[![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/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](usage/semantic_textual_similarity.html)
* [Similarity Calculation](usage/semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../../examples/applications/semantic-search/README.html)
* [Background](../../examples/applications/semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../../examples/applications/semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../../examples/applications/semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../../examples/applications/semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../../examples/applications/semantic-search/README.html#speed-optimization)
* [Elasticsearch](../../examples/applications/semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../../examples/applications/semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../../examples/applications/semantic-search/README.html#retrieve-re-rank)
* [Examples](../../examples/applications/semantic-search/README.html#examples)

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

Mining](../../examples/applications/parallel-sentence-mining/README.html#margin-based-mining)
    * [Examples](../../examples/applications/parallel-sentence-mining/README.html#examples)
  * [Image Search](../../examples/applications/image-search/README.html)
    * [Installation](../../examples/applications/image-search/README.html#installation)
    * [Usage](../../examples/applications/image-search/README.html#usage)
    * [Examples](../../examples/applications/image-search/README.html#examples)
  * [Embedding Quantization](../../examples/applications/embedding-quantization/README.html)
    * [Binary Quantization](../../examples/applications/embedding-quantization/README.html#binary-quantization)
    * [Scalar (int8) Quantization](../../examples/applications/embedding-quantization/README.html#scalar-int8-quantization)
    * [Additional extensions](../../examples/applications/embedding-quantization/README.html#additional-extensions)
    * [Demo](../../examples/applications/embedding-quantization/README.html#demo)
    * [Try it yourself](../../examples/applications/embedding-quantization/README.html#try-it-yourself)
  * [Speeding up Inference](usage/efficiency.html)
    * [PyTorch](usage/efficiency.html#pytorch)
    * [ONNX](usage/efficiency.html#onnx)
    * [OpenVINO](usage/efficiency.html#openvino)
    * [Benchmarks](usage/efficiency.html#benchmarks)
  * [Creating Custom Models](usage/custom_models.html)
    * [Structure of Sentence Transformer Models](usage/custom_models.html#structure-of-sentence-transformer-models)

* [Sentence Transformer Model from a Transformers Model](usage/custom_models.html#sentence-transformer-model-from-a-transformers-model)
* Pretrained Models
  * Original Models
  * Semantic Search Models
    * Multi-QA Models
    * MSMARCO Passage Models
  * Multilingual Models
    * Semantic Similarity Models
    * Bitext Mining
  * Image & Text-Models
  * INSTRUCTOR models
  * 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)

* [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#noduplicatesd

ataloader)
    * [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#cecorrelationevaluat

or)
    * [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)

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

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

* * *

Pretrained Modelsïƒ•

We provide various pre-trained Sentence Transformers models via our Sentence Transformers Hugging Face organization. Additionally, over 6,000 community Sentence Transformers models have been publicly released on the Hugging Face Hub. All models can be found here:

* **Original models** : [Sentence Transformers Hugging Face organization](https://huggingface.co/models?library=sentence-transformers&author=sentence-transformers).

* **Community models** : [All Sentence Transformer models on Hugging Face](https://huggingface.co/models?library=sentence-transformers).

Each of these models can be easily downloaded and used like so:

Original Models

For the original models from the [Sentence Transformers Hugging Face organization](https://huggingface.co/models?library=sentence-transformers&author=sentence-transformers), it is not necessary to include the model author or organization prefix. For example, this snippet loads [sentence-transformers/all-mpnet-base-v2](https://huggingface.co/sentence-transformers/all-mpnet-base-v2).

```
from sentence_transformers import SentenceTransformer

# Load https://huggingface.co/sentence-transformers/all-mpnet-base-v2
model = SentenceTransformer("all-mpnet-base-v2")
embeddings = model.encode([
    "The weather is lovely today.",
    "It's so sunny outside!",
    "He drove to the stadium.",
])
similarities = model.similarity(embeddings, embeddings)
```

Note

Consider using the [Massive Textual Embedding Benchmark leaderboard](https://huggingface.co/spaces/mteb/leaderboard) as an inspiration of strong Sentence Transformer models. Be wary:

* **Model sizes** : it is recommended to filter away the large models that might not be feasible without excessive hardware.

* **Experimentation is key** : models that perform well on the leaderboard do not necessarily do well on your tasks, it is **crucial** to experiment with various promising models.

Tip

Read [Sentence Transformer > Usage > Speeding up Inference](./usage/efficiency.html) for tips on how to speed up inference of models by up to 2x-3x.

Original Modelsïƒ•

The following table provides an overview of a selection of our models. They have been extensively evaluated for their quality to embedded sentences (Performance Sentence Embeddings) and to embedded search queries & paragraphs (Performance Semantic Search).

The **all-** * models were trained on all available training data (more than 1 billion training pairs) and are designed as **general purpose** models. The [**all-mpnet-base-v2**](https://huggingface.co/sentence-transformers/all-

mpnet-base-v2) model provides the best quality, while [**all-MiniLM-L6-v2**](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2) is 5 times faster and still offers good quality. Toggle _All models_ to see all evaluated original models.

* * *

Semantic Search Modelsïƒ•

The following models have been specifically trained for **Semantic Search** : Given a question / search query, these models are able to find relevant text passages. For more details, see [Usage > Semantic Search](../../examples/applications/semantic-search/README.html).

Documentation

1. [multi-qa-mpnet-base-cos-v1](https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-cos-v1)

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

3. [`SentenceTransformer.encode`](../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.encode

"sentence_transformers.SentenceTransformer.encode")

4.

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

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("multi-qa-mpnet-base-cos-v1")

query_embedding = model.encode("How big is London")
passage_embeddings = model.encode([
    "London is known for its financial district",
    "London has 9,787,426 inhabitants at the 2011 census",
    "The United Kingdom is the fourth largest exporter of goods in the world",
])

similarity = model.similarity(query_embedding, passage_embeddings)
# => tensor([[0.4659, 0.6142, 0.2697]])
```

Multi-QA Modelsïƒ•

The following models have been trained on [215M question-answer

pairs](https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-dot-v1#training) from various sources and domains, including StackExchange, Yahoo Answers, Google & Bing search queries and many more.

These model perform well across many search tasks and domains.

These models were tuned to be used with the dot-product similarity score:

Model	Performance Semantic Search (6 Datasets)	Queries (GPU / CPU) per sec.
[multi-qa-mpnet-base-dot-v1](https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1)	57.60	4,000 / 170
[multi-qa-distilbert-dot-v1](https://huggingface.co/sentence-transformers/multi-qa-distilbert-dot-v1)	52.51	7,000 / 350
[multi-qa-MiniLM-L6-dot-v1](https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-dot-v1)	49.19	18,000 / 750

These models produce normalized vectors of length 1, which can be used with dot-product, cosine-similarity and Euclidean distance as the similarity functions:

Model	Performance Semantic Search (6 Datasets)	Queries (GPU / CPU) per sec.
[multi-qa-mpnet-base-cos-v1](https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-cos-v1)	57.46	4,000 / 170
[multi-qa-distilbert-cos-v1](https://huggingface.co/sentence-transformers/multi-qa-distilbert-cos-v1)	52.83	7,000 / 350
[multi-qa-MiniLM-L6-cos-v1](https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v

1) | 51.83 | 18,000 / 750

MSMARCO Passage Modelsïƒ•

The following models have been trained on the [MSMARCO Passage Ranking Dataset](https://github.com/microsoft/MSMARCO-Passage-Ranking), which contains 500k real queries from Bing search together with the relevant passages from various web sources. Given the diversity of the MSMARCO dataset, models also perform well on other domains.

These models were tuned to be used with the dot-product similarity score:

Model	MSMARCO MRR@10 dev set	Performance Semantic Search (6 Datasets)	Queries (GPU / CPU) per sec.
[msmarco-bert-base-dot-v5](https://huggingface.co/sentence-transformers/msmarco-bert-base-dot-v5)	38.08	52.11	4,000 / 170
[msmarco-distilbert-dot-v5](https://huggingface.co/sentence-transformers/msmarco-distilbert-dot-v5)	37.25	49.47	7,000 / 350
[msmarco-distilbert-base-tas-b](https://huggingface.co/sentence-transformers/msmarco-distilbert-base-tas-b)	34.43	49.25	7,000 / 350

These models produce normalized vectors of length 1, which can be used with dot-product, cosine-similarity and Euclidean distance as the similarity functions:

Model | MSMARCO MRR@10 dev set | Performance Semantic Search (6 Datasets) | Queries (GPU

/ CPU) per sec.

---|---|---|---

[msmarco-distilbert-cos-v5](https://huggingface.co/sentence-transformers/msmarco-distilbert-cos-v5) | 33.79 | 44.98 | 7,000 / 350

[msmarco-MiniLM-L12-cos-v5](https://huggingface.co/sentence-transformers/msmarco-MiniLM-L12-cos-v5) | 32.75 | 43.89 | 11,000 / 400

[msmarco-MiniLM-L6-cos-v5](https://huggingface.co/sentence-transformers/msmarco-MiniLM-L6-cos-v5) | 32.27 | 42.16 | 18,000 / 750

[MSMARCO Models - More details](../pretrained-models/msmarco-v5.html)

* * *

Multilingual Modelsïƒ•

The following models similar embeddings for the same texts in different languages. You do not need to specify the input language. Details are in our publication [Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation](https://arxiv.org/abs/2004.09813). We used the following 50+ languages: ar, bg, ca, cs, da, de, el, en, es, et, fa, fi, fr, fr-ca, gl, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, ku, lt, lv, mk, mn, mr, ms, my, nb, nl, pl, pt, pt-br, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh-cn, zh-tw.

Semantic Similarity Modelsïƒ•

These models find semantically similar sentences within one language or across

languages:

*

[distiluse-base-multilingual-cased-v1](https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v1) : Multilingual knowledge distilled version of [multilingual Universal Sentence Encoder](https://arxiv.org/abs/1907.04307). Supports 15 languages: Arabic, Chinese, Dutch, English, French, German, Italian, Korean, Polish, Portuguese, Russian, Spanish, Turkish.

*

[distiluse-base-multilingual-cased-v2](https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2) : Multilingual knowledge distilled version of [multilingual Universal Sentence Encoder](https://arxiv.org/abs/1907.04307). This version supports 50+ languages, but performs a bit weaker than the v1 model.

*

[paraphrase-multilingual-MiniLM-L12-v2](https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2) \- Multilingual version of [paraphrase-MiniLM-L12-v2](https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L12-v2), trained on parallel data for 50+ languages.

*

[paraphrase-multilingual-mpnet-base-v2](https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2) \- Multilingual version of [paraphrase-mpnet-base-v2](https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2), trained on parallel data for 50+ languages.

Bitext Miningï•

Bitext mining describes the process of finding translated sentence pairs in two languages. If this is your use-case, the following model gives the best performance:

* **[LaBSE](https://huggingface.co/sentence-transformers/LaBSE)** \-[LaBSE](https://arxiv.org/abs/2007.01852) Model. Supports 109 languages. Works well for finding translation pairs in multiple languages. As detailed [here](https://arxiv.org/abs/2004.09813), LaBSE works less well for assessing the similarity of sentence pairs that are not translations of each other.

Extending a model to new languages is easy by following [Training Examples > Multilingual Models](../../examples/training/multilingual/README.html).

Image & Text-Modelsïƒ•

The following models can embed images and text into a joint vector space. See [Usage > Image Search](../../examples/applications/image-search/README.html) for more details how to use for text2image-search, image2image-search, image clustering, and zero-shot image classification.

The following models are available with their respective Top 1 accuracy on zero-shot ImageNet validation dataset.

Model	Top 1 Performance
[clip-ViT-L-14](https://huggingface.co/sentence-transformers/clip-ViT-L-14)	75.4
[clip-ViT-B-16](https://huggingface.co/sentence-transformers/clip-ViT-B-16)	68.1

[clip-ViT-B-32](https://huggingface.co/sentence-transformers/clip-ViT-B-32) | 63.3

We further provide this multilingual text-image model:

*

[clip-ViT-B-32-multilingual-v1](https://huggingface.co/sentence-transformers/clip-ViT-B-32-multilingual-v1) \- Multilingual text encoder for the [clip-ViT-B-32](https://huggingface.co/sentence-transformers/clip-ViT-B-32) model using [Multilingual Knowledge Distillation](https://arxiv.org/abs/2004.09813). This model can encode text in 50+ languages to match the image vectors from the [clip-ViT-B-32](https://huggingface.co/sentence-transformers/clip-ViT-B-32) model.

INSTRUCTOR modelsïƒ•

Some INSTRUCTOR models, such as [hkunlp/instructor-large](https://huggingface.co/hkunlp/instructor-large), are natively supported in Sentence Transformers. These models are special, as they are trained with instructions in mind. Notably, the primary difference between normal Sentence Transformer models and Instructor models is that the latter do not include the instructions themselves in the pooling step.

The following models work out of the box:

* [hkunlp/instructor-base](https://huggingface.co/hkunlp/instructor-base)

* [hkunlp/instructor-large](https://huggingface.co/hkunlp/instructor-large)

* [hkunlp/instructor-xl](https://huggingface.co/hkunlp/instructor-xl)

You can use these models like so:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("hkunlp/instructor-large")
embeddings = model.encode(
    [
        "Dynamical Scalar Degree of Freedom in Horava-Lifshitz Gravity",
        "Comparison of Atmospheric Neutrino Flux Calculations at Low Energies",
        "Fermion Bags in the Massive Gross-Neveu Model",
        "QCD corrections to Associated t-tbar-H production at the Tevatron",
    ],
    prompt="Represent the Medicine sentence for clustering: ",
)
print(embeddings.shape)
# => (4, 768)
```

For example, for information retrieval:

```
from sentence_transformers import SentenceTransformer
```

```
from sentence_transformers.util import cos_sim


model = SentenceTransformer("hkunlp/instructor-large")
query = "where is the food stored in a yam plant"
query_instruction = (
    "Represent the Wikipedia question for retrieving supporting documents: "
)
corpus = [
    'Yams are perennial herbaceous vines native to Africa, Asia, and the Americas and cultivated for the consumption of their starchy tubers in many temperate and tropical regions. The tubers themselves, also called "yams", come in a variety of forms owing to numerous cultivars and related species.',
    "The disparate impact theory is especially controversial under the Fair Housing Act because the Act regulates many activities relating to housing, insurance, and mortgage loansÃ¢â‚¬â€•and some scholars have argued that the theory's use under the Fair Housing Act, combined with extensions of the Community Reinvestment Act, contributed to rise of sub-prime lending and the crash of the U.S. housing market and ensuing global economic recession",
    "Disparate impact in United States labor law refers to practices in employment, housing, and other areas that adversely affect one group of people of a protected characteristic more than another, even though rules applied by employers or landlords are formally neutral. Although the protected classes vary by statute, most federal civil rights laws protect based on race, color, religion, national origin, and sex as protected traits, and some laws include disability status and other traits as well.",
]
corpus_instruction = "Represent the Wikipedia document for retrieval: "


query_embedding = model.encode(query, prompt=query_instruction)
```

```
    corpus_embeddings = model.encode(corpus, prompt=corpus_instruction)
    similarities = cos_sim(query_embedding, corpus_embeddings)
    print(similarities)
    # => tensor([[0.8835, 0.7037, 0.6970]])
```

All other Instructor models either 1) will not load as they refer to `InstructorEmbedding` in their `modules.json` or 2) require calling `model.set_pooling_include_prompt(include_prompt=False)` after loading.

Scientific Similarity Modelsïƒ•

[SPECTER](https://arxiv.org/abs/2004.07180) is a model trained on scientific citations and can be used to estimate the similarity of two publications. We can use it to find similar papers.

* **[allenai-specter](https://huggingface.co/sentence-transformers/allenai-specter)** \- [Semantic Search Python Example](https://github.com/UKPLab/sentence-transformers/tree/master/docs/sentence_transformer/../../examples/applications/semantic-search/semantic_search_publications.py) / [Semantic Search Colab Example](https://colab.research.google.com/drive/12hfBveGHRsxhPIUMmJYrll2lFU4fOX06)

[Previous](usage/custom_models.html "Creating Custom Models") [Next](training_overview.html "Training Overview")

* * *

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).