[![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](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
* 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](efficiency.html)
    * [PyTorch](efficiency.html#pytorch)
    * [ONNX](efficiency.html#onnx)
    * [OpenVINO](efficiency.html#openvino)
    * [Benchmarks](efficiency.html#benchmarks)
  * [Creating Custom Models](custom_models.html)
    * [Structure of Sentence Transformer

Models](custom_models.html#structure-of-sentence-transformer-models)

* [Sentence Transformer Model from a Transformers Model](custom_models.html#sentence-transformer-model-from-a-transformers-model)

* [Pretrained Models](../pretrained_models.html)
  * [Original Models](../pretrained_models.html#original-models)
  * [Semantic Search Models](../pretrained_models.html#semantic-search-models)
    * [Multi-QA Models](../pretrained_models.html#multi-qa-models)
    * [MSMARCO Passage Models](../pretrained_models.html#msmarco-passage-models)
  * [Multilingual Models](../pretrained_models.html#multilingual-models)
    * [Semantic Similarity Models](../pretrained_models.html#semantic-similarity-models)
    * [Bitext Mining](../pretrained_models.html#bitext-mining)
  * [Image & Text-Models](../pretrained_models.html#image-text-models)
  * [INSTRUCTOR models](../pretrained_models.html#instructor-models)
  * [Scientific Similarity Models](../pretrained_models.html#scientific-similarity-models)
* [Training Overview](../training_overview.html)
  * [Why Finetune?](../training_overview.html#why-finetune)
  * [Training Components](../training_overview.html#training-components)
  * [Dataset](../training_overview.html#dataset)
    * [Dataset Format](../training_overview.html#dataset-format)
  * [Loss Function](../training_overview.html#loss-function)
  * [Training Arguments](../training_overview.html#training-arguments)
  * [Evaluator](../training_overview.html#evaluator)
  * [Trainer](../training_overview.html#trainer)
    * [Callbacks](../training_overview.html#callbacks)
  * [Multi-Dataset Training](../training_overview.html#multi-dataset-training)
  * [Deprecated Training](../training_overview.html#deprecated-training)
  * [Best Base Embedding Models](../training_overview.html#best-base-embedding-models)

* [Dataset Overview](../dataset_overview.html)
* [Datasets on the Hugging Face Hub](../dataset_overview.html#datasets-on-the-hugging-face-hub)
* [Pre-existing Datasets](../dataset_overview.html#pre-existing-datasets)
* [Loss Overview](../loss_overview.html)
* [Loss modifiers](../loss_overview.html#loss-modifiers)
* [Distillation](../loss_overview.html#distillation)
* [Commonly used Loss Functions](../loss_overview.html#commonly-used-loss-functions)
* [Custom Loss Functions](../loss_overview.html#custom-loss-functions)
* [Training Examples](../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#multiplenegativesrankingloss)
* [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-labelin

g)

* [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](../training/distributed.html)

* [Comparison](../training/distributed.html#comparison)

* [FSDP](../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](../../package_reference/sentence_transformer/index.html)
    * [SentenceTransformer](../../package_reference/sentence_transformer/SentenceTransformer.html)
    * [SentenceTransformer](../../package_reference/sentence_transformer/SentenceTransformer.html#id1)
    * [SentenceTransformerModelCardData](../../package_reference/sentence_transformer/SentenceTran

sformer.html#sentencetransformermodelcarddata)

*

[SimilarityFunction](../../package_reference/sentence_transformer/SentenceTransformer.html#similarityfunction)

* [Trainer](../../package_reference/sentence_transformer/trainer.html)

*

[SentenceTransformerTrainer](../../package_reference/sentence_transformer/trainer.html#sentencetransformertrainer)

* [Training Arguments](../../package_reference/sentence_transformer/training_args.html)

*

[SentenceTransformerTrainingArguments](../../package_reference/sentence_transformer/training_args.html#sentencetransformertrainingarguments)

* [Losses](../../package_reference/sentence_transformer/losses.html)

*

[BatchAllTripletLoss](../../package_reference/sentence_transformer/losses.html#batchalltripletloss)

*

[BatchHardSoftMarginTripletLoss](../../package_reference/sentence_transformer/losses.html#batchhardsoftmargintripletloss)

*

[BatchHardTripletLoss](../../package_reference/sentence_transformer/losses.html#batchhardtripletloss)

*

[BatchSemiHardTripletLoss](../../package_reference/sentence_transformer/losses.html#batchsemihardtripletloss)

* [ContrastiveLoss](../../package_reference/sentence_transformer/losses.html#contrastiveloss)

*

[OnlineContrastiveLoss](../../package_reference/sentence_transformer/losses.html#onlinecontrastiv

eloss)

*

[ContrastiveTensionLoss](../../package_reference/sentence_transformer/losses.html#contrastivetensionloss)

*

[ContrastiveTensionLossInBatchNegatives](../../package_reference/sentence_transformer/losses.html#contrastivetensionlossinbatchnegatives)

    * [CoSENTLoss](../../package_reference/sentence_transformer/losses.html#cosentloss)

    * [AnglELoss](../../package_reference/sentence_transformer/losses.html#angleloss)

*

[CosineSimilarityLoss](../../package_reference/sentence_transformer/losses.html#cosinesimilarityloss)

*

[DenoisingAutoEncoderLoss](../../package_reference/sentence_transformer/losses.html#denoisingautoencoderloss)

    * [GISTEmbedLoss](../../package_reference/sentence_transformer/losses.html#gistembedloss)

*

[CachedGISTEmbedLoss](../../package_reference/sentence_transformer/losses.html#cachedgistembedloss)

    * [MSELoss](../../package_reference/sentence_transformer/losses.html#mseloss)

    * [MarginMSELoss](../../package_reference/sentence_transformer/losses.html#marginmseloss)

    * [MatryoshkaLoss](../../package_reference/sentence_transformer/losses.html#matryoshkaloss)

*

[Matryoshka2dLoss](../../package_reference/sentence_transformer/losses.html#matryoshka2dloss)

*

[AdaptiveLayerLoss](../../package_reference/sentence_transformer/losses.html#adaptivelayerloss)

*

[MegaBatchMarginLoss](../../package_reference/sentence_transformer/losses.html#megabatchmarginloss)
*
[MultipleNegativesRankingLoss](../../package_reference/sentence_transformer/losses.html#multiplenegativesrankingloss)
*
[CachedMultipleNegativesRankingLoss](../../package_reference/sentence_transformer/losses.html#cachedmultiplenegativesrankingloss)
*
[MultipleNegativesSymmetricRankingLoss](../../package_reference/sentence_transformer/losses.html#multiplenegativessymmetricrankingloss)
*
[CachedMultipleNegativesSymmetricRankingLoss](../../package_reference/sentence_transformer/losses.html#cachedmultiplenegativessymmetricrankingloss)
    * [SoftmaxLoss](../../package_reference/sentence_transformer/losses.html#softmaxloss)
    * [TripletLoss](../../package_reference/sentence_transformer/losses.html#tripletloss)
  * [Samplers](../../package_reference/sentence_transformer/sampler.html)
    * [BatchSamplers](../../package_reference/sentence_transformer/sampler.html#batchsamplers)
*
[MultiDatasetBatchSamplers](../../package_reference/sentence_transformer/sampler.html#multidatasetbatchsamplers)
  * [Evaluation](../../package_reference/sentence_transformer/evaluation.html)
*
[BinaryClassificationEvaluator](../../package_reference/sentence_transformer/evaluation.html#binaryclassificationevaluator)
*
[EmbeddingSimilarityEvaluator](../../package_reference/sentence_transformer/evaluation.html#emb

eddingsimilarityevaluator)

*

[InformationRetrievalEvaluator](../../package_reference/sentence_transformer/evaluation.html#informationretrievalevaluator)

*

[NanoBEIREvaluator](../../package_reference/sentence_transformer/evaluation.html#nanobeirevaluator)

* [MSEEvaluator](../../package_reference/sentence_transformer/evaluation.html#mseevaluator)

*

[ParaphraseMiningEvaluator](../../package_reference/sentence_transformer/evaluation.html#paraphraseminingevaluator)

*

[RerankingEvaluator](../../package_reference/sentence_transformer/evaluation.html#rerankingevaluator)

*

[SentenceEvaluator](../../package_reference/sentence_transformer/evaluation.html#sentenceevaluator)

*

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

*

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

*

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

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

*

[ParallelSentencesDataset](../../package_reference/sentence_transformer/datasets.html#parallelsentencesdataset)
*
[SentenceLabelDataset](../../package_reference/sentence_transformer/datasets.html#sentencelabeldataset)
*
[DenoisingAutoEncoderDataset](../../package_reference/sentence_transformer/datasets.html#denoisingautoencoderdataset)
*
[NoDuplicatesDataLoader](../../package_reference/sentence_transformer/datasets.html#noduplicatesdataloader)
    * [Models](../../package_reference/sentence_transformer/models.html)
      * [Main Classes](../../package_reference/sentence_transformer/models.html#main-classes)
      * [Further Classes](../../package_reference/sentence_transformer/models.html#further-classes)
    * [quantization](../../package_reference/sentence_transformer/quantization.html)
*
[`quantize_embeddings()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.quantize_embeddings)
*
[`semantic_search_faiss()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_faiss)
*
[`semantic_search_usearch()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_usearch)
  * [Cross Encoder](../../package_reference/cross_encoder/index.html)
    * [CrossEncoder](../../package_reference/cross_encoder/cross_encoder.html)
      * [CrossEncoder](../../package_reference/cross_encoder/cross_encoder.html#id1)

* [Training Inputs](../../package_reference/cross_encoder/cross_encoder.html#training-inputs)
* [Evaluation](../../package_reference/cross_encoder/evaluation.html)
* [CEBinaryAccuracyEvaluator](../../package_reference/cross_encoder/evaluation.html#cebinaryaccuracyevaluator)
* [CEBinaryClassificationEvaluator](../../package_reference/cross_encoder/evaluation.html#cebinaryclassificationevaluator)
* [CECorrelationEvaluator](../../package_reference/cross_encoder/evaluation.html#cecorrelationevaluator)
* [CEF1Evaluator](../../package_reference/cross_encoder/evaluation.html#cef1evaluator)
* [CESoftmaxAccuracyEvaluator](../../package_reference/cross_encoder/evaluation.html#cesoftmaxaccuracyevaluator)
* [CERerankingEvaluator](../../package_reference/cross_encoder/evaluation.html#cererankingevaluator)
* [util](../../package_reference/util.html)
* [Helper Functions](../../package_reference/util.html#module-sentence_transformers.util)
* [`community_detection()`](../../package_reference/util.html#sentence_transformers.util.community_detection)
* [`http_get()`](../../package_reference/util.html#sentence_transformers.util.http_get)
* [`is_training_available()`](../../package_reference/util.html#sentence_transformers.util.is_training_available)

*
[`mine_hard_negatives()`](../../package_reference/util.html#sentence_transformers.util.mine_hard_negatives)
*
[`normalize_embeddings()`](../../package_reference/util.html#sentence_transformers.util.normalize_embeddings)
*
[`paraphrase_mining()`](../../package_reference/util.html#sentence_transformers.util.paraphrase_mining)
*
[`semantic_search()`](../../package_reference/util.html#sentence_transformers.util.semantic_search)
*
[`truncate_embeddings()`](../../package_reference/util.html#sentence_transformers.util.truncate_embeddings)
* [Model Optimization](../../package_reference/util.html#module-sentence_transformers.backend)
*
[`export_dynamic_quantized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)
*
[`export_optimized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model)
*
[`export_static_quantized_openvino_model()`](../../package_reference/util.html#sentence_transformers.backend.export_static_quantized_openvino_model)
* [Similarity Metrics](../../package_reference/util.html#module-sentence_transformers.util)
* [`cos_sim()`](../../package_reference/util.html#sentence_transformers.util.cos_sim)
* [`dot_score()`](../../package_reference/util.html#sentence_transformers.util.dot_score)

* [`euclidean_sim()`](../../package_reference/util.html#sentence_transformers.util.euclidean_sim)
* [`manhattan_sim()`](../../package_reference/util.html#sentence_transformers.util.manhattan_sim)
* [`pairwise_cos_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_cos_sim)
* [`pairwise_dot_score()`](../../package_reference/util.html#sentence_transformers.util.pairwise_dot_score)
* [`pairwise_euclidean_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_euclidean_sim)
* [`pairwise_manhattan_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_manhattan_sim)

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

* []()
* [Usage](usage.html)
* Semantic Textual Similarity
* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/docs/sentence_transformer/usage/semantic_textual_similarity.rst)

* * *

Semantic Textual Similarityï ƒ•

For Semantic Textual Similarity (STS), we want to produce embeddings for all texts involved and calculate the similarities between them. The text pairs with the highest similarity score are most semantically similar. See also the [Computing Embeddings](../../../examples/applications/computing-embeddings/README.html) documentation for more advanced details on getting embedding scores.

Documentation

1. [`SentenceTransformer`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer "sentence_transformers.SentenceTransformer")

2. [`SentenceTransformer.encode`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.encode "sentence_transformers.SentenceTransformer.encode")

3. [`SentenceTransformer.similarity`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.similarity "sentence_transformers.SentenceTransformer.similarity")

```
from sentence_transformers import SentenceTransformer

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

# Two lists of sentences
sentences1 = [
    "The new movie is awesome",
    "The cat sits outside",
    "A man is playing guitar",
]

sentences2 = [
    "The dog plays in the garden",
    "The new movie is so great",
    "A woman watches TV",
]

# Compute embeddings for both lists
embeddings1 = model.encode(sentences1)
embeddings2 = model.encode(sentences2)

# Compute cosine similarities
similarities = model.similarity(embeddings1, embeddings2)

# Output the pairs with their score
for idx_i, sentence1 in enumerate(sentences1):
    print(sentence1)
```

```
        for idx_j, sentence2 in enumerate(sentences2):
            print(f" - {sentence2: <30}: {similarities[idx_i][idx_j]:.4f}")
```

```
The new movie is awesome
- The dog plays in the garden    : 0.0543
- The new movie is so great      : 0.8939
- A woman watches TV             : -0.0502
The cat sits outside
- The dog plays in the garden    : 0.2838
- The new movie is so great      : -0.0029
- A woman watches TV             : 0.1310
A man is playing guitar
- The dog plays in the garden    : 0.2277
- The new movie is so great      : -0.0136
- A woman watches TV             : -0.0327
```

In this example, the
[`SentenceTransformer.similarity`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.similarity "sentence_transformers.SentenceTransformer.similarity") method returns a 3x3
matrix with the respective cosine similarity scores for all possible pairs
between `embeddings1` and `embeddings2`.

Similarity Calculationïƒ•

The similarity metric that is used is stored in the SentenceTransformer
instance under
[`SentenceTransformer.similarity_fn_name`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.similarity_fn_name
"sentence_transformers.SentenceTransformer.similarity_fn_name"). Valid options
are:

* `SimilarityFunction.COSINE` (a.k.a â€œcosineâ€•): Cosine Similarity (**default**)

* `SimilarityFunction.DOT_PRODUCT` (a.k.a â€œdotâ€•): Dot Product

* `SimilarityFunction.EUCLIDEAN` (a.k.a â€œeuclideanâ€•): Negative Euclidean Distance

* `SimilarityFunction.MANHATTAN` (a.k.a. â€œmanhattanâ€•): Negative Manhattan Distance

This value can be changed in a handful of ways:

1. By initializing the SentenceTransformer instance with the desired similarity function:

```
from sentence_transformers import SentenceTransformer, SimilarityFunction

model = SentenceTransformer("all-MiniLM-L6-v2", similarity_fn_name=SimilarityFunction.DOT_PRODUCT)
```

2. By setting the value directly on the SentenceTransformer instance:

```
from sentence_transformers import SentenceTransformer, SimilarityFunction

model = SentenceTransformer("all-MiniLM-L6-v2")
model.similarity_fn_name = SimilarityFunction.DOT_PRODUCT
```

3. By setting the value under the `"similarity_fn_name"` key in the `config_sentence_transformers.json` file of a saved model. When you save a Sentence Transformer model, this value will be automatically saved as well.

Sentence Transformers implements two methods to calculate the similarity between embeddings:

* [`SentenceTransformer.similarity`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.similarity "sentence_transformers.SentenceTransformer.similarity"): Calculates the similarity between all pairs of embeddings.

* `SentenceTransformer.pairwise_similarity`: Calculates the similarity between embeddings in a pairwise fashion.

```
from sentence_transformers import SentenceTransformer, SimilarityFunction
```

```
# Load a pretrained Sentence Transformer model
model = SentenceTransformer("all-MiniLM-L6-v2")

# Embed some sentences
sentences = [
    "The weather is lovely today.",
    "It's so sunny outside!",
    "He drove to the stadium.",
]
embeddings = model.encode(sentences)

similarities = model.similarity(embeddings, embeddings)
print(similarities)
# tensor([[1.0000, 0.6660, 0.1046],
#         [0.6660, 1.0000, 0.1411],
#         [0.1046, 0.1411, 1.0000]])

# Change the similarity function to Manhattan distance
model.similarity_fn_name = SimilarityFunction.MANHATTAN
print(model.similarity_fn_name)
# => "manhattan"

similarities = model.similarity(embeddings, embeddings)
print(similarities)
# tensor([[ -0.0000, -12.6269, -20.2167],
#         [-12.6269,  -0.0000, -20.1288],
#         [-20.2167, -20.1288,  -0.0000]])
```

Note

If a Sentence Transformer instance ends with a
[`Normalize`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Normalize
"sentence_transformers.models.Normalize") module, then it is sensible to
choose the â€œdotâ€• metric instead of â€œcosineâ€•.

Dot product on normalized embeddings is equivalent to cosine similarity, but
â€œcosineâ€• will re-normalize the embeddings again. As a result, the
â€œdotâ€• metric will be faster than â€œcosineâ€•.

If you want find the highest scoring pairs in a long list of sentences, have a
look at [Paraphrase Mining](../../../examples/applications/paraphrase-
mining/README.html).

[Previous](../../../examples/applications/computing-embeddings/README.html
"Computing Embeddings") [Next](../../../examples/applications/semantic-
search/README.html "Semantic Search")

* * *

[theme](https://github.com/readthedocs/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org).