[![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_rerank/README.html)
* [Retrieve & Re-Rank Pipeline](../retrieve_rerank/README.html#retrieve-re-rank-pipeline)
* [Retrieval: Bi-Encoder](../retrieve_rerank/README.html#retrieval-bi-encoder)
* [Re-Ranker: Cross-Encoder](../retrieve_rerank/README.html#re-ranker-cross-encoder)
* [Example Scripts](../retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../retrieve_rerank/README.html#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()`
  * [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)

* [Domain Adaptation](../../domain_adaptation/README.html)
* [Domain Adaptation vs. Unsupervised Learning](../../domain_adaptation/README.html#domain-adaptation-vs-unsupervised-learning)
* [Adaptive Pre-Training](../../domain_adaptation/README.html#adaptive-pre-training)
* [GPL: Generative Pseudo-Labeling](../../domain_adaptation/README.html#gpl-generative-pseudo-labeling)
* [Hyperparameter Optimization](../../training/hpo/README.html)
* [HPO Components](../../training/hpo/README.html#hpo-components)
* [Putting It All Together](../../training/hpo/README.html#putting-it-all-together)
* [Example Scripts](../../training/hpo/README.html#example-scripts)
* [Distributed Training](../../../docs/sentence_transformer/training/distributed.html)
* [Comparison](../../../docs/sentence_transformer/training/distributed.html#comparison)
* [FSDP](../../../docs/sentence_transformer/training/distributed.html#fsdp)

Cross Encoder

* [Usage](../../../docs/cross_encoder/usage/usage.html)
* [Retrieve & Re-Rank](../retrieve_rerank/README.html)
* [Retrieve & Re-Rank Pipeline](../retrieve_rerank/README.html#retrieve-re-rank-pipeline)
* [Retrieval: Bi-Encoder](../retrieve_rerank/README.html#retrieval-bi-encoder)
* [Re-Ranker: Cross-Encoder](../retrieve_rerank/README.html#re-ranker-cross-encoder)
* [Example Scripts](../retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
* [Pretrained Models](../../../docs/cross_encoder/pretrained_models.html)

    * [MS MARCO](../../../docs/cross_encoder/pretrained_models.html#ms-marco)
    * [SQuAD (QNLI)](../../../docs/cross_encoder/pretrained_models.html#squad-qnli)
    * [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#sentenceevaluator)

*

[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#cecorrelatio

nevaluator)
    * [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)
* Paraphrase Mining

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

* * *

Paraphrase Miningï•

Paraphrase mining is the task of finding paraphrases (texts with identical /
similar meaning) in a large corpus of sentences. In [Semantic Textual
Similarity](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html)

we saw a simplified version of finding paraphrases in a list of sentences. The approach presented there used a brute-force approach to score and rank all pairs.

However, as this has a quadratic runtime, it fails to scale to large (10,000 and more) collections of sentences. For larger collections, the `paraphrase_mining()` function can be used:

```
from sentence_transformers import SentenceTransformer
from sentence_transformers.util import paraphrase_mining

model = SentenceTransformer("all-MiniLM-L6-v2")

# Single list of sentences - Possible tens of thousands of sentences
sentences = [
    "The cat sits outside",
    "A man is playing guitar",
    "I love pasta",
    "The new movie is awesome",
    "The cat plays in the garden",
    "A woman watches TV",
    "The new movie is so great",
    "Do you like pizza?",
]
```

```
    paraphrases = paraphrase_mining(model, sentences)

    for paraphrase in paraphrases[0:10]:
        score, i, j = paraphrase
        print("{} \t\t {} \t\t Score: {:.4f}".format(sentences[i], sentences[j], score))
```

The `paraphrase_mining()` accepts the following parameters:

sentence_transformers.util.paraphrase_mining(_model, sentences: list[str], show_progress_bar: bool = False, batch_size: int = 32, query_chunk_size: int = 5000, corpus_chunk_size: int = 100000, max_pairs: int = 500000, top_k: int = 100, score_function: ~typing.Callable[[~torch.Tensor, ~torch.Tensor], ~torch.Tensor] = <function cos_sim>_) -> list[list[float | int]][[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\util.py#L317-L359)ïƒ•

Given a list of sentences / texts, this function performs paraphrase mining.
It compares all sentences against all other sentences and returns a list with
the pairs that have the highest cosine similarity score.

Parameters:

* **model**
([_SentenceTransformer_](../../../docs/package_reference/sentence_transformer/SentenceTransfor

mer.html#sentence_transformers.SentenceTransformer "sentence_transformers.SentenceTransformer")) â€“ SentenceTransformer model for embedding computation

* **sentences** (_List_ _[__str_ _]_) â€“ A list of strings (texts or sentences)

* **show_progress_bar** (_bool_ _,__optional_) â€“ Plotting of a progress bar. Defaults to False.

* **batch_size** (_int_ _,__optional_) â€“ Number of texts that are encoded simultaneously by the model. Defaults to 32.

* **query_chunk_size** (_int_ _,__optional_) â€“ Search for most similar pairs for #query_chunk_size at the same time. Decrease, to lower memory footprint (increases run-time). Defaults to 5000.

* **corpus_chunk_size** (_int_ _,__optional_) â€“ Compare a sentence simultaneously against #corpus_chunk_size other sentences. Decrease, to lower memory footprint (increases run-time). Defaults to 100000.

* **max_pairs** (_int_ _,__optional_) â€“ Maximal number of text pairs returned. Defaults to 500000.

* **top_k** (_int_ _,__optional_) â€“ For each sentence, we retrieve up to top_k other sentences. Defaults to 100.

* **score_function** (_Callable_ _[__[__Tensor_ _,__Tensor_ _]__,__Tensor_ _]__,__optional_) â€“ Function for computing scores. By default, cosine similarity. Defaults to cos_sim.

Returns:

Returns a list of triplets with the format [score, id1, id2]

Return type:

List[List[Union[float, int]]]

To optimize memory and computation time, paraphrase mining is performed in chunks, as specified by `query_chunk_size` and `corpus_chunk_size`. To be specific, only `query_chunk_size * corpus_chunk_size` pairs will be compared at a time, rather than `len(sentences) * len(sentences)`. This is more time- and memory-efficient. Additionally, `paraphrase_mining()` only considers the `top_k` best scores per sentences per chunk. You can experiment with this value as an efficiency-performance trade-off.

For example, for each sentence you will get only the one most relevant sentence in this script.

```
paraphrases = paraphrase_mining(model, sentences, corpus_chunk_size=len(sentences),
```

top_k=1)

The final key parameter is `max_pairs`, which determines the maximum number of paraphrase pairs that the function returns. Usually, you get fewer pairs returned because the list is cleaned of duplicates, e.g., if it contains (A, B) and (B, A), then only one is returned.

Note

If B is the most similar sentence for A, A is not necessarily the most similar sentence for B. So it can happen that the returned list contains entries like (A, B) and (B, C).

[Previous](../clustering/README.html "Clustering") [Next](../parallel-sentence-mining/README.html "Translated Sentence Mining")

* * *