[![Logo](../../../_static/logo.png)](../../../index.html)

Getting Started

* [Installation](../../../docs/installation.html)
  * [Install with pip](../../../docs/installation.html#install-with-pip)
  * [Install with Conda](../../../docs/installation.html#install-with-conda)
  * [Install from Source](../../../docs/installation.html#install-from-source)
  * [Editable Install](../../../docs/installation.html#editable-install)
  * [Install PyTorch with CUDA support](../../../docs/installation.html#install-pytorch-with-cuda-support)
* [Quickstart](../../../docs/quickstart.html)
  * [Sentence Transformer](../../../docs/quickstart.html#sentence-transformer)
  * [Cross Encoder](../../../docs/quickstart.html#cross-encoder)
  * [Next Steps](../../../docs/quickstart.html#next-steps)

Sentence Transformer

* [Usage](../../../docs/sentence_transformer/usage/usage.html)
  * [Computing Embeddings](../computing-embeddings/README.html)
    * [Initializing a Sentence Transformer Model](../computing-embeddings/README.html#initializing-a-sentence-transformer-model)
    * [Calculating Embeddings](../computing-embeddings/README.html#calculating-embeddings)
    * [Prompt Templates](../computing-embeddings/README.html#prompt-templates)
    * [Input Sequence Length](../computing-embeddings/README.html#id1)
    * [Multi-Process / Multi-GPU Encoding](../computing-embeddings/README.html#multi-process-multi-gpu-encoding)

* [Semantic Textual Similarity](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html)
* [Similarity Calculation](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../semantic-search/README.html)
* [Background](../semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../semantic-search/README.html#speed-optimization)
* [Elasticsearch](../semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../semantic-search/README.html#retrieve-re-rank)
* [Examples](../semantic-search/README.html#examples)
* Retrieve & Re-Rank
* Retrieve & Re-Rank Pipeline
* Retrieval: Bi-Encoder
* Re-Ranker: Cross-Encoder
* Example Scripts
* Pre-trained Bi-Encoders (Retrieval)
* Pre-trained Cross-Encoders (Re-Ranker)
* [Clustering](../clustering/README.html)
* [k-Means](../clustering/README.html#k-means)
* [Agglomerative Clustering](../clustering/README.html#agglomerative-clustering)

    * [Fast Clustering](../clustering/README.html#fast-clustering)
    * [Topic Modeling](../clustering/README.html#topic-modeling)
  * [Paraphrase Mining](../paraphrase-mining/README.html)
    * [`paraphrase_mining()`](../paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
  * [Translated Sentence Mining](../parallel-sentence-mining/README.html)
    * [Margin Based Mining](../parallel-sentence-mining/README.html#margin-based-mining)
    * [Examples](../parallel-sentence-mining/README.html#examples)
  * [Image Search](../image-search/README.html)
    * [Installation](../image-search/README.html#installation)
    * [Usage](../image-search/README.html#usage)
    * [Examples](../image-search/README.html#examples)
  * [Embedding Quantization](../embedding-quantization/README.html)
    * [Binary Quantization](../embedding-quantization/README.html#binary-quantization)
    * [Scalar (int8) Quantization](../embedding-quantization/README.html#scalar-int8-quantization)
    * [Additional extensions](../embedding-quantization/README.html#additional-extensions)
    * [Demo](../embedding-quantization/README.html#demo)
    * [Try it yourself](../embedding-quantization/README.html#try-it-yourself)
  * [Speeding up Inference](../../../docs/sentence_transformer/usage/efficiency.html)
    * [PyTorch](../../../docs/sentence_transformer/usage/efficiency.html#pytorch)
    * [ONNX](../../../docs/sentence_transformer/usage/efficiency.html#onnx)
    * [OpenVINO](../../../docs/sentence_transformer/usage/efficiency.html#openvino)
    * [Benchmarks](../../../docs/sentence_transformer/usage/efficiency.html#benchmarks)
  * [Creating Custom Models](../../../docs/sentence_transformer/usage/custom_models.html)
    * [Structure of Sentence Transformer Models](../../../docs/sentence_transformer/usage/custom_models.html#structure-of-sentence-transfo

rmer-models)
* [Sentence Transformer Model from a Transformers Model](../../../docs/sentence_transformer/usage/custom_models.html#sentence-transformer-model-from-a-transformers-model)
* [Pretrained Models](../../../docs/sentence_transformer/pretrained_models.html)
* [Original Models](../../../docs/sentence_transformer/pretrained_models.html#original-models)
* [Semantic Search Models](../../../docs/sentence_transformer/pretrained_models.html#semantic-search-models)
* [Multi-QA Models](../../../docs/sentence_transformer/pretrained_models.html#multi-qa-models)
* [MSMARCO Passage Models](../../../docs/sentence_transformer/pretrained_models.html#msmarco-passage-models)
* [Multilingual Models](../../../docs/sentence_transformer/pretrained_models.html#multilingual-models)
* [Semantic Similarity Models](../../../docs/sentence_transformer/pretrained_models.html#semantic-similarity-models)
* [Bitext Mining](../../../docs/sentence_transformer/pretrained_models.html#bitext-mining)
* [Image & Text-Models](../../../docs/sentence_transformer/pretrained_models.html#image-text-models)
* [INSTRUCTOR models](../../../docs/sentence_transformer/pretrained_models.html#instructor-models)
* [Scientific Similarity Models](../../../docs/sentence_transformer/pretrained_models.html#scientific-similarity-models)
* [Training Overview](../../../docs/sentence_transformer/training_overview.html)
* [Why Finetune?](../../../docs/sentence_transformer/training_overview.html#why-finetune)
* [Training Components](../../../docs/sentence_transformer/training_overview.html#training-components)
* [Dataset](../../../docs/sentence_transformer/training_overview.html#dataset)

* [Dataset Format](../../../docs/sentence_transformer/training_overview.html#dataset-format)
* [Loss Function](../../../docs/sentence_transformer/training_overview.html#loss-function)
* [Training Arguments](../../../docs/sentence_transformer/training_overview.html#training-arguments)
* [Evaluator](../../../docs/sentence_transformer/training_overview.html#evaluator)
* [Trainer](../../../docs/sentence_transformer/training_overview.html#trainer)
* [Callbacks](../../../docs/sentence_transformer/training_overview.html#callbacks)
* [Multi-Dataset Training](../../../docs/sentence_transformer/training_overview.html#multi-dataset-training)
* [Deprecated Training](../../../docs/sentence_transformer/training_overview.html#deprecated-training)
* [Best Base Embedding Models](../../../docs/sentence_transformer/training_overview.html#best-base-embedding-models)
* [Dataset Overview](../../../docs/sentence_transformer/dataset_overview.html)
* [Datasets on the Hugging Face Hub](../../../docs/sentence_transformer/dataset_overview.html#datasets-on-the-hugging-face-hub)
* [Pre-existing Datasets](../../../docs/sentence_transformer/dataset_overview.html#pre-existing-datasets)
* [Loss Overview](../../../docs/sentence_transformer/loss_overview.html)
* [Loss modifiers](../../../docs/sentence_transformer/loss_overview.html#loss-modifiers)
* [Distillation](../../../docs/sentence_transformer/loss_overview.html#distillation)
* [Commonly used Loss Functions](../../../docs/sentence_transformer/loss_overview.html#commonly-used-loss-functions)
* [Custom Loss Functions](../../../docs/sentence_transformer/loss_overview.html#custom-loss-functions)
* [Training Examples](../../../docs/sentence_transformer/training/examples.html)
* [Semantic Textual Similarity](../../training/sts/README.html)

    * [Training data](../../training/sts/README.html#training-data)
    * [Loss Function](../../training/sts/README.html#loss-function)
  * [Natural Language Inference](../../training/nli/README.html)
    * [Data](../../training/nli/README.html#data)
    * [SoftmaxLoss](../../training/nli/README.html#softmaxloss)
    * [MultipleNegativesRankingLoss](../../training/nli/README.html#multiplenegativesrankingloss)
  * [Paraphrase Data](../../training/paraphrases/README.html)
    * [Pre-Trained Models](../../training/paraphrases/README.html#pre-trained-models)
  * [Quora Duplicate Questions](../../training/quora_duplicate_questions/README.html)
    * [Training](../../training/quora_duplicate_questions/README.html#training)
    * [MultipleNegativesRankingLoss](../../training/quora_duplicate_questions/README.html#multiplenegativesrankingloss)
    * [Pretrained Models](../../training/quora_duplicate_questions/README.html#pretrained-models)
  * [MS MARCO](../../training/ms_marco/README.html)
    * [Bi-Encoder](../../training/ms_marco/README.html#bi-encoder)
  * [Matryoshka Embeddings](../../training/matryoshka/README.html)
    * [Use Cases](../../training/matryoshka/README.html#use-cases)
    * [Results](../../training/matryoshka/README.html#results)
    * [Training](../../training/matryoshka/README.html#training)
    * [Inference](../../training/matryoshka/README.html#inference)
    * [Code Examples](../../training/matryoshka/README.html#code-examples)
  * [Adaptive Layers](../../training/adaptive_layer/README.html)
    * [Use Cases](../../training/adaptive_layer/README.html#use-cases)
    * [Results](../../training/adaptive_layer/README.html#results)
    * [Training](../../training/adaptive_layer/README.html#training)
    * [Inference](../../training/adaptive_layer/README.html#inference)

sentence-pairs)](../../training/data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)
    * [Training](../../training/data_augmentation/README.html#training)
    * [Citation](../../training/data_augmentation/README.html#citation)
  * [Training with Prompts](../../training/prompts/README.html)
    * [What are Prompts?](../../training/prompts/README.html#what-are-prompts)
    * [Why would we train with Prompts?](../../training/prompts/README.html#why-would-we-train-with-prompts)
    * [How do we train with Prompts?](../../training/prompts/README.html#how-do-we-train-with-prompts)
  * [Training with PEFT Adapters](../../training/peft/README.html)
    * [Compatibility Methods](../../training/peft/README.html#compatibility-methods)
    * [Adding a New Adapter](../../training/peft/README.html#adding-a-new-adapter)
    * [Loading a Pretrained Adapter](../../training/peft/README.html#loading-a-pretrained-adapter)
    * [Training Script](../../training/peft/README.html#training-script)
  * [Unsupervised Learning](../../unsupervised_learning/README.html)
    * [TSDAE](../../unsupervised_learning/README.html#tsdae)
    * [SimCSE](../../unsupervised_learning/README.html#simcse)
    * [CT](../../unsupervised_learning/README.html#ct)
    * [CT (In-Batch Negative Sampling)](../../unsupervised_learning/README.html#ct-in-batch-negative-sampling)
    * [Masked Language Model (MLM)](../../unsupervised_learning/README.html#masked-language-model-mlm)
    * [GenQ](../../unsupervised_learning/README.html#genq)
    * [GPL](../../unsupervised_learning/README.html#gpl)
    * [Performance Comparison](../../unsupervised_learning/README.html#performance-comparison)

Cross Encoder

    * [STSbenchmark](../../../docs/cross_encoder/pretrained_models.html#stsbenchmark)
    * [Quora Duplicate Questions](../../../docs/cross_encoder/pretrained_models.html#quora-duplicate-questions)
    * [NLI](../../../docs/cross_encoder/pretrained_models.html#nli)
    * [Community Models](../../../docs/cross_encoder/pretrained_models.html#community-models)
  * [Training Overview](../../../docs/cross_encoder/training_overview.html)
  * [Training Examples](../../../docs/cross_encoder/training/examples.html)
    * [MS MARCO](../../training/ms_marco/cross_encoder_README.html)
      * [Cross-Encoder](../../training/ms_marco/cross_encoder_README.html#cross-encoder)
      * [Cross-Encoder Knowledge Distillation](../../training/ms_marco/cross_encoder_README.html#cross-encoder-knowledge-distillation)

Package Reference

  * [Sentence Transformer](../../../docs/package_reference/sentence_transformer/index.html)
    * [SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html)
      * [SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#id1)
      * [SentenceTransformerModelCardData](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentencetransformermodelcarddata)
      * [SimilarityFunction](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html

#similarityfunction)
    * [Trainer](../../../docs/package_reference/sentence_transformer/trainer.html)
        * [SentenceTransformerTrainer](../../../docs/package_reference/sentence_transformer/trainer.html#sentencetransformertrainer)
    * [Training Arguments](../../../docs/package_reference/sentence_transformer/training_args.html)
        * [SentenceTransformerTrainingArguments](../../../docs/package_reference/sentence_transformer/training_args.html#sentencetransformertrainingarguments)
    * [Losses](../../../docs/package_reference/sentence_transformer/losses.html)
        * [BatchAllTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchalltripletloss)
        * [BatchHardSoftMarginTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardsoftmargintripletloss)
        * [BatchHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardtripletloss)
        * [BatchSemiHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchsemihardtripletloss)
        * [ContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastiveloss)
        * [OnlineContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#onlinecontrastiveloss)

*
[ContrastiveTensionLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionloss)
*
[ContrastiveTensionLossInBatchNegatives](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionlossinbatchnegatives)
    * [CoSENTLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosentloss)
    * [AnglELoss](../../../docs/package_reference/sentence_transformer/losses.html#angleloss)
*
[CosineSimilarityLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosinesimilarityloss)
*
[DenoisingAutoEncoderLoss](../../../docs/package_reference/sentence_transformer/losses.html#denoisingautoencoderloss)
*
[GISTEmbedLoss](../../../docs/package_reference/sentence_transformer/losses.html#gistembedloss)
*
[CachedGISTEmbedLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedgistembedloss)
    * [MSELoss](../../../docs/package_reference/sentence_transformer/losses.html#mseloss)
*
[MarginMSELoss](../../../docs/package_reference/sentence_transformer/losses.html#marginmseloss)
*
[MatryoshkaLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshkaloss)

*
[Matryoshka2dLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshka2dloss)
*
[AdaptiveLayerLoss](../../../docs/package_reference/sentence_transformer/losses.html#adaptivelayerloss)
*
[MegaBatchMarginLoss](../../../docs/package_reference/sentence_transformer/losses.html#megabatchmarginloss)
*
[MultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#multiplenegativesrankingloss)
*
[CachedMultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativesrankingloss)
*
[MultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#multiplenegativessymmetricrankingloss)
*
[CachedMultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativessymmetricrankingloss)
    * [SoftmaxLoss](../../../docs/package_reference/sentence_transformer/losses.html#softmaxloss)
    * [TripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#tripletloss)
  * [Samplers](../../../docs/package_reference/sentence_transformer/sampler.html)
*
[BatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#batchsamplers)

*
[MultiDatasetBatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#multidatasetbatchsamplers)
    * [Evaluation](../../../docs/package_reference/sentence_transformer/evaluation.html)
*
[BinaryClassificationEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#binaryclassificationevaluator)
*
[EmbeddingSimilarityEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#embeddingsimilarityevaluator)
*
[InformationRetrievalEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#informationretrievalevaluator)
*
[NanoBEIREvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#nanobeirevaluator)
*
[MSEEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#mseevaluator)
*
[ParaphraseMiningEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#paraphraseminingevaluator)
*
[RerankingEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#rerankingevaluator)
*
[SentenceEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#sentenc

eevaluator)

*

[SequentialEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#sequentialevaluator)

*

[TranslationEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#translationevaluator)

*

[TripletEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#tripletevaluator)

* [Datasets](../../../docs/package_reference/sentence_transformer/datasets.html)

*

[ParallelSentencesDataset](../../../docs/package_reference/sentence_transformer/datasets.html#parallelsentencesdataset)

*

[SentenceLabelDataset](../../../docs/package_reference/sentence_transformer/datasets.html#sentencelabeldataset)

*

[DenoisingAutoEncoderDataset](../../../docs/package_reference/sentence_transformer/datasets.html#denoisingautoencoderdataset)

*

[NoDuplicatesDataLoader](../../../docs/package_reference/sentence_transformer/datasets.html#noduplicatesdataloader)

* [Models](../../../docs/package_reference/sentence_transformer/models.html)

* [Main Classes](../../../docs/package_reference/sentence_transformer/models.html#main-classes)

* [Further

Classes](../../../docs/package_reference/sentence_transformer/models.html#further-classes)
    * [quantization](../../../docs/package_reference/sentence_transformer/quantization.html)
*
[`quantize_embeddings()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.quantize_embeddings)
*
[`semantic_search_faiss()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_faiss)
*
[`semantic_search_usearch()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_usearch)
  * [Cross Encoder](../../../docs/package_reference/cross_encoder/index.html)
    * [CrossEncoder](../../../docs/package_reference/cross_encoder/cross_encoder.html)
      * [CrossEncoder](../../../docs/package_reference/cross_encoder/cross_encoder.html#id1)
* [Training Inputs](../../../docs/package_reference/cross_encoder/cross_encoder.html#training-inputs)
    * [Evaluation](../../../docs/package_reference/cross_encoder/evaluation.html)
*
[CEBinaryAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryaccuracyevaluator)
*
[CEBinaryClassificationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryclassificationevaluator)
*
[CECorrelationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cecorrelationevaluator)
      * [CEF1Evaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cef1evaluator)

*
[CESoftmaxAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cesoftmaxaccuracyevaluator)
*
[CERerankingEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cererankingevaluator)
  * [util](../../../docs/package_reference/util.html)
    * [Helper Functions](../../../docs/package_reference/util.html#module-sentence_transformers.util)
*
[`community_detection()`](../../../docs/package_reference/util.html#sentence_transformers.util.community_detection)
      * [`http_get()`](../../../docs/package_reference/util.html#sentence_transformers.util.http_get)
*
[`is_training_available()`](../../../docs/package_reference/util.html#sentence_transformers.util.is_training_available)
*
[`mine_hard_negatives()`](../../../docs/package_reference/util.html#sentence_transformers.util.mine_hard_negatives)
*
[`normalize_embeddings()`](../../../docs/package_reference/util.html#sentence_transformers.util.normalize_embeddings)
*
[`paraphrase_mining()`](../../../docs/package_reference/util.html#sentence_transformers.util.paraphrase_mining)
*
[`semantic_search()`](../../../docs/package_reference/util.html#sentence_transformers.util.semantic_search)

*
[`truncate_embeddings()`](../../../docs/package_reference/util.html#sentence_transformers.util.truncate_embeddings)
* [Model Optimization](../../../docs/package_reference/util.html#module-sentence_transformers.backend)
*
[`export_dynamic_quantized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)
*
[`export_optimized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model)
*
[`export_static_quantized_openvino_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_static_quantized_openvino_model)
* [Similarity Metrics](../../../docs/package_reference/util.html#module-sentence_transformers.util)
* [`cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.cos_sim)
* [`dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.dot_score)
*
[`euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.euclidean_sim)
*
[`manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.manhattan_sim)
*
[`pairwise_cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_cos_sim)
*

[`pairwise_dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_dot_score)

*

[`pairwise_euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_euclidean_sim)

*

[`pairwise_manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_manhattan_sim)

__[Sentence Transformers](../../../index.html)

  * [](../../../index.html)
  * [Usage](../../../docs/sentence_transformer/usage/usage.html)
  * Retrieve & Re-Rank

* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/examples/applications/retrieve_rerank/README.md)

* * *

Retrieve & Re-Rankïƒ•

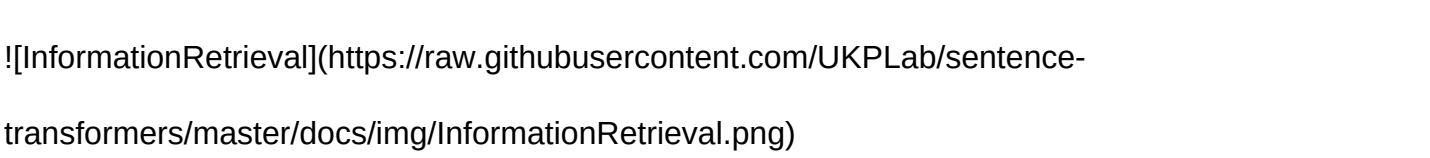
In [Semantic Search](../semantic-search/README.html) we have shown how to use SentenceTransformer to compute embeddings for queries, sentences, and paragraphs and how to use this for semantic search.

For complex search tasks, for example question answering retrieval, the search

can significantly be improved by using **Retrieve & Re-Rank**.

Retrieve & Re-Rank Pipelineïƒ•

The following pipeline for Information Retrieval / Question Answering Retrieval works very well. All components are provided and explained in this article:

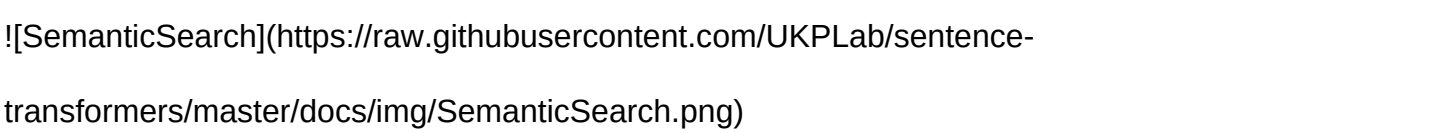


Given a search query, we first use a **retrieval system** that retrieves a large list of e.g. 100 possible hits which are potentially relevant for the query. For the retrieval, we can use either lexical search, e.g. with a vector engine like Elasticsearch, or we can use dense retrieval with a bi-encoder. However, the retrieval system might retrieve documents that are not that relevant for the search query. Hence, in a second stage, we use a **re-ranker** based on a **cross-encoder** that scores the relevancy of all candidates for the given search query. The output will be a ranked list of hits we can present to the user.

Retrieval: Bi-Encoderïƒ•

For the retrieval of the candidate set, we can either use lexical search (e.g. [Elasticsearch](https://www.elastic.co/elasticsearch/)), or we can use a bi-encoder which is implemented in Sentence Transformers.

Lexical search looks for literal matches of the query words in your document collection. It will not recognize synonyms, acronyms or spelling variations.
In contrast, semantic search (or dense retrieval) encodes the search query into vector space and retrieves the document embeddings that are close in vector space.



Semantic search overcomes the shortcomings of lexical search and can recognize synonym and acronyms. Have a look at the [semantic search article](../semantic-search/README.html) for different options to implement semantic search.

Re-Ranker: Cross-Encoderï ƒ•

The retriever has to be efficient for large document collections with millions of entries. However, it might return irrelevant candidates. A re-ranker based on a Cross-Encoder can substantially improve the final results for the user.
The query and a possible document is passed simultaneously to transformer network, which then outputs a single score between 0 and 1 indicating how relevant the document is for the given query.

The advantage of Cross-Encoders is the higher performance, as they perform

attention across the query and the document. Scoring thousands or millions of (query, document)-pairs would be rather slow. Hence, we use the retriever to create a set of e.g. 100 possible candidates which are then re-ranked by the Cross-Encoder.

Example Scriptsïƒ•

*

[retrieve_rerank_simple_wikipedia.ipynb](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/retrieve_rerank/retrieve_rerank_simple_wikipedia.ipynb) [[Colab Version](https://colab.research.google.com/github/UKPLab/sentence-transformers/blob/master/examples/applications/retrieve_rerank/retrieve_rerank_simple_wikipedia.ipynb)]: This script uses the smaller [Simple English Wikipedia](https://simple.wikipedia.org/wiki/Main_Page) as document collection to provide answers to user questions / search queries. First, we split all Wikipedia articles into paragraphs and encode them with a bi-encoder. If a new query / question is entered, it is encoded by the same bi-encoder and the paragraphs with the highest cosine-similarity are retrieved (see [semantic search](../semantic-search/README.html)). Next, the retrieved candidates are scored by a Cross-Encoder re-ranker and the 5 passages with the highest score from the Cross-Encoder are presented to the user.

*

[in_document_search_crossencoder.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/retrieve_rerank/in_document_search_crossencoder.py): If you only have a small set of paragraphs, we donâ€™t do the retrieval stage. This is for example the case if you want to perform search within a single document. In this example, we take the Wikipedia article about Europe and split it into paragraphs. Then, the search query / question and all paragraphs are scored using the Cross-Encoder re-ranker. The most relevant passages for the query are returned.

Pre-trained Bi-Encoders (Retrieval)ïƒ•

The bi-encoder produces embeddings independently for your paragraphs and for your search queries. You can use it like this:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("multi-qa-mpnet-base-dot-v1")

docs = [
    "My first paragraph. That contains information",
    "Python is a programming language.",
]
document_embeddings = model.encode(docs)

query = "What is Python?"
query_embedding = model.encode(query)
```

For more details how to compare the embeddings, see [semantic search](../semantic-search/README.html).

We provide pre-trained models based on:

* **MS MARCO:** 500k real user queries from Bing search engine. See [MS MARCO models](../../../docs/pretrained-models/msmarco-v3.html)

Pre-trained Cross-Encoders (Re-Ranker)ïƒ•

For pre-trained Cross Encoder models, see: [MS MARCO Cross-Encoders](../../../docs/pretrained-models/ce-msmarco.html)

[Previous](../semantic-search/README.html "Semantic Search") [Next](../clustering/README.html "Clustering")

* * *