Technology Review

Transformer Document Embedding
CS410 - UIUC - fnu Kishan Borule Rahul - UIN: *****6641

Transformer

Transformer is a prominent deep learning model that has been widely adopted in various fields, such as natural language processing (NLP), computer vision (CV) and speech processing. Transformer was originally proposed as a sequence-to-sequence model for machine translation. Later works show that Transformer-based pre-trained models (PTMs) can achieve state-of-the-art performances on various tasks. As a consequence, Transformer has become the go-to architecture in NLP, especially for PTMs.

Due to the success, a variety of Transformer variants (a.k.a. X-formers) have been proposed over the past few years. These X-formers improve the vanilla Transformer from different perspectives.

- **Model Efficiency**: A key challenge of applying a Transformer is its inefficiency at processing long sequences mainly due to the computation and memory complexity of the self-attention module.
- **Model Generalization:** Since the transformer is a flexible architecture and makes few as- sumptions on the structural bias of input data, it is hard to train on small-scale data.
- **Model Adaptation:** This line of work aims to adapt the Transformer to specific downstream tasks and applications.

Sentence Transformers [Link]

Framework of sentence transformers provides an easy method to compute dense vector representations for sentences and paragraphs of documents. The models are based on transformer networks like BERT / RoBERTa / XLM-RoBERTa etc. (X-formers) and achieve state-of-the-art performance in various tasks. Text is embedded in vector space such that similar text is close and can efficiently be found using cosine similarity.

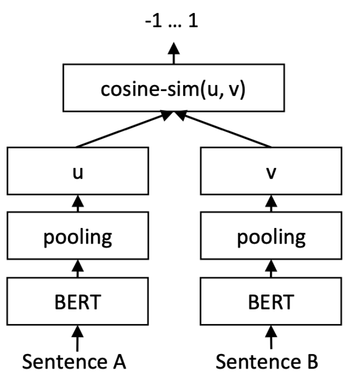
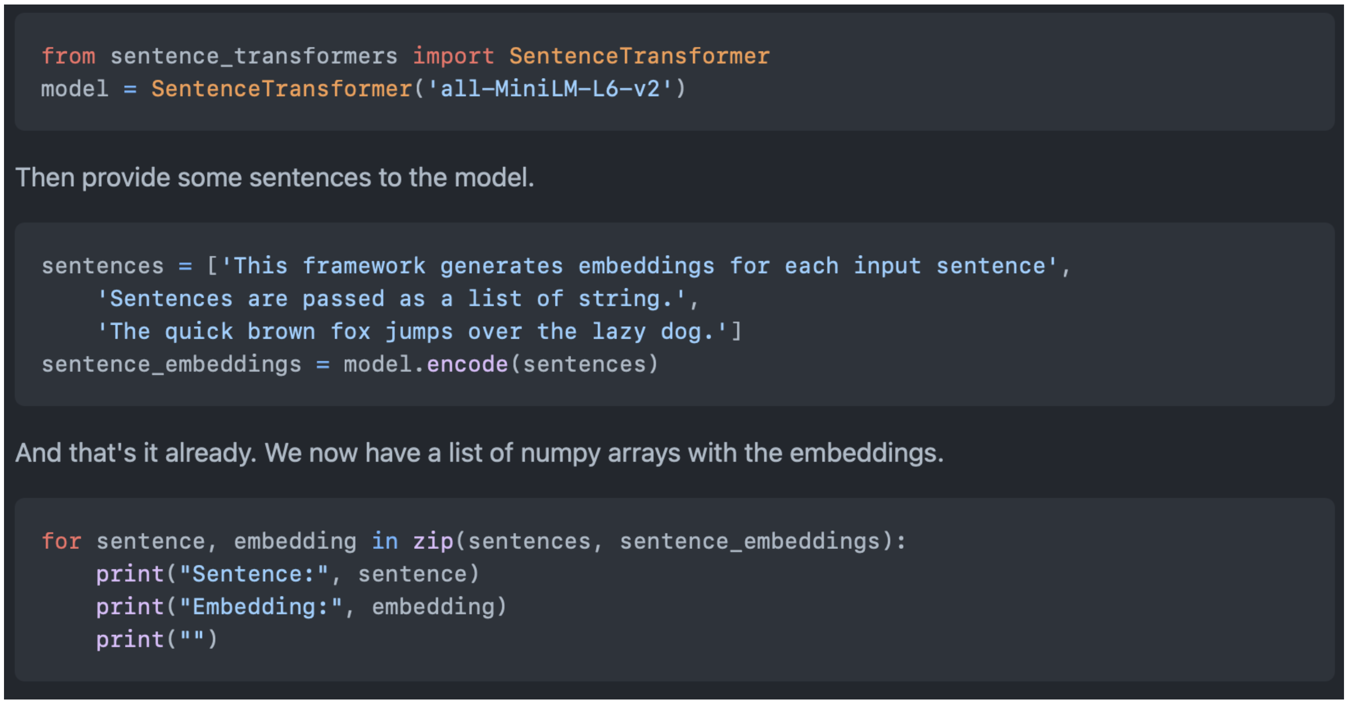


Fig1: Pooling and Similarity Operation

Sentence transformer Python API - How to Use:



```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')
```

Then provide some sentences to the model.

```
sentences = ['This framework generates embeddings for each input sentence',
    'Sentences are passed as a list of string.',
    'The quick brown fox jumps over the lazy dog.']
sentence_embeddings = model.encode(sentences)
```

And that's it already. We now have a list of numpy arrays with the embeddings.

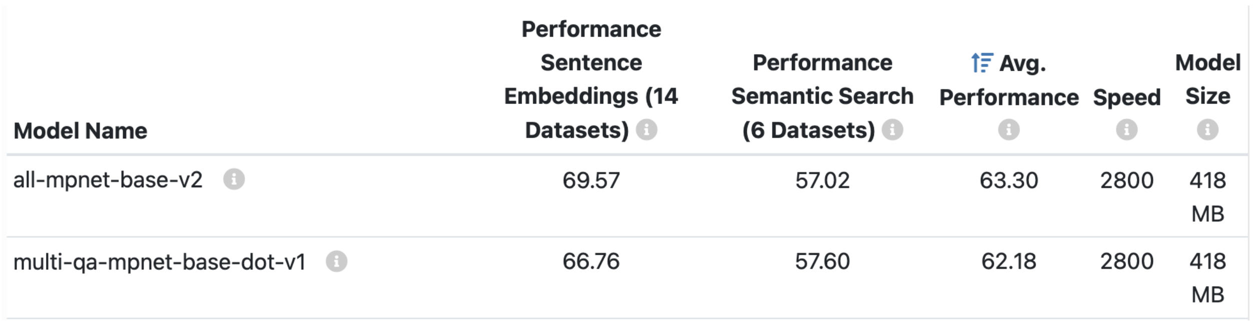
```
for sentence, embedding in zip(sentences, sentence_embeddings):
    print("Sentence:", sentence)
    print("Embedding:", embedding)
    print("")
```

Modelling Strategy:

Sentence transformer adds a pooling operation on top of a transformer model (like BERT/RoBerta - X-formers) to derive a fixed size sentence embedding which can be used for document representation and ranking for the downstream models.

Sentence transformer aims to adapt the transformer architecture by using siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity.

Performance Metrics:



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	418 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	418 MB

Fig2: Top performing model based on accuracy

References:

- Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
- Sentence Transformer Toolkit
- A Survey of Transformers