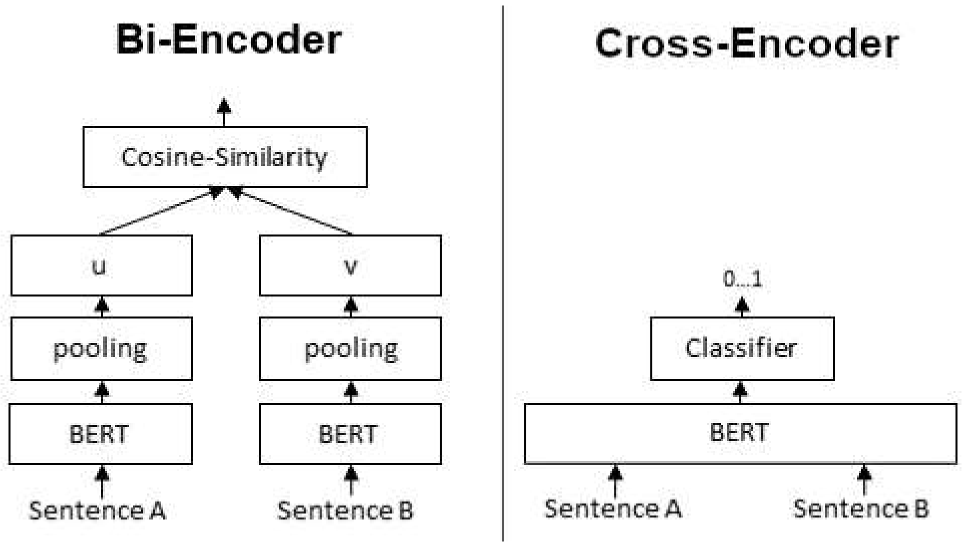
Cross-Encoders

SentenceTransformers also supports to load Cross-Encoders for sentence pair scoring and sentence pair classification tasks.

Bi-Encoder vs. Cross-Encoder

First, it is important to understand the difference between Bi- and Cross-Encoder.

Bi-Encoders produce for a given sentence a sentence embedding. We pass to a BERT independently the sentences A and B, which result in the sentence embeddings u and v. These sentence embedding can then be compared using cosine similarity:



In contrast, for a **Cross-Encoder**, we pass both sentences simultaneously to the Transformer network. It produces then an output value between 0 and 1 indicating the similarity of the input sentence pair:

A **Cross-Encoder does not produce a sentence embedding**. Also, we are not able to pass individual sentences to a Cross-Encoder.

As detailed in our paper, Cross-Encoder achieve better performances than Bi-Encoders. However, for many application they are not practical as they do not produce embeddings we could e.g. index or efficiently compare using cosine similarity.

When to use Cross- / Bi-Encoders?

Cross-Encoders can be used whenever you have a pre-defined set of sentence pairs you want to score. For example, you have 100 sentence pairs and you want to get similarity scores for these 100 pairs.

Bi-Encoders (see Computing Sentence Embeddings) are used whenever you need a sentence embedding in a vector space for efficient comparison. Applications are for example Information Retrieval / Semantic Search or Clustering. Cross-Encoders would be the wrong choice for these application: Clustering 10,000 sentence with CrossEncoders would require computing similarity scores for about 50 Million sentence combinations, which takes about 65 hours. With a Bi-Encoder, you compute the embedding for each sentence, which takes only 5 seconds. You can then perform the clustering.

Cross-Encoders Usage

Using Cross-Encoders is quite easy:

```
from sentence_transformers.cross_encoder import CrossEncoder

model = CrossEncoder("model_name_or_path")
scores = model.predict([["My first", "sentence pair"], ["Second text", "pair"]])
```

You pass to `model.predict` a list of sentence **pairs**. Note, Cross-Encoder do not work on individual sentence, you have to pass sentence pairs.

As model name, you can pass any model or path that is compatible with Hugging Face AutoModel class

For a full example, to score a query with all possible sentences in a corpus see cross-encoder_usage.py.

Combining Bi- and Cross-Encoders

Cross-Encoder achieve higher performance than Bi-Encoders, however, they do not scale well for large datasets. Here, it can make sense to combine Cross- and Bi-Encoders, for example in Information Retrieval / Semantic Search scenarios: First, you use an efficient Bi-Encoder to retrieve e.g. the top-100 most similar sentences for a query. Then, you use a Cross-Encoder to re-rank these 100 hits by computing the score for every (query, hit) combination.

For more details on combing Bi- and Cross-Encoders, see Application - Information Retrieval.