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

Getting Started

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

Sentence Transformer

* [Usage](../../../docs/sentence_transformer/usage/usage.html)
  * Computing Embeddings
    * Initializing a Sentence Transformer Model
    * Calculating Embeddings
    * Prompt Templates
    * Input Sequence Length
    * Multi-Process / Multi-GPU Encoding
  * [Semantic Textual Similarity](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html)

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

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

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

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

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

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

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

Cross Encoder

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

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

Package Reference

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

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

[OnlineContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#onlinecontrastiveloss)

*

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

*

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

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

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

*

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

*

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

*

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

*

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

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

*

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

*

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

[BatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#batchsamplers)

*

[MultiDatasetBatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#multidatasetbatchsamplers)

* [Evaluation](../../../docs/package_reference/sentence_transformer/evaluation.html)

*

[BinaryClassificationEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#binaryclassificationevaluator)

*

[EmbeddingSimilarityEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#embeddingsimilarityevaluator)

*

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

*

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

*

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

*

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

*

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

*

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

*

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

*

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

*

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

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

*

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

*

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

*

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

*

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

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

* [Main

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

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

* [quantization](../../../docs/package_reference/sentence_transformer/quantization.html)

* [`quantize_embeddings()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.quantize_embeddings)

* [`semantic_search_faiss()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_faiss)

* [`semantic_search_usearch()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_usearch)

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

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

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

* [Training Inputs](../../../docs/package_reference/cross_encoder/cross_encoder.html#training-inputs)

* [Evaluation](../../../docs/package_reference/cross_encoder/evaluation.html)

* [CEBinaryAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryaccuracyevaluator)

* [CEBinaryClassificationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryclassificationevaluator)

* [CECorrelationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cecorrelatio

nevaluator)
    * [CEF1Evaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cef1evaluator)
*
[CESoftmaxAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cesoftmaxaccuracyevaluator)
*
[CERerankingEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cererankingevaluator)
  * [util](../../../docs/package_reference/util.html)
    * [Helper Functions](../../../docs/package_reference/util.html#module-sentence_transformers.util)
*
[`community_detection()`](../../../docs/package_reference/util.html#sentence_transformers.util.community_detection)
    * [`http_get()`](../../../docs/package_reference/util.html#sentence_transformers.util.http_get)
*
[`is_training_available()`](../../../docs/package_reference/util.html#sentence_transformers.util.is_training_available)
*
[`mine_hard_negatives()`](../../../docs/package_reference/util.html#sentence_transformers.util.mine_hard_negatives)
*
[`normalize_embeddings()`](../../../docs/package_reference/util.html#sentence_transformers.util.normalize_embeddings)
*
[`paraphrase_mining()`](../../../docs/package_reference/util.html#sentence_transformers.util.paraphrase_mining)
*

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

*

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

* [Model Optimization](../../../docs/package_reference/util.html#module-sentence_transformers.backend)

*

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

*

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

*

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

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

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

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

*

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

*

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

*

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

cos_sim)

*

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

*

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

*

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

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

  * [](../../../index.html)
  * [Usage](../../../docs/sentence_transformer/usage/usage.html)
  * Computing Embeddings

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

* * *

Computing Embeddingsï*f*•

Once you have [installed](installation.md) Sentence Transformers, you can easily use Sentence Transformer models:

Documentation

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

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

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

```
from sentence_transformers import SentenceTransformer

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

# The sentences to encode
sentences = [
    "The weather is lovely today.",
```

```
    "It's so sunny outside!",
    "He drove to the stadium.",
]

# 2. Calculate embeddings by calling model.encode()
embeddings = model.encode(sentences)
print(embeddings.shape)
# [3, 384]

# 3. Calculate the embedding similarities
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]])
```

Note

Even though we talk about sentence embeddings, you can use Sentence Transformers for shorter phrases as well as for longer texts with multiple sentences. See Input Sequence Length for notes on embeddings for longer texts.

Initializing a Sentence Transformer Modelïƒ•

The first step is to load a pretrained Sentence Transformer model. You can use any of the models from the [Pretrained

Models](../docs/sentence_transformer/pretrained_models.html) or a local model.

See also

[`SentenceTransformer`](../../../docs/package_reference/sentence_transformer/SentenceTransforme

r.html#sentence_transformers.SentenceTransformer

"sentence_transformers.SentenceTransformer") for information on parameters.

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-mpnet-base-v2")
# Alternatively, you can pass a path to a local model directory:
model = SentenceTransformer("output/models/mpnet-base-finetuned-all-nli")
```

The model will automatically be placed on the most performant available

device, e.g. `cuda` or `mps` if available. You can also specify the device

explicitly:

```
model = SentenceTransformer("all-mpnet-base-v2", device="cuda")
```

Calculating Embeddingsïƒ•

The method to calculate embeddings is `SentenceTransformer.encode`.

Prompt Templatesïƒ•

Some models require using specific text _prompts_ to achieve optimal
performance. For example, with
[intfloat/multilingual-e5-large](https://huggingface.co/intfloat/multilingual-e5-large)
you should prefix all queries with `"query: "` and all passages with
`"passage: "`. Another example is [BAAI/bge-large-
en-v1.5](https://huggingface.co/BAAI/bge-large-en-v1.5), which performs best
for retrieval when the input texts are prefixed with `"Represent this sentence
for searching relevant passages: "`.

Sentence Transformer models can be initialized with `prompts` and
`default_prompt_name` parameters:

* `prompts` is an optional argument that accepts a dictionary of prompts with prompt names to prompt texts. The prompt will be prepended to the input text during inference. For example:

```
    model = SentenceTransformer(
    "intfloat/multilingual-e5-large",
    prompts={
        "classification": "Classify the following text: ",
        "retrieval": "Retrieve semantically similar text: ",
        "clustering": "Identify the topic or theme based on the text: ",
    },
)
# or
```

```
model.prompts = {
    "classification": "Classify the following text: ",
    "retrieval": "Retrieve semantically similar text: ",
    "clustering": "Identify the topic or theme based on the text: ",
}
```

* `default_prompt_name` is an optional argument that determines the default prompt to be used. It has to correspond with a prompt name from `prompts`. If `None`, then no prompt is used by default. For example:

```
    model = SentenceTransformer(
    "intfloat/multilingual-e5-large",
    prompts={
        "classification": "Classify the following text: ",
        "retrieval": "Retrieve semantically similar text: ",
        "clustering": "Identify the topic or theme based on the text: ",
    },
    default_prompt_name="retrieval",
)
# or
model.default_prompt_name="retrieval"
```

Both of these parameters can also be specified in the `config_sentence_transformers.json` file of a saved model. That way, you wonâ€™t have to specify these options manually when loading. When you save a

Sentence Transformer model, these options will be automatically saved as well.

During inference, prompts can be applied in a few different ways. All of these scenarios result in identical texts being embedded:

1. Explicitly using the `prompt` option in `SentenceTransformer.encode`:

    embeddings = model.encode("How to bake a strawberry cake", prompt="Retrieve semantically similar text: ")

2. Explicitly using the `prompt_name` option in `SentenceTransformer.encode` by relying on the prompts loaded from a) initialization or b) the model config:

    embeddings = model.encode("How to bake a strawberry cake", prompt_name="retrieval")

3. If `prompt` nor `prompt_name` are specified in `SentenceTransformer.encode`, then the prompt specified by `default_prompt_name` will be applied. If it is `None`, then no prompt will be applied:

    embeddings = model.encode("How to bake a strawberry cake")

Input Sequence Lengthïƒ•

For transformer models like BERT, RoBERTa, DistilBERT etc., the runtime and memory requirement grows quadratic with the input length. This limits

transformers to inputs of certain lengths. A common value for BERT-based models are 512 tokens, which corresponds to about 300-400 words (for English).

Each model has a maximum sequence length under `model.max_seq_length`, which is the maximal number of tokens that can be processed. Longer texts will be truncated to the first `model.max_seq_length` tokens:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")
print("Max Sequence Length:", model.max_seq_length)
# => Max Sequence Length: 256

# Change the length to 200
model.max_seq_length = 200

print("Max Sequence Length:", model.max_seq_length)
# => Max Sequence Length: 200
```

Note

You cannot increase the length higher than what is maximally supported by the respective transformer model. Also note that if a model was trained on short texts, the representations for long texts might not be that good.

Multi-Process / Multi-GPU Encodingïƒ•

You can encode input texts with more than one GPU (or with multiple processes
on a CPU machine). For an example, see:
[computing_embeddings_multi_gpu.py](https://github.com/UKPLab/sentence-
transformers/blob/master/examples/applications/computing-
embeddings/computing_embeddings_multi_gpu.py).

The relevant method is
[`start_multi_process_pool()`](../../../docs/package_reference/sentence_transformer/SentenceTransf
ormer.html#sentence_transformers.SentenceTransformer.start_multi_process_pool
"sentence_transformers.SentenceTransformer.start_multi_process_pool"), which
starts multiple processes that are used for encoding.

[Previous](../../../docs/sentence_transformer/usage/usage.html "Usage") [Next
](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html
"Semantic Textual Similarity")

* * *

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).