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

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-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](../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/SentenceTransformer.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#onlinecontrastiveloss)
        *

[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#embeddingsimilarityevaluator)
*

[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_n

egatives)

*

[`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)

  * [](../../../index.html)
  * [Usage](usage.html)
  * Creating Custom Models

* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/docs/sentence_transformer/usage/custom_models.rst)

* * *

Creating Custom Modelsïƒ•

Structure of Sentence Transformer Modelsïƒ•

A Sentence Transformer model consists of a collection of modules
([docs](../../package_reference/sentence_transformer/models.html)) that are
executed sequentially. The most common architecture is a combination of a
[`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.
models.Transformer
"sentence_transformers.models.Transformer") module, a
[`Pooling`](../../package_reference/sentence_transformer/models.html#sentence_transformers.mode
ls.Pooling
"sentence_transformers.models.Pooling") module, and optionally, a
[`Dense`](../../package_reference/sentence_transformer/models.html#sentence_transformers.model
s.Dense
"sentence_transformers.models.Dense") module and/or a
[`Normalize`](../../package_reference/sentence_transformer/models.html#sentence_transformers.mo
dels.Normalize
"sentence_transformers.models.Normalize") module.

*

[`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.
models.Transformer "sentence_transformers.models.Transformer"): This module is responsible for processing the input text and generating contextualized embeddings.

*

[`Pooling`](../../package_reference/sentence_transformer/models.html#sentence_transformers.mode
ls.Pooling "sentence_transformers.models.Pooling"): This module reduces the dimensionality of the output from the Transformer module by aggregating the embeddings. Common pooling strategies

include mean pooling and CLS pooling.

*

[`Dense`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Dense "sentence_transformers.models.Dense"): This module contains a linear layer that post-processes the embedding output from the Pooling module.

*

[`Normalize`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Normalize "sentence_transformers.models.Normalize"): This module normalizes the embedding from the previous layer.

For example, the popular [all-MiniLM-L6-v2](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2) model can also be loaded by initializing the 3 specific modules that make up that model:

```
from sentence_transformers import models, SentenceTransformer

transformer = models.Transformer("sentence-transformers/all-MiniLM-L6-v2", max_seq_length=256)
pooling = models.Pooling(transformer.get_word_embedding_dimension(), pooling_mode="mean")
normalize = models.Normalize()

model = SentenceTransformer(modules=[transformer, pooling, normalize])
```

Saving Sentence Transformer Modelsïƒ•

Whenever a Sentence Transformer model is saved, three types of files are
generated:

* `modules.json`: This file contains a list of module names, paths, and types that are used to reconstruct the model.

* `config_sentence_transformers.json`: This file contains some configuration options of the Sentence Transformer model, including saved prompts, the model its similarity function, and the Sentence Transformer package version used by the model author.

* **Module-specific files** : Each module is saved in separate subfolders named after the module index and the model name (e.g., `1_Pooling`, `2_Normalize`), except the first module may be saved in the root directory if it has a `save_in_root` attribute set to `True`. In Sentence Transformers, this is the case for the [`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Transformer "sentence_transformers.models.Transformer") and `CLIPModel` modules. Most module folders contain a `config.json` (or `sentence_bert_config.json` for the [`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Transformer "sentence_transformers.models.Transformer") module) file that stores default values for keyword arguments passed to that Module. So, a `sentence_bert_config.json` of:

```
{
"max_seq_length": 4096,
"do_lower_case": false
```

```
}
```

means that the
[`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Transformer "sentence_transformers.models.Transformer") module will be initialized with `max_seq_length=4096` and `do_lower_case=False`.

As a result, if I call [`SentenceTransformer.save_pretrained("local-all-MiniLM-L6-v2")`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.save_pretrained "sentence_transformers.SentenceTransformer.save_pretrained") on the `model` from the previous snippet, the following files are generated:

```
local-all-MiniLM-L6-v2/
â”œâ”€â”€ 1_Pooling
â”‚   â””â”€â”€ config.json
â”œâ”€â”€ 2_Normalize
â”œâ”€â”€ README.md
â”œâ”€â”€ config.json
â”œâ”€â”€ config_sentence_transformers.json
â”œâ”€â”€ model.safetensors
â”œâ”€â”€ modules.json
â”œâ”€â”€ sentence_bert_config.json
```

â”œâ”€â”€ special_tokens_map.json

â”œâ”€â”€ tokenizer.json

â”œâ”€â”€ tokenizer_config.json

â””â”€â”€ vocab.txt

This contains a `modules.json` with these contents:


```
[
  {
    "idx": 0,
    "name": "0",
    "path": "",
    "type": "sentence_transformers.models.Transformer"
  },
  {
    "idx": 1,
    "name": "1",
    "path": "1_Pooling",
    "type": "sentence_transformers.models.Pooling"
  },
  {
    "idx": 2,
    "name": "2",
    "path": "2_Normalize",
```

```
    "type": "sentence_transformers.models.Normalize"
  }
]
```

And a `config_sentence_transformers.json` with these contents:

```
{
  "__version__": {
    "sentence_transformers": "3.0.1",
    "transformers": "4.43.4",
    "pytorch": "2.5.0"
  },
  "prompts": {},
  "default_prompt_name": null,
  "similarity_fn_name": null
}
```

Additionally, the `1_Pooling` directory contains the configuration file for
the
[`Pooling`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Pooling
"sentence_transformers.models.Pooling") module, while the `2_Normalize`
directory is empty because the

[`Normalize`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Normalize "sentence_transformers.models.Normalize") module does not require any configuration. The `sentence_bert_config.json` file contains the configuration of the [`Transformer`](../../package_reference/sentence_transformer/models.html#sentence_transformers.models.Transformer "sentence_transformers.models.Transformer") module, and this module also saved a lot of files related to the tokenizer and the model itself in the root directory.

Loading Sentence Transformer Modelsïƒ•

To load a Sentence Transformer model from a saved model directory, the `modules.json` is read to determine the modules that make up the model. Each module is initialized with the configuration stored in the corresponding module directory, after which the SentenceTransformer class is instantiated with the loaded modules.

Sentence Transformer Model from a Transformers Modelïƒ•

When you initialize a Sentence Transformer model with a pure Transformers model (e.g., BERT, RoBERTa, DistilBERT, T5), Sentence Transformers creates a Transformer module and a Mean Pooling module by default. This provides a simple way to leverage pre-trained language models for sentence embeddings.

To be specific, these two snippets are identical:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("bert-base-uncased")



from sentence_transformers import models, SentenceTransformer

transformer = models.Transformer("bert-base-uncased")
pooling = models.Pooling(transformer.get_word_embedding_dimension(), pooling_mode="mean")
model = SentenceTransformer(modules=[transformer, pooling])
```

Advanced: Custom Modulesïƒ•

To create custom Sentence Transformer models, you can implement your own modules by subclassing PyTorchâ€™s [`torch.nn.Module`](https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module "\(in PyTorch v2.5\)") class and implementing these methods:

* A [`torch.nn.Module.forward()`](https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module.forward "\(in PyTorch v2.5\)") method that accepts a `features` dictionary with keys like `input_ids`, `attention_mask`, `token_type_ids`, `token_embeddings`, and `sentence_embedding`,

depending on where the module is in the model pipeline.

* A `save` method that accepts a `save_dir` argument and saves the moduleâ€™s configuration to that directory.

* A `load` static method that accepts a `load_dir` argument and initializes the Module given the moduleâ€™s configuration from that directory.

* (If 1st module) A `get_max_seq_length` method that returns the maximum sequence length the module can process. Only required if the module processes input text.

* (If 1st module) A `tokenize` method that accepts a list of inputs and returns a dictionary with keys like `input_ids`, `attention_mask`, `token_type_ids`, `pixel_values`, etc. This dictionary will be passed along to the moduleâ€™s `forward` method.

* (Optional) A `get_sentence_embedding_dimension` method that returns the dimensionality of the sentence embeddings produced by the module. Only required if the module generated the embeddings or updates the embeddingsâ€™ dimensionality.

* (Optional) A `get_config_dict` method that returns a dictionary with the moduleâ€™s configuration. This method can be used to save the moduleâ€™s configuration to disk and to save the module config in a model card.

For example, we can create a custom pooling method by implementing a custom
Module.

```
# decay_pooling.py

import json
import os
import torch
import torch.nn as nn

class DecayMeanPooling(nn.Module):
    def __init__(self, dimension: int, decay: float = 0.95) -> None:
        super(DecayMeanPooling, self).__init__()
        self.dimension = dimension
        self.decay = decay

    def forward(self, features: dict[str, torch.Tensor], **kwargs) -> dict    [str, torch.Tensor]:
        token_embeddings = features["token_embeddings"]
        attention_mask = features["attention_mask"].unsqueeze(-1)

        # Apply the attention mask to filter away padding tokens
        token_embeddings = token_embeddings * attention_mask
        # Calculate mean of token embeddings
        sentence_embeddings = token_embeddings.sum(1) / attention_mask.sum(1)
        # Apply exponential decay
            importance_per_dim  =  self.decay  **  torch.arange(sentence_embeddings.      size(1),
device=sentence_embeddings.device)
        features["sentence_embedding"] = sentence_embeddings *    importance_per_dim
        return features
```

```
    def get_config_dict(self) -> dict[str, float]:
        return {"dimension": self.dimension, "decay": self.decay}

    def get_sentence_embedding_dimension(self) -> int:
        return self.dimension

    def save(self, save_dir: str, **kwargs) -> None:
        with open(os.path.join(save_dir, "config.json"), "w") as fOut:
            json.dump(self.get_config_dict(), fOut, indent=4)

    def load(load_dir: str, **kwargs) -> "DecayMeanPooling":
        with open(os.path.join(load_dir, "config.json")) as fIn:
            config = json.load(fIn)

        return DecayMeanPooling(**config)
```

Note

Adding `**kwargs` to the `__init__`, `forward`, `save`, `load`, and `tokenize` methods is recommended to ensure that the methods are compatible with future updates to the Sentence Transformers library.

Note

If your module is the first module, then you can set `save_in_root = True` in

the moduleâ€™s class definition if you want your module to be saved in the root directory upon save. Do note that unlike the subdirectories, the root directory is not downloaded from the Hugging Face Hub before loading the module. As a result, the module should first check if the required files exist locally and otherwise use [`huggingface_hub.hf_hub_download()`](https://huggingface.co/docs/huggingface_hub/main/en/package_reference/file_download#huggingface_hub.hf_hub_download "\(in huggingface_hub vmain\)") to download them.

This can now be used as a module in a Sentence Transformer model:

```
    from sentence_transformers import models, SentenceTransformer
    from decay_pooling import DecayMeanPooling

    transformer = models.Transformer("bert-base-uncased", max_seq_length=256)
        decay_mean_pooling = DecayMeanPooling(transformer.get_word_embedding_dimension(), decay=0.99)
    normalize = models.Normalize()

    model = SentenceTransformer(modules=[transformer, decay_mean_pooling, normalize])
    print(model)
    """
    SentenceTransformer(
        (0): Transformer({'max_seq_length': 256, 'do_lower_case': False}) with Transformer model: BertModel
```

```
    (1): DecayMeanPooling()
    (2): Normalize()
)
"""

texts = [
    "Hello, World!",
    "The quick brown fox jumps over the lazy dog.",
    "I am a sentence that is used for testing purposes.",
    "This is a test sentence.",
    "This is another test sentence.",
]
embeddings = model.encode(texts)
print(embeddings.shape)
# [5, 384]
```

You can save this model with [`SentenceTransformer.save_pretrained`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.save_pretrained "sentence_transformers.SentenceTransformer.save_pretrained"), resulting in a `modules.json` of:

```
[
  {
```

```
    "idx": 0,
    "name": "0",
    "path": "",
    "type": "sentence_transformers.models.Transformer"
  },
  {
    "idx": 1,
    "name": "1",
    "path": "1_DecayMeanPooling",
    "type": "decay_pooling.DecayMeanPooling"
  },
  {
    "idx": 2,
    "name": "2",
    "path": "2_Normalize",
    "type": "sentence_transformers.models.Normalize"
  }
]
```


To ensure that `decay_pooling.DecayMeanPooling` can be imported, you should copy over the `decay_pooling.py` file to the directory where you saved the model. If you push the model to the [Hugging Face Hub](https://huggingface.co/models), then you should also upload the `decay_pooling.py` file to the modelâ€™s repository. Then, everyone can use your custom module by calling [`SentenceTransformer("your-username/your-model-id",

trust_remote_code=True)`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer "sentence_transformers.SentenceTransformer").

Note

Using a custom module with remote code stored on the Hugging Face Hub requires that your users specify `trust_remote_code` as `True` when loading the model. This is a security measure to prevent remote code execution attacks.

If you have your models and custom modelling code on the Hugging Face Hub, then it might make sense to separate your custom modules into a separate repository. This way, you only have to maintain one implementation of your custom module, and you can reuse it across multiple models. You can do this by updating the `type` in `modules.json` file to include the path to the repository where the custom module is stored like `{repository_id}--{dot_path_to_module}`. For example, if the `decay_pooling.py` file is stored in a repository called `my-user/my-model-implementation` and the module is called `DecayMeanPooling`, then the `modules.json` file may look like this:

```
[
  {
    "idx": 0,
    "name": "0",
```

```
    "path": "",
    "type": "sentence_transformers.models.Transformer"
  },
  {
    "idx": 1,
    "name": "1",
    "path": "1_DecayMeanPooling",
    "type": "my-user/my-model-implementation--decay_pooling.DecayMeanPooling"
  },
  {
    "idx": 2,
    "name": "2",
    "path": "2_Normalize",
    "type": "sentence_transformers.models.Normalize"
  }
]
```

Advanced: Keyword argument passthrough in Custom Modulesïƒ•

If you want your users to be able to specify custom keyword arguments via the [`SentenceTransformer.encode`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.encode "sentence_transformers.SentenceTransformer.encode") method, then you can add their names to the `modules.json` file. For example, if my module should behave differently if your users specify a `task_type` keyword argument, then your `modules.json` might look like:

```
[
  {
    "idx": 0,
    "name": "0",
    "path": "",
    "type": "custom_transformer.CustomTransformer",
    "kwargs": ["task_type"]
  },
  {
    "idx": 1,
    "name": "1",
    "path": "1_Pooling",
    "type": "sentence_transformers.models.Pooling"
  },
  {
    "idx": 2,
    "name": "2",
    "path": "2_Normalize",
    "type": "sentence_transformers.models.Normalize"
  }
]
```


Then, you can access the `task_type` keyword argument in the `forward` method

of your custom module:

```
from sentence_transformers.models import Transformer

class CustomTransformer(Transformer):
    def forward(self, features: dict[str, torch.Tensor], task_type: Optional[str] = None) -> dict[str, torch.Tensor]:
        if task_type == "default":
            # Do something
        else:
            # Do something else
        return features
```

This way, users can specify the `task_type` keyword argument when calling [`SentenceTransformer.encode`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.encode "sentence_transformers.SentenceTransformer.encode"):

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("your-username/your-model-id", trust_remote_code=True)
texts = [...]
```

```
model.encode(texts, task_type="default")
```

[Previous](efficiency.html "Speeding up Inference") [Next](../pretrained_models.html "Pretrained Models")

* * *