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

Getting Started

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

Sentence Transformer

* [Usage](../../sentence_transformer/usage/usage.html)
    * [Computing Embeddings](../../../examples/applications/computing-embeddings/README.html)
        * [Initializing a Sentence Transformer Model](../../../examples/applications/computing-embeddings/README.html#initializing-a-sentence-transformer-model)
        * [Calculating Embeddings](../../../examples/applications/computing-embeddings/README.html#calculating-embeddings)
        * [Prompt Templates](../../../examples/applications/computing-embeddings/README.html#prompt-templates)

* [Input Sequence Length](../../../examples/applications/computing-embeddings/README.html#id1)
* [Multi-Process / Multi-GPU Encoding](../../../examples/applications/computing-embeddings/README.html#multi-process-multi-gpu-encoding)
* [Semantic Textual Similarity](../../sentence_transformer/usage/semantic_textual_similarity.html)
* [Similarity Calculation](../../sentence_transformer/usage/semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../../../examples/applications/semantic-search/README.html)
* [Background](../../../examples/applications/semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../../../examples/applications/semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../../../examples/applications/semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../../../examples/applications/semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../../../examples/applications/semantic-search/README.html#speed-optimization)
* [Elasticsearch](../../../examples/applications/semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../../../examples/applications/semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../../../examples/applications/semantic-search/README.html#retrieve-re-rank)

    * [Examples](../../../examples/applications/semantic-search/README.html#examples)
  * [Retrieve & Re-Rank](../../../examples/applications/retrieve_rerank/README.html)
    * [Retrieve & Re-Rank Pipeline](../../../examples/applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)
    * [Retrieval: Bi-Encoder](../../../examples/applications/retrieve_rerank/README.html#retrieval-bi-encoder)
    * [Re-Ranker: Cross-Encoder](../../../examples/applications/retrieve_rerank/README.html#re-ranker-cross-encoder)
    * [Example Scripts](../../../examples/applications/retrieve_rerank/README.html#example-scripts)
    * [Pre-trained Bi-Encoders (Retrieval)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
    * [Pre-trained Cross-Encoders (Re-Ranker)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
  * [Clustering](../../../examples/applications/clustering/README.html)
    * [k-Means](../../../examples/applications/clustering/README.html#k-means)
    * [Agglomerative Clustering](../../../examples/applications/clustering/README.html#agglomerative-clustering)
    * [Fast Clustering](../../../examples/applications/clustering/README.html#fast-clustering)
    * [Topic Modeling](../../../examples/applications/clustering/README.html#topic-modeling)
  * [Paraphrase Mining](../../../examples/applications/paraphrase-mining/README.html)
    * [`paraphrase_mining()`](../../../examples/applications/paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
    * [Translated Sentence

Mining](../../../examples/applications/parallel-sentence-mining/README.html)

  * [Margin Based Mining](../../../examples/applications/parallel-sentence-mining/README.html#margin-based-mining)

  * [Examples](../../../examples/applications/parallel-sentence-mining/README.html#examples)

 * [Image Search](../../../examples/applications/image-search/README.html)

  * [Installation](../../../examples/applications/image-search/README.html#installation)

  * [Usage](../../../examples/applications/image-search/README.html#usage)

  * [Examples](../../../examples/applications/image-search/README.html#examples)

 * [Embedding Quantization](../../../examples/applications/embedding-quantization/README.html)

  * [Binary Quantization](../../../examples/applications/embedding-quantization/README.html#binary-quantization)

  * [Scalar (int8) Quantization](../../../examples/applications/embedding-quantization/README.html#scalar-int8-quantization)

  * [Additional extensions](../../../examples/applications/embedding-quantization/README.html#additional-extensions)

  * [Demo](../../../examples/applications/embedding-quantization/README.html#demo)

  * [Try it yourself](../../../examples/applications/embedding-quantization/README.html#try-it-yourself)

 * [Speeding up Inference](../../sentence_transformer/usage/efficiency.html)

  * [PyTorch](../../sentence_transformer/usage/efficiency.html#pytorch)

  * [ONNX](../../sentence_transformer/usage/efficiency.html#onnx)

  * [OpenVINO](../../sentence_transformer/usage/efficiency.html#openvino)

  * [Benchmarks](../../sentence_transformer/usage/efficiency.html#benchmarks)

 * [Creating Custom Models](../../sentence_transformer/usage/custom_models.html)

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

    * [Training Arguments](../../sentence_transformer/training_overview.html#training-arguments)

    * [Evaluator](../../sentence_transformer/training_overview.html#evaluator)

    * [Trainer](../../sentence_transformer/training_overview.html#trainer)

      * [Callbacks](../../sentence_transformer/training_overview.html#callbacks)

    * [Multi-Dataset Training](../../sentence_transformer/training_overview.html#multi-dataset-training)

    * [Deprecated Training](../../sentence_transformer/training_overview.html#deprecated-training)

    * [Best Base Embedding Models](../../sentence_transformer/training_overview.html#best-base-embedding-models)

  * [Dataset Overview](../../sentence_transformer/dataset_overview.html)

    * [Datasets on the Hugging Face Hub](../../sentence_transformer/dataset_overview.html#datasets-on-the-hugging-face-hub)

    * [Pre-existing Datasets](../../sentence_transformer/dataset_overview.html#pre-existing-datasets)

  * [Loss Overview](../../sentence_transformer/loss_overview.html)

    * [Loss modifiers](../../sentence_transformer/loss_overview.html#loss-modifiers)

    * [Distillation](../../sentence_transformer/loss_overview.html#distillation)

    * [Commonly used Loss Functions](../../sentence_transformer/loss_overview.html#commonly-used-loss-functions)

    * [Custom Loss Functions](../../sentence_transformer/loss_overview.html#custom-loss-functions)

  * [Training Examples](../../sentence_transformer/training/examples.html)

    * [Semantic Textual Similarity](../../../examples/training/sts/README.html)

      * [Training data](../../../examples/training/sts/README.html#training-data)

      * [Loss Function](../../../examples/training/sts/README.html#loss-function)

    * [Natural Language Inference](../../../examples/training/nli/README.html)

      * [Data](../../../examples/training/nli/README.html#data)

      * [SoftmaxLoss](../../../examples/training/nli/README.html#softmaxloss)

      * [MultipleNegativesRankingLoss](../../../examples/training/nli/README.html#multiplenegativesrankin

gloss)

  * [Paraphrase Data](../../../examples/training/paraphrases/README.html)

    * [Pre-Trained Models](../../../examples/training/paraphrases/README.html#pre-trained-models)

  * [Quora Duplicate Questions](../../../examples/training/quora_duplicate_questions/README.html)

    * [Training](../../../examples/training/quora_duplicate_questions/README.html#training)

    * [MultipleNegativesRankingLoss](../../../examples/training/quora_duplicate_questions/README.html#multiplenegativesrankingloss)

    * [Pretrained Models](../../../examples/training/quora_duplicate_questions/README.html#pretrained-models)

  * [MS MARCO](../../../examples/training/ms_marco/README.html)

    * [Bi-Encoder](../../../examples/training/ms_marco/README.html#bi-encoder)

  * [Matryoshka Embeddings](../../../examples/training/matryoshka/README.html)

    * [Use Cases](../../../examples/training/matryoshka/README.html#use-cases)

    * [Results](../../../examples/training/matryoshka/README.html#results)

    * [Training](../../../examples/training/matryoshka/README.html#training)

    * [Inference](../../../examples/training/matryoshka/README.html#inference)

    * [Code Examples](../../../examples/training/matryoshka/README.html#code-examples)

  * [Adaptive Layers](../../../examples/training/adaptive_layer/README.html)

    * [Use Cases](../../../examples/training/adaptive_layer/README.html#use-cases)

    * [Results](../../../examples/training/adaptive_layer/README.html#results)

    * [Training](../../../examples/training/adaptive_layer/README.html#training)

    * [Inference](../../../examples/training/adaptive_layer/README.html#inference)

    * [Code Examples](../../../examples/training/adaptive_layer/README.html#code-examples)

  * [Multilingual Models](../../../examples/training/multilingual/README.html)

    * [Extend your own models](../../../examples/training/multilingual/README.html#extend-your-own-models)

* [Training](../../../examples/training/multilingual/README.html#training)

* [Datasets](../../../examples/training/multilingual/README.html#datasets)

* [Sources for Training Data](../../../examples/training/multilingual/README.html#sources-for-training-data)

* [Evaluation](../../../examples/training/multilingual/README.html#evaluation)

* [Available Pre-trained Models](../../../examples/training/multilingual/README.html#available-pre-trained-models)

* [Usage](../../../examples/training/multilingual/README.html#usage)

* [Performance](../../../examples/training/multilingual/README.html#performance)

* [Citation](../../../examples/training/multilingual/README.html#citation)

* [Model Distillation](../../../examples/training/distillation/README.html)

* [Knowledge Distillation](../../../examples/training/distillation/README.html#knowledge-distillation)

* [Speed - Performance Trade-Off](../../../examples/training/distillation/README.html#speed-performance-trade-off)

* [Dimensionality Reduction](../../../examples/training/distillation/README.html#dimensionality-reduction)

* [Quantization](../../../examples/training/distillation/README.html#quantization)

* [Augmented SBERT](../../../examples/training/data_augmentation/README.html)

* [Motivation](../../../examples/training/data_augmentation/README.html#motivation)

* [Extend to your own datasets](../../../examples/training/data_augmentation/README.html#extend-to-your-own-datasets)

* [Methodology](../../../examples/training/data_augmentation/README.html#methodology)

* [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../../../examples/training/data_augmentation/README.html#scenario-1-limited-or-small-annotated-datasets-few-labeled-sentence-pairs)

* [Scenario 2: No annotated datasets (Only unlabeled

sentence-pairs)](../../../examples/training/data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)

    * [Training](../../../examples/training/data_augmentation/README.html#training)

    * [Citation](../../../examples/training/data_augmentation/README.html#citation)

  * [Training with Prompts](../../../examples/training/prompts/README.html)

    * [What are Prompts?](../../../examples/training/prompts/README.html#what-are-prompts)

    * [Why would we train with Prompts?](../../../examples/training/prompts/README.html#why-would-we-train-with-prompts)

    * [How do we train with Prompts?](../../../examples/training/prompts/README.html#how-do-we-train-with-prompts)

  * [Training with PEFT Adapters](../../../examples/training/peft/README.html)

    * [Compatibility Methods](../../../examples/training/peft/README.html#compatibility-methods)

    * [Adding a New Adapter](../../../examples/training/peft/README.html#adding-a-new-adapter)

    * [Loading a Pretrained Adapter](../../../examples/training/peft/README.html#loading-a-pretrained-adapter)

    * [Training Script](../../../examples/training/peft/README.html#training-script)

  * [Unsupervised Learning](../../../examples/unsupervised_learning/README.html)

    * [TSDAE](../../../examples/unsupervised_learning/README.html#tsdae)

    * [SimCSE](../../../examples/unsupervised_learning/README.html#simcse)

    * [CT](../../../examples/unsupervised_learning/README.html#ct)

    * [CT (In-Batch Negative Sampling)](../../../examples/unsupervised_learning/README.html#ct-in-batch-negative-sampling)

    * [Masked Language Model (MLM)](../../../examples/unsupervised_learning/README.html#masked-language-model-mlm)

    * [GenQ](../../../examples/unsupervised_learning/README.html#genq)

    * [GPL](../../../examples/unsupervised_learning/README.html#gpl)

    * [Performance

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

Cross Encoder

  * [Usage](../../cross_encoder/usage/usage.html)
    * [Retrieve & Re-Rank](../../../examples/applications/retrieve_rerank/README.html)
      * [Retrieve & Re-Rank Pipeline](../../../examples/applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)
      * [Retrieval: Bi-Encoder](../../../examples/applications/retrieve_rerank/README.html#retrieval-bi-encoder)
      * [Re-Ranker:

Cross-Encoder](../../../examples/applications/retrieve_rerank/README.html#re-ranker-cross-encoder)

* [Example Scripts](../../../examples/applications/retrieve_rerank/README.html#example-scripts)

* [Pre-trained Bi-Encoders (Retrieval)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)

* [Pre-trained Cross-Encoders (Re-Ranker)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)

* [Pretrained Models](../../cross_encoder/pretrained_models.html)

* [MS MARCO](../../cross_encoder/pretrained_models.html#ms-marco)

* [SQuAD (QNLI)](../../cross_encoder/pretrained_models.html#squad-qnli)

* [STSbenchmark](../../cross_encoder/pretrained_models.html#stsbenchmark)

* [Quora Duplicate Questions](../../cross_encoder/pretrained_models.html#quora-duplicate-questions)

* [NLI](../../cross_encoder/pretrained_models.html#nli)

* [Community Models](../../cross_encoder/pretrained_models.html#community-models)

* [Training Overview](../../cross_encoder/training_overview.html)

* [Training Examples](../../cross_encoder/training/examples.html)

* [MS MARCO](../../../examples/training/ms_marco/cross_encoder_README.html)

* [Cross-Encoder](../../../examples/training/ms_marco/cross_encoder_README.html#cross-encoder)

* [Cross-Encoder Knowledge Distillation](../../../examples/training/ms_marco/cross_encoder_README.html#cross-encoder-knowledge-distillation)

Package Reference

* [Sentence Transformer](index.html)
  * [SentenceTransformer](SentenceTransformer.html)
    * [SentenceTransformer](SentenceTransformer.html#id1)
    * [SentenceTransformerModelCardData](SentenceTransformer.html#sentencetransformermodelcarddata)
    * [SimilarityFunction](SentenceTransformer.html#similarityfunction)
  * [Trainer](trainer.html)
    * [SentenceTransformerTrainer](trainer.html#sentencetransformertrainer)
  * [Training Arguments](training_args.html)
    * [SentenceTransformerTrainingArguments](training_args.html#sentencetransformertrainingarguments)
  * [Losses](losses.html)
    * [BatchAllTripletLoss](losses.html#batchalltripletloss)
    * [BatchHardSoftMarginTripletLoss](losses.html#batchhardsoftmargintripletloss)
    * [BatchHardTripletLoss](losses.html#batchhardtripletloss)
    * [BatchSemiHardTripletLoss](losses.html#batchsemihardtripletloss)
    * [ContrastiveLoss](losses.html#contrastiveloss)
    * [OnlineContrastiveLoss](losses.html#onlinecontrastiveloss)
    * [ContrastiveTensionLoss](losses.html#contrastivetensionloss)
    * [ContrastiveTensionLossInBatchNegatives](losses.html#contrastivetensionlossinbatchnegatives)
    * [CoSENTLoss](losses.html#cosentloss)
    * [AnglELoss](losses.html#angleloss)
    * [CosineSimilarityLoss](losses.html#cosinesimilarityloss)

  * [DenoisingAutoEncoderLoss](losses.html#denoisingautoencoderloss)
  * [GISTEmbedLoss](losses.html#gistembedloss)
  * [CachedGISTEmbedLoss](losses.html#cachedgistembedloss)
  * [MSELoss](losses.html#mseloss)
  * [MarginMSELoss](losses.html#marginmseloss)
  * [MatryoshkaLoss](losses.html#matryoshkaloss)
  * [Matryoshka2dLoss](losses.html#matryoshka2dloss)
  * [AdaptiveLayerLoss](losses.html#adaptivelayerloss)
  * [MegaBatchMarginLoss](losses.html#megabatchmarginloss)
  * [MultipleNegativesRankingLoss](losses.html#multiplenegativesrankingloss)
  * [CachedMultipleNegativesRankingLoss](losses.html#cachedmultiplenegativesrankingloss)
  * [MultipleNegativesSymmetricRankingLoss](losses.html#multiplenegativessymmetricrankingloss)
  * [CachedMultipleNegativesSymmetricRankingLoss](losses.html#cachedmultiplenegativessymmetricrankingloss)
  * [SoftmaxLoss](losses.html#softmaxloss)
  * [TripletLoss](losses.html#tripletloss)
* Samplers
  * BatchSamplers
  * MultiDatasetBatchSamplers
* [Evaluation](evaluation.html)
  * [BinaryClassificationEvaluator](evaluation.html#binaryclassificationevaluator)
  * [EmbeddingSimilarityEvaluator](evaluation.html#embeddingsimilarityevaluator)
  * [InformationRetrievalEvaluator](evaluation.html#informationretrievalevaluator)
  * [NanoBEIREvaluator](evaluation.html#nanobeirevaluator)
  * [MSEEvaluator](evaluation.html#mseevaluator)

    * [ParaphraseMiningEvaluator](evaluation.html#paraphraseminingevaluator)

    * [RerankingEvaluator](evaluation.html#rerankingevaluator)

    * [SentenceEvaluator](evaluation.html#sentenceevaluator)

    * [SequentialEvaluator](evaluation.html#sequentialevaluator)

    * [TranslationEvaluator](evaluation.html#translationevaluator)

    * [TripletEvaluator](evaluation.html#tripletevaluator)

  * [Datasets](datasets.html)

    * [ParallelSentencesDataset](datasets.html#parallelsentencesdataset)

    * [SentenceLabelDataset](datasets.html#sentencelabeldataset)

    * [DenoisingAutoEncoderDataset](datasets.html#denoisingautoencoderdataset)

    * [NoDuplicatesDataLoader](datasets.html#noduplicatesdataloader)

  * [Models](models.html)

    * [Main Classes](models.html#main-classes)

    * [Further Classes](models.html#further-classes)

  * [quantization](quantization.html)

    * [`quantize_embeddings()`](quantization.html#sentence_transformers.quantization.quantize_embeddings)

    * [`semantic_search_faiss()`](quantization.html#sentence_transformers.quantization.semantic_search_faiss)

    * [`semantic_search_usearch()`](quantization.html#sentence_transformers.quantization.semantic_search_usearch)

 * [Cross Encoder](../cross_encoder/index.html)

  * [CrossEncoder](../cross_encoder/cross_encoder.html)

    * [CrossEncoder](../cross_encoder/cross_encoder.html#id1)

* [Training Inputs](../cross_encoder/cross_encoder.html#training-inputs)
* [Evaluation](../cross_encoder/evaluation.html)
* [CEBinaryAccuracyEvaluator](../cross_encoder/evaluation.html#cebinaryaccuracyevaluator)
* [CEBinaryClassificationEvaluator](../cross_encoder/evaluation.html#cebinaryclassificationevaluator)
* [CECorrelationEvaluator](../cross_encoder/evaluation.html#cecorrelationevaluator)
* [CEF1Evaluator](../cross_encoder/evaluation.html#cef1evaluator)
* [CESoftmaxAccuracyEvaluator](../cross_encoder/evaluation.html#cesoftmaxaccuracyevaluator)
* [CERerankingEvaluator](../cross_encoder/evaluation.html#cererankingevaluator)
* [util](../util.html)
* [Helper Functions](../util.html#module-sentence_transformers.util)
* [`community_detection()`](../util.html#sentence_transformers.util.community_detection)
* [`http_get()`](../util.html#sentence_transformers.util.http_get)
* [`is_training_available()`](../util.html#sentence_transformers.util.is_training_available)
* [`mine_hard_negatives()`](../util.html#sentence_transformers.util.mine_hard_negatives)
* [`normalize_embeddings()`](../util.html#sentence_transformers.util.normalize_embeddings)
* [`paraphrase_mining()`](../util.html#sentence_transformers.util.paraphrase_mining)
* [`semantic_search()`](../util.html#sentence_transformers.util.semantic_search)
* [`truncate_embeddings()`](../util.html#sentence_transformers.util.truncate_embeddings)
* [Model Optimization](../util.html#module-sentence_transformers.backend)
* [`export_dynamic_quantized_onnx_model()`](../util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)
* [`export_optimized_onnx_model()`](../util.html#sentence_transformers.backend.export_optimized_onnx_model)

*

[`export_static_quantized_openvino_model()`](../util.html#sentence_transformers.backend.export_static_quantized_openvino_model)

* [Similarity Metrics](../util.html#module-sentence_transformers.util)

* [`cos_sim()`](../util.html#sentence_transformers.util.cos_sim)

* [`dot_score()`](../util.html#sentence_transformers.util.dot_score)

* [`euclidean_sim()`](../util.html#sentence_transformers.util.euclidean_sim)

* [`manhattan_sim()`](../util.html#sentence_transformers.util.manhattan_sim)

* [`pairwise_cos_sim()`](../util.html#sentence_transformers.util.pairwise_cos_sim)

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

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

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

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

* [](../../../index.html)

* [Sentence Transformer](index.html)

* Samplers

* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/docs/package_reference/sentence_transformer/sampler.md)

* * *

# Samplersïƒ•

## BatchSamplersïƒ•

_class

_sentence_transformers.training_args.BatchSamplers(_value_)[[source]](https://github.com/UKPLab

/sentence-

transformers/blob/master/sentence_transformers\\training_args.py#L14-L77)ïƒ•

Stores the acceptable string identifiers for batch samplers.

The batch sampler is responsible for determining how samples are grouped into

batches during training. Valid options are:

* `BatchSamplers.BATCH_SAMPLER`: **[default]** Uses `DefaultBatchSampler`, the default PyTorch batch sampler.

* `BatchSamplers.NO_DUPLICATES`: Uses `NoDuplicatesBatchSampler`, ensuring no duplicate samples in a batch. Recommended for losses that use in-batch negatives, such as:

> *

>

[`MultipleNegativesRankingLoss`](losses.html#sentence_transformers.losses.MultipleNegativesRan

kingLoss

> "sentence_transformers.losses.MultipleNegativesRankingLoss")

>

> *

>

[`CachedMultipleNegativesRankingLoss`](losses.html#sentence_transformers.losses.CachedMultipleNegativesRankingLoss

> "sentence_transformers.losses.CachedMultipleNegativesRankingLoss")

>

> *

>

[`MultipleNegativesSymmetricRankingLoss`](losses.html#sentence_transformers.losses.MultipleNegativesSymmetricRankingLoss

> "sentence_transformers.losses.MultipleNegativesSymmetricRankingLoss")

>

> *

>

[`CachedMultipleNegativesSymmetricRankingLoss`](losses.html#sentence_transformers.losses.CachedMultipleNegativesSymmetricRankingLoss

> "sentence_transformers.losses.CachedMultipleNegativesSymmetricRankingLoss")

>

> *

> [`MegaBatchMarginLoss`](losses.html#sentence_transformers.losses.MegaBatchMarginLoss

> "sentence_transformers.losses.MegaBatchMarginLoss")

>

> *

> [`GISTEmbedLoss`](losses.html#sentence_transformers.losses.GISTEmbedLoss

> "sentence_transformers.losses.GISTEmbedLoss")

>

> *

> [`CachedGISTEmbedLoss`](losses.html#sentence_transformers.losses.CachedGISTEmbedLoss

> "sentence_transformers.losses.CachedGISTEmbedLoss")

* `BatchSamplers.GROUP_BY_LABEL`: Uses `GroupByLabelBatchSampler`, ensuring that each batch has 2+ samples from the same label. Recommended for losses that require multiple samples from the same label, such as:

> *
> [`BatchAllTripletLoss`](losses.html#sentence_transformers.losses.BatchAllTripletLoss
> "sentence_transformers.losses.BatchAllTripletLoss")
>
> *
>
[`BatchHardSoftMarginTripletLoss`](losses.html#sentence_transformers.losses.BatchHardSoftMarginTripletLoss
> "sentence_transformers.losses.BatchHardSoftMarginTripletLoss")
>
> *
> [`BatchHardTripletLoss`](losses.html#sentence_transformers.losses.BatchHardTripletLoss
> "sentence_transformers.losses.BatchHardTripletLoss")
>
> *
>
[`BatchSemiHardTripletLoss`](losses.html#sentence_transformers.losses.BatchSemiHardTripletLoss
> "sentence_transformers.losses.BatchSemiHardTripletLoss")

If you want to use a custom batch sampler, you can create a new Trainer class
that inherits from

[`SentenceTransformerTrainer`](trainer.html#sentence_transformers.trainer.SentenceTransformerTr
ainer
"sentence_transformers.trainer.SentenceTransformerTrainer") and overrides the
[`get_batch_sampler()`](trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer.get
_batch_sampler
"sentence_transformers.trainer.SentenceTransformerTrainer.get_batch_sampler")
method. The method must return a class instance that supports `__iter__` and
`__len__` methods. The former should yield a list of indices for each batch,
and the latter should return the number of batches.

Usage:

```
    from sentence_transformers import SentenceTransformer, SentenceTransformerTrainer, SentenceTransformerTrainingArguments
    from sentence_transformers.training_args import BatchSamplers
    from sentence_transformers.losses import MultipleNegativesRankingLoss
    from datasets import Dataset

    model = SentenceTransformer("microsoft/mpnet-base")
    train_dataset = Dataset.from_dict({
        "anchor": ["It's nice weather outside today.", "He drove to work."],
        "positive": ["It's so sunny.", "He took the car to the office."],
    })
    loss = MultipleNegativesRankingLoss(model)
```

```
args = SentenceTransformerTrainingArguments(
    output_dir="checkpoints",
    batch_sampler=BatchSamplers.NO_DUPLICATES,
)
trainer = SentenceTransformerTrainer(
    model=model,
    args=args,
    train_dataset=train_dataset,
    loss=loss,
)
trainer.train()
```

_class _sentence_transformers.sampler.DefaultBatchSampler(_* args_, _** kwargs_)[[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\sampler.py#L35-L46)ïƒ•

This sampler is the default batch sampler used in the SentenceTransformer library. It is equivalent to the PyTorch BatchSampler.

Parameters:

* **sampler** (_Sampler_ _or_ _Iterable_) â€“ The sampler used for sampling elements from the

dataset, such as SubsetRandomSampler.

* **batch_size** (_int_) â€“ Number of samples per batch.

* **drop_last** (_bool_) â€“ If True, drop the last incomplete batch if the dataset size is not divisible by the batch size.

_class _sentence_transformers.sampler.NoDuplicatesBatchSampler(_dataset : Dataset_, _batch_size : int_, _drop_last : bool_, _valid_label_columns : list[str] = []_, _generator : [Generator](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") | None = None_, _seed : int = 0_)[[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\sampler.py#L133-L222)ïƒ•

This sampler creates batches such that each batch contains samples where the

values are unique, even across columns. This is useful when losses consider

other samples in a batch to be in-batch negatives, and you want to ensure that

the negatives are not duplicates of the anchor/positive sample.

Recommended for:

*

[`MultipleNegativesRankingLoss`](losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss")

* [`CachedMultipleNegativesRankingLoss`](losses.html#sentence_transformers.losses.CachedMultipleNegativesRankingLoss "sentence_transformers.losses.CachedMultipleNegativesRankingLoss")

* [`MultipleNegativesSymmetricRankingLoss`](losses.html#sentence_transformers.losses.MultipleNegativesSymmetricRankingLoss "sentence_transformers.losses.MultipleNegativesSymmetricRankingLoss")

* [`CachedMultipleNegativesSymmetricRankingLoss`](losses.html#sentence_transformers.losses.CachedMultipleNegativesSymmetricRankingLoss "sentence_transformers.losses.CachedMultipleNegativesSymmetricRankingLoss")

* [`MegaBatchMarginLoss`](losses.html#sentence_transformers.losses.MegaBatchMarginLoss "sentence_transformers.losses.MegaBatchMarginLoss")

* [`GISTEmbedLoss`](losses.html#sentence_transformers.losses.GISTEmbedLoss "sentence_transformers.losses.GISTEmbedLoss")

* [`CachedGISTEmbedLoss`](losses.html#sentence_transformers.losses.CachedGISTEmbedLoss "sentence_transformers.losses.CachedGISTEmbedLoss")

Parameters:

* **dataset** (_Dataset_) â€“ The dataset to sample from.

* **batch_size** (_int_) â€“ Number of samples per batch.

* **drop_last** (_bool_) â€“ If True, drop the last incomplete batch if the dataset size is not divisible by the batch size.

* **valid_label_columns** (_List_ _[__str_ _]_) â€“ List of column names to check for labels. The first column name from `valid_label_columns` found in the dataset will be used as the label column.

* **generator** ([_torch.Generator_](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") _,__optional_) â€“ Optional random number generator for shuffling the indices.

* **seed** (_int_ _,__optional_) â€“ Seed for the random number generator to ensure reproducibility.

_class _sentence_transformers.sampler.GroupByLabelBatchSampler(_dataset : Dataset_, _batch_size : int_, _drop_last : bool_, _valid_label_columns : list[str] | None = None_, _generator : [Generator](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") | None = None_, _seed : int = 0_)[[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\sampler.py#L49-L130)ïƒ•

This sampler groups samples by their labels and aims to create batches such that each batch contains samples where the labels are as homogeneous as possible. This sampler is meant to be used alongside the `Batch...TripletLoss` classes, which require that each batch contains at least 2 examples per label class.

Recommended for:

- [`BatchAllTripletLoss`](losses.html#sentence_transformers.losses.BatchAllTripletLoss "sentence_transformers.losses.BatchAllTripletLoss")
- [`BatchHardSoftMarginTripletLoss`](losses.html#sentence_transformers.losses.BatchHardSoftMarginTripletLoss "sentence_transformers.losses.BatchHardSoftMarginTripletLoss")
- [`BatchHardTripletLoss`](losses.html#sentence_transformers.losses.BatchHardTripletLoss "sentence_transformers.losses.BatchHardTripletLoss")
- [`BatchSemiHardTripletLoss`](losses.html#sentence_transformers.losses.BatchSemiHardTripletLoss "sentence_transformers.losses.BatchSemiHardTripletLoss")

Parameters:

* **dataset** (_Dataset_) â€“ The dataset to sample from.

* **batch_size** (_int_) â€“ Number of samples per batch. Must be divisible by 2.

* **drop_last** (_bool_) â€“ If True, drop the last incomplete batch if the dataset size is not divisible by the batch size.

* **valid_label_columns** (_List_ _[__str_ _]_) â€“ List of column names to check for labels. The first column name from `valid_label_columns` found in the dataset will be used as the label column.

* **generator** ([_torch.Generator_](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") _,__optional_) â€“ Optional random number generator for shuffling the indices.

* **seed** (_int_ _,__optional_) â€“ Seed for the random number generator to ensure reproducibility.

## MultiDatasetBatchSamplersïƒ•

_class _sentence_transformers.training_args.MultiDatasetBatchSamplers(_value_)[[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\training_args.py#L80-L140)ïƒ•

Stores the acceptable string identifiers for multi-dataset batch samplers.

The multi-dataset batch sampler is responsible for determining in what order
batches are sampled from multiple datasets during training. Valid options are:

* `MultiDatasetBatchSamplers.ROUND_ROBIN`: Uses `RoundRobinBatchSampler`, which uses round-robin sampling from each dataset until one is exhausted. With this strategy, itâ€™s likely that not all samples from each dataset are used, but each dataset is sampled from equally.

* `MultiDatasetBatchSamplers.PROPORTIONAL`: **[default]** Uses `ProportionalBatchSampler`, which samples from each dataset in proportion to its size. With this strategy, all samples from each dataset are used and larger datasets are sampled from more frequently.

Usage:

```
from sentence_transformers import SentenceTransformer, SentenceTransformerTrainer, SentenceTransformerTrainingArguments
from sentence_transformers.training_args import MultiDatasetBatchSamplers
from sentence_transformers.losses import CoSENTLoss
from datasets import Dataset, DatasetDict

model = SentenceTransformer("microsoft/mpnet-base")
train_general = Dataset.from_dict({
    "sentence_A": ["It's nice weather outside today.", "He drove to work."],
    "sentence_B": ["It's so sunny.", "He took the car to the bank."],
    "score": [0.9, 0.4],
})
train_medical = Dataset.from_dict({
    "sentence_A": ["The patient has a fever.", "The doctor prescribed medication.", "The patient is sweating."],
    "sentence_B": ["The patient feels hot.", "The medication was given to the patient.", "The patient is perspiring."],
    "score": [0.8, 0.6, 0.7],
})
train_legal = Dataset.from_dict({
    "sentence_A": ["This contract is legally binding.", "The parties agree to the terms and conditions."],
    "sentence_B": ["Both parties acknowledge their obligations.", "By signing this agreement, the parties enter into a legal relationship."],
    "score": [0.7, 0.8],
})
train_dataset = DatasetDict({
    "general": train_general,
    "medical": train_medical,
    "legal": train_legal,
})

loss = CoSENTLoss(model)
args = SentenceTransformerTrainingArguments(
    output_dir="checkpoints",
    multi_dataset_batch_sampler=MultiDatasetBatchSamplers.PROPORTIONAL,
)
trainer = SentenceTransformerTrainer(
    model=model,
    args=args,
    train_dataset=train_dataset,
    loss=loss,
)
trainer.train()
```

_class _sentence_transformers.sampler.RoundRobinBatchSampler(_dataset : [ConcatDataset](https://pytorch.org/docs/stable/data.html#torch.utils.data.ConcatDataset "\(in PyTorch v2.5\)")_, _batch_samplers : list[[BatchSampler](https://pytorch.org/docs/stable/data.html#torch.utils.data.BatchSampler "\(in PyTorch v2.5\)")]_, _generator : [Generator](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") | None = None_, _seed : int | None = None_)[[source]](https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers\\sampler.py#L225-L270)ïƒ•

Batch sampler that yields batches in a round-robin fashion from multiple batch samplers, until one is exhausted. With this sampler, itâ€™s unlikely that all samples from each dataset are used, but we do ensure that each dataset is sampled from equally.

Parameters:

* **dataset** (_ConcatDataset_) â€“ A concatenation of multiple datasets.

* **batch_samplers** (_List_ _[__BatchSampler_ _]_) â€“ A list of batch samplers, one for each dataset in the ConcatDataset.

* **generator** ([_torch.Generator_](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") _,__optional_) â€“ A generator for reproducible sampling. Defaults to None.

* **seed** (_int_ _,__optional_) â€“ A seed for the generator. Defaults to None.

_class _sentence_transformers.sampler.ProportionalBatchSampler(_dataset :

[ConcatDataset](https://pytorch.org/docs/stable/data.html#torch.utils.data.ConcatDataset

"\(in PyTorch v2.5\)")_, _batch_samplers :

list[[BatchSampler](https://pytorch.org/docs/stable/data.html#torch.utils.data.BatchSampler

"\(in PyTorch v2.5\)")]_, _generator :

[Generator](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator

"\(in PyTorch v2.5\)")_, _seed :

int_)[[source]](https://github.com/UKPLab/sentence-

transformers/blob/master/sentence_transformers\\sampler.py#L273-L316)ïƒ•

Batch sampler that samples from each dataset in proportion to its size, until all are exhausted simultaneously. With this sampler, all samples from each dataset are used and larger datasets are sampled from more frequently.

Parameters:

* **dataset** (_ConcatDataset_) â€“ A concatenation of multiple datasets.

* **batch_samplers** (_List_ _[__BatchSampler_ _]_) â€“ A list of batch samplers, one for each dataset in the ConcatDataset.

* **generator** ([_torch.Generator_](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator "\(in PyTorch v2.5\)") _,__optional_) â€“ A generator for reproducible sampling. Defaults to None.

* **seed** (_int_ _,__optional_) â€“ A seed for the generator. Defaults to None.

[Previous](losses.html "Losses") [Next](evaluation.html "Evaluation")

* * *

(C) Copyright 2025.

Built with [Sphinx](https://www.sphinx-doc.org/) using a [theme](https://github.com/readthedocs/sphinx_rtd_theme) provided by [Read the

Docs](https://readthedocs.org).