[![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](../../applications/computing-embeddings/README.html)
      * [Initializing a Sentence Transformer Model](../../applications/computing-embeddings/README.html#initializing-a-sentence-transformer-model)
      * [Calculating Embeddings](../../applications/computing-embeddings/README.html#calculating-embeddings)
      * [Prompt Templates](../../applications/computing-embeddings/README.html#prompt-templates)
      * [Input Sequence Length](../../applications/computing-embeddings/README.html#id1)

* [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)

Cross Encoder

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#den

oisingautoencoderloss)

* [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#par

allelsentencesdataset)

* [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#cecorrelationevaluator)

* [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_tra

nsformers.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)

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

* MS MARCO

* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/ms_marco/README.md)

* * *

# MS MARCOïƒ•

[MS MARCO Passage Ranking](https://github.com/microsoft/MSMARCO-Passage-Ranking) is a large dataset to train models for information retrieval. It consists of about 500k real search queries from Bing search engine with the relevant text passage that answers the query.

This page shows how to **train** Sentence Transformer models on this dataset so that it can be used for searching text passages given queries (key words, phrases or questions).

If you are interested in how to use these models, see [Application - Retrieve & Re-Rank](../../applications/retrieve_rerank/README.html).

There are **pre-trained models** available, which you can directly use without the need of training your own models. For more information, see: [Pretrained Models > MSMARCO Passage Models](../../../docs/sentence_transformer/pretrained_models.html#msmarco-passage-models).

## Bi-Encoderïƒ•

For retrieval of suitable documents from a large collection, we have to use a Sentence Transformer (a.k.a. bi-encoder) model. The documents are independently encoded into fixed-sized embeddings. A query is embedded into the same vector space. Relevant documents can then be found by using cosine similarity or dot-product.

![BiEncoder](https://raw.githubusercontent.com/UKPLab/sentence-transformers/master/docs/img/BiEncoder.png)

This page describes two strategies to **train an bi-encoder** on the MS MARCO dataset:

### MultipleNegativesRankingLossïƒ•

Training code:[train_bi-encoder_mnrl.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/ms_marco/train_bi-encoder_mnrl.py)

When we use [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss"), we provide triplets: `(query, positive_passage, negative_passage)` where `positive_passage` is the relevant passage to the query and `negative_passage` is a non-relevant passage to the query. We compute the embeddings for all queries, positive passages, and negative passages in the corpus and then optimize the following objective: The `(query, positive_passage)` pair must be close in the vector space, while ``(query, negative_passage)` should be distant in vector space.

To further improve the training, we use **in-batch negatives** :

![MultipleNegativesRankingLoss](https://raw.githubusercontent.com/UKPLab/sentence-transformers/master/docs/img/MultipleNegativeRankingLoss.png)

We embed all `queries`, `positive_passages`, and `negative_passages` into the vector space. The matching `(query_i, positive_passage_i)` should be close, while there should be a large distance between a `query` and all other (positive/negative) passages from all other triplets in a batch. For a batch size of 64, we compare a query against 64+64=128 passages, from which only one passage should be close and the 127 others should be distant in vector space.

One way to **improve training** is to choose really good negatives, also know as **hard negative** : The negative should look really similar to the positive passage, but it should not be relevant to the query.

We find these hard negatives in the following way: We use existing retrieval systems (e.g. lexical search and other bi-encoder retrieval systems), and for each query we find the most relevant passages. We then use a powerful [cross-encoder/ms-marco-MiniLM-L-6-v2](https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2) [Cross-Encoder](../../applications/cross-encoder/README.html) to score the found `(query, passage)` pairs. We provide scores for 160 million such pairs in our [MS MARCO Mined Triplet dataset collection](https://huggingface.co/collections/sentence-transformers/ms-marco-mined-triplets-6644d6f1ff58c5103fe65f23).

For [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss"), we must ensure that in the triplet `(query, positive_passage, negative_passage)` that the `negative_passage` is indeed not relevant for the query. The MS MARCO dataset is sadly **highly redundant** , and even though that there is on average only one passage marked as relevant for a query, it actually contains many passages that humans would consider as relevant. We must ensure that these passages are **not passed as negatives** : We do this by ensuring a certain threshold in the CrossEncoder scores between the relevant passages and the mined hard negative. By default, we set a threshold of 3: If the `(query, positive_passage)` gets a score of 9 from the CrossEncoder, than we will only consider negatives with a score below 6 from the CrossEncoder. This threshold ensures that we actually use negatives in our triplets.

You can find this data by traversing to any of the datasets in the [MS MARCO Mined Triplet dataset collection](https://huggingface.co/collections/sentence-transformers/ms-marco-mined-triplets-6644d6f1ff58c5103fe65f23) and using the `triplet-hard` subset. Across all datasets, this refers to 175.7 million triplets. The original data can be found [here](https://huggingface.co/datasets/sentence-transformers/msmarco-hard-negatives). Load some of it using:

```
from datasets import load_dataset

train_dataset = load_dataset("sentence-transformers/msmarco-co-condenser-margin-mse-sym-mnrl-mean-v1", "triplet-hard", split="train")
# Dataset({
#     features: ['query', 'positive', 'negative'],
#     num_rows: 11662655
# })
print(train_dataset[0])
# {'query': 'what are the liberal arts?', 'positive': 'liberal arts. 1. the academic course of instruction at a college intended to provide general knowledge and comprising the arts, humanities, natural sciences, and social sciences, as opposed to professional or technical subjects.', 'negative': "Rather than preparing students for a specific career, liberal arts programs focus on cultural literacy and hone communication and analytical skills. They often cover various disciplines, ranging from the humanities to social sciences. 1  Program Levels in Liberal Arts: Associate degree, Bachelor's degree, Master's degree."}
```

### MarginMSEïƒ•

Training code:[train_bi-encoder_margin-mse.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/ms_marco/train_bi-encoder_margin-mse.py)

[`MarginMSELoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MarginMSELoss "sentence_transformers.losses.MarginMSELoss") is based on the paper of [HofstÃ¤tter et al](https://arxiv.org/abs/2010.02666). Like when training with [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss"), we can use triplets: `(query, passage1, passage2)`. However, in contrast to [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss"), passage1 and passage2 do not have to be strictly positive/negative, both can be relevant or not relevant for a given query.

We then compute the [Cross-Encoder](../../applications/cross-encoder/README.html) score for `(query, passage1)` and `(query, passage2)`. We provide scores for 160 million such pairs in our [msmarco-hard-negatives dataset](https://huggingface.co/datasets/sentence-transformers/msmarco-hard-negatives). We then compute the distance: `CE_distance = CEScore(query, passage1) - CEScore(query, passage2)`.

For our Sentence Transformer (e.g. bi-encoder) training, we encode `query`, `passage1`, and `passage2` into embeddings and then measure the dot-product between `(query, passage1)` and `(query, passage2)`. Again, we measure the distance: `BE_distance = DotScore(query, passage1) - DotScore(query, passage2)`

We then want to ensure that the distance predicted by the bi-encoder is close to the distance predicted by the cross-encoder, i.e., we optimize the mean-squared error (MSE) between `CE_distance` and `BE_distance`.

An **advantage** of [`MarginMSELoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MarginMSELoss "sentence_transformers.losses.MarginMSELoss") compared to [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html #sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss") is that we **donâ€™t require** a `positive` and `negative` passage. As mentioned before, MS MARCO is redundant and many passages contain the same or similar content. With [`MarginMSELoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MarginMSELoss "sentence_transformers.losses.MarginMSELoss"), we can train on two relevant passages without issues: In that case, the `CE_distance` will be smaller and we expect that our bi-encoder also puts both passages closer in the vector space.

And **disadvantage** of [`MarginMSELoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MarginMSELoss "sentence_transformers.losses.MarginMSELoss") is the slower training time: We need way more epochs to get good results. In [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss"), with a batch size of 64, we compare one query against 128 passages. With [`MarginMSELoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MarginMSELoss "sentence_transformers.losses.MarginMSELoss"), we compare a query only against two passages.

[Previous](../quora_duplicate_questions/README.html "Quora Duplicate Questions") [Next](../matryoshka/README.html "Matryoshka Embeddings")

* * *

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).