[![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
  * Sentence Transformer
  * Cross Encoder
  * Next Steps

Sentence Transformer

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

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)

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

* [Pretrained Models](../examples/training/quora_duplicate_questions/README.html#pretrained-models)
  * [MS MARCO](../examples/training/ms_marco/README.html)
    * [Bi-Encoder](../examples/training/ms_marco/README.html#bi-encoder)
  * [Matryoshka Embeddings](../examples/training/matryoshka/README.html)
    * [Use Cases](../examples/training/matryoshka/README.html#use-cases)
    * [Results](../examples/training/matryoshka/README.html#results)
    * [Training](../examples/training/matryoshka/README.html#training)
    * [Inference](../examples/training/matryoshka/README.html#inference)
    * [Code Examples](../examples/training/matryoshka/README.html#code-examples)
  * [Adaptive Layers](../examples/training/adaptive_layer/README.html)
    * [Use Cases](../examples/training/adaptive_layer/README.html#use-cases)
    * [Results](../examples/training/adaptive_layer/README.html#results)
    * [Training](../examples/training/adaptive_layer/README.html#training)
    * [Inference](../examples/training/adaptive_layer/README.html#inference)
    * [Code Examples](../examples/training/adaptive_layer/README.html#code-examples)
  * [Multilingual Models](../examples/training/multilingual/README.html)
    * [Extend your own models](../examples/training/multilingual/README.html#extend-your-own-models)
    * [Training](../examples/training/multilingual/README.html#training)
    * [Datasets](../examples/training/multilingual/README.html#datasets)
      * [Sources for Training Data](../examples/training/multilingual/README.html#sources-for-training-data)
    * [Evaluation](../examples/training/multilingual/README.html#evaluation)
      * [Available Pre-trained Models](../examples/training/multilingual/README.html#available-pre-trained-models)
    * [Usage](../examples/training/multilingual/README.html#usage)

    * [Performance](../examples/training/multilingual/README.html#performance)
    * [Citation](../examples/training/multilingual/README.html#citation)
  * [Model Distillation](../examples/training/distillation/README.html)
    * [Knowledge Distillation](../examples/training/distillation/README.html#knowledge-distillation)
    * [Speed - Performance Trade-Off](../examples/training/distillation/README.html#speed-performance-trade-off)
    * [Dimensionality Reduction](../examples/training/distillation/README.html#dimensionality-reduction)
    * [Quantization](../examples/training/distillation/README.html#quantization)
  * [Augmented SBERT](../examples/training/data_augmentation/README.html)
    * [Motivation](../examples/training/data_augmentation/README.html#motivation)
    * [Extend to your own datasets](../examples/training/data_augmentation/README.html#extend-to-your-own-datasets)
    * [Methodology](../examples/training/data_augmentation/README.html#methodology)
      * [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../examples/training/data_augmentation/README.html#scenario-1-limited-or-small-annotated-datasets-few-labeled-sentence-pairs)
      * [Scenario 2: No annotated datasets (Only unlabeled sentence-pairs)](../examples/training/data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)
    * [Training](../examples/training/data_augmentation/README.html#training)
    * [Citation](../examples/training/data_augmentation/README.html#citation)
  * [Training with Prompts](../examples/training/prompts/README.html)
    * [What are Prompts?](../examples/training/prompts/README.html#what-are-prompts)
    * [Why would we train with Prompts?](../examples/training/prompts/README.html#why-would-we-train-with-prompts)
    * [How do we train with

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

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

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

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

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

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

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

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

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

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

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

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

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

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

* [Performance Comparison](../examples/unsupervised_learning/README.html#performance-comparison)

* [Domain Adaptation](../examples/domain_adaptation/README.html)

* [Domain Adaptation vs. Unsupervised Learning](../examples/domain_adaptation/README.html#domain-adaptation-vs-unsupervised-learning)

* [Adaptive Pre-Training](../examples/domain_adaptation/README.html#adaptive-pre-training)

* [GPL: Generative Pseudo-Labeling](../examples/domain_adaptation/README.html#gpl-generative-pseudo-labeling)

* [Hyperparameter Optimization](../examples/training/hpo/README.html)

      * [HPO Components](../examples/training/hpo/README.html#hpo-components)
      * [Putting It All Together](../examples/training/hpo/README.html#putting-it-all-together)
      * [Example Scripts](../examples/training/hpo/README.html#example-scripts)
    * [Distributed Training](sentence_transformer/training/distributed.html)
      * [Comparison](sentence_transformer/training/distributed.html#comparison)
      * [FSDP](sentence_transformer/training/distributed.html#fsdp)

Cross Encoder

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

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

Package Reference

  * [Sentence Transformer](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_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)
* Quickstart
* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/docs/quickstart.rst)

* * *

Quickstartïƒ•

Sentence Transformerïƒ•

Characteristics of Sentence Transformer (a.k.a bi-encoder) models:

1. Calculates a **fixed-size vector representation (embedding)** given **texts or images**.

2. Embedding calculation is often **efficient** , embedding similarity calculation is **very fast**.

3. Applicable for a **wide range of tasks** , such as semantic textual similarity, semantic search, clustering, classification, paraphrase mining, and more.

4. Often used as a **first step in a two-step retrieval process** , where a Cross-Encoder (a.k.a. reranker) model is used to re-rank the top-k results from the bi-encoder.

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

Documentation

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

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

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

Other useful methods and links:

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

* [SentenceTransformer > Usage](./sentence_transformer/usage/usage.html)

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

* [SentenceTransformer > Pretrained Models](./sentence_transformer/pretrained_models.html)

* [SentenceTransformer > Training Overview](./sentence_transformer/training_overview.html)

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

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

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

```
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]])
```

With `SentenceTransformer("all-MiniLM-L6-v2")` we pick which [Sentence Transformer model](https://huggingface.co/models?library=sentence-transformers) we load. In this example, we load [all-MiniLM-L6-v2](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2), which is a MiniLM model finetuned on a large dataset of over 1 billion training pairs. Using [`SentenceTransformer.similarity()`](package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.similarity "sentence_transformers.SentenceTransformer.similarity"), we compute the similarity between all pairs of sentences. As expected, the similarity between the first two sentences (0.6660) is higher than the similarity between the first and the third sentence (0.1046) or the second and the third sentence (0.1411).

Finetuning Sentence Transformer models is easy and requires only a few lines of code. For more information, see the [Training Overview](./sentence_transformer/training_overview.html) section.

Tip

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

Cross Encoderïƒ•

Characteristics of Cross Encoder (a.k.a reranker) models:

1. Calculates a **similarity score** given **pairs of texts**.

2. Generally provides **superior performance** compared to a Sentence Transformer (a.k.a. bi-encoder) model.

3. Often **slower** than a Sentence Transformer model, as it requires computation for each pair rather than each text.

4. Due to the previous 2 characteristics, Cross Encoders are often used to **re-rank the top-k results** from a Sentence Transformer model.

The usage for Cross Encoder (a.k.a. reranker) models is similar to Sentence Transformers:

Documentation

1. `CrossEncoder`

2. `CrossEncoder.rank`

3. `CrossEncoder.predict`

Other useful methods and links:

* [CrossEncoder > Usage](./cross_encoder/usage/usage.html)

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

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

* [CrossEncoder > Dataset Overview](./cross_encoder/dataset_overview.html)

* [CrossEncoder > Loss Overview](./cross_encoder/loss_overview.html)

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

```
from sentence_transformers.cross_encoder import CrossEncoder

# 1. Load a pretrained CrossEncoder model
model = CrossEncoder("cross-encoder/stsb-distilroberta-base")
```

```
# We want to compute the similarity between the query sentence...
query = "A man is eating pasta."

# ... and all sentences in the corpus
corpus = [
    "A man is eating food.",
    "A man is eating a piece of bread.",
    "The girl is carrying a baby.",
    "A man is riding a horse.",
    "A woman is playing violin.",
    "Two men pushed carts through the woods.",
    "A man is riding a white horse on an enclosed ground.",
    "A monkey is playing drums.",
    "A cheetah is running behind its prey.",
]

# 2. We rank all sentences in the corpus for the query
ranks = model.rank(query, corpus)

# Print the scores
print("Query: ", query)
for rank in ranks:
    print(f"{rank['score']:.2f}\t{corpus[rank['corpus_id']]}")
"""
Query:  A man is eating pasta.
0.67    A man is eating food.
0.34    A man is eating a piece of bread.
```

```
0.08    A man is riding a horse.
0.07    A man is riding a white horse on an enclosed ground.
0.01    The girl is carrying a baby.
0.01    Two men pushed carts through the woods.
0.01    A monkey is playing drums.
0.01    A woman is playing violin.
0.01    A cheetah is running behind its prey.
"""

# 3. Alternatively, you can also manually compute the score between two sentences
import numpy as np

sentence_combinations = [[query, sentence] for sentence in corpus]
scores = model.predict(sentence_combinations)

# Sort the scores in decreasing order to get the corpus indices
ranked_indices = np.argsort(scores)[::-1]
print("Scores:", scores)
print("Indices:", ranked_indices)
"""
    Scores:  [0.6732372,  0.34102544,  0.00542465,  0.07569341,  0.00525378,  0.00536814,
0.06676237, 0.00534825, 0.00516717]
Indices: [0 1 3 6 2 5 7 4 8]
"""
```

With `CrossEncoder("cross-encoder/stsb-distilroberta-base")` we pick which

[CrossEncoder model](./cross_encoder/pretrained_models.html) we load. In this example, we load [cross-encoder/stsb-distilroberta-base](https://huggingface.co/cross-encoder/stsb-distilroberta-base), which is a [DistilRoBERTa](https://huggingface.co/distilbert/distilroberta-base) model finetuned on the [STS Benchmark](https://huggingface.co/datasets/sentence-transformers/stsb) dataset.

Next Stepsïƒ•

Consider reading one of the following sections next:

* [Sentence Transformers > Usage](./sentence_transformer/usage/usage.html)

* [Sentence Transformers > Pretrained Models](./sentence_transformer/pretrained_models.html)

* [Sentence Transformers > Training Overview](./sentence_transformer/training_overview.html)

* [Sentence Transformers > Training Examples > Multilingual Models](../examples/training/multilingual/README.html)

* [Cross Encoder > Usage](./cross_encoder/usage/usage.html)

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

[Previous](installation.html "Installation") [Next](sentence_transformer/usage/usage.html "Usage")

* * *

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).