the effectiveness of semantic textual similarity methods for...

4 min read · Sep 19

3 1





[CLS]
my
dog
is
cute
[SEP]
he
likes
play
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play}
E_A E_A E_A E_A E_A E_A E_B E_B E_B
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8

TeeTracker

Chat with your PDF (Streamlit Demo)

Conversation with specific files

4 min read · Sep 15

56

Zain ul Abideen

A Comparative Analysis of LLMs like BERT, BART, and T5

Exploring Language Models

6 min read · Jun 26

20 1

See more recommendations