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

Getting Started

Sentence Transformer

* [Multi-Process / Multi-GPU Encoding](../../applications/computing-embeddings/README.html#multi-process-multi-gpu-encoding)
* [Semantic Textual Similarity](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html)
  * [Similarity Calculation](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../../applications/semantic-search/README.html)
  * [Background](../../applications/semantic-search/README.html#background)
  * [Symmetric vs. Asymmetric Semantic Search](../../applications/semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
  * [Manual Implementation](../../applications/semantic-search/README.html#manual-implementation)
  * [Optimized Implementation](../../applications/semantic-search/README.html#optimized-implementation)
  * [Speed Optimization](../../applications/semantic-search/README.html#speed-optimization)
  * [Elasticsearch](../../applications/semantic-search/README.html#elasticsearch)
  * [Approximate Nearest Neighbor](../../applications/semantic-search/README.html#approximate-nearest-neighbor)
  * [Retrieve & Re-Rank](../../applications/semantic-search/README.html#retrieve-re-rank)
  * [Examples](../../applications/semantic-search/README.html#examples)
* [Retrieve & Re-Rank](../../applications/retrieve_rerank/README.html)
  * [Retrieve & Re-Rank Pipeline](../../applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)
  * [Retrieval: Bi-Encoder](../../applications/retrieve_rerank/README.html#retrieval-bi-encoder)

      * [Scalar (int8) Quantization](../../applications/embedding-quantization/README.html#scalar-int8-quantization)
      * [Additional extensions](../../applications/embedding-quantization/README.html#additional-extensions)
      * [Demo](../../applications/embedding-quantization/README.html#demo)
      * [Try it yourself](../../applications/embedding-quantization/README.html#try-it-yourself)
    * [Speeding up Inference](../../../docs/sentence_transformer/usage/efficiency.html)
      * [PyTorch](../../../docs/sentence_transformer/usage/efficiency.html#pytorch)
      * [ONNX](../../../docs/sentence_transformer/usage/efficiency.html#onnx)
      * [OpenVINO](../../../docs/sentence_transformer/usage/efficiency.html#openvino)
      * [Benchmarks](../../../docs/sentence_transformer/usage/efficiency.html#benchmarks)
    * [Creating Custom Models](../../../docs/sentence_transformer/usage/custom_models.html)
      * [Structure of Sentence Transformer Models](../../../docs/sentence_transformer/usage/custom_models.html#structure-of-sentence-transformer-models)
      * [Sentence Transformer Model from a Transformers Model](../../../docs/sentence_transformer/usage/custom_models.html#sentence-transformer-model-from-a-transformers-model)
  * [Pretrained Models](../../../docs/sentence_transformer/pretrained_models.html)
    * [Original Models](../../../docs/sentence_transformer/pretrained_models.html#original-models)
      * [Semantic Search Models](../../../docs/sentence_transformer/pretrained_models.html#semantic-search-models)
      * [Multi-QA Models](../../../docs/sentence_transformer/pretrained_models.html#multi-qa-models)
      * [MSMARCO Passage Models](../../../docs/sentence_transformer/pretrained_models.html#msmarco-passage-models)
      * [Multilingual Models](../../../docs/sentence_transformer/pretrained_models.html#multilingual-models)

      * [Semantic Similarity Models](../../../docs/sentence_transformer/pretrained_models.html#semantic-similarity-models)

      * [Bitext Mining](../../../docs/sentence_transformer/pretrained_models.html#bitext-mining)

      * [Image & Text-Models](../../../docs/sentence_transformer/pretrained_models.html#image-text-models)

      * [INSTRUCTOR models](../../../docs/sentence_transformer/pretrained_models.html#instructor-models)

      * [Scientific Similarity Models](../../../docs/sentence_transformer/pretrained_models.html#scientific-similarity-models)

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

    * [Why Finetune?](../../../docs/sentence_transformer/training_overview.html#why-finetune)

    * [Training Components](../../../docs/sentence_transformer/training_overview.html#training-components)

    * [Dataset](../../../docs/sentence_transformer/training_overview.html#dataset)

      * [Dataset Format](../../../docs/sentence_transformer/training_overview.html#dataset-format)

    * [Loss Function](../../../docs/sentence_transformer/training_overview.html#loss-function)

    * [Training Arguments](../../../docs/sentence_transformer/training_overview.html#training-arguments)

    * [Evaluator](../../../docs/sentence_transformer/training_overview.html#evaluator)

    * [Trainer](../../../docs/sentence_transformer/training_overview.html#trainer)

      * [Callbacks](../../../docs/sentence_transformer/training_overview.html#callbacks)

    * [Multi-Dataset Training](../../../docs/sentence_transformer/training_overview.html#multi-dataset-training)

    * [Deprecated Training](../../../docs/sentence_transformer/training_overview.html#deprecated-training)

    * [Best Base Embedding Models](../../../docs/sentence_transformer/training_overview.html#best-base-embedding-models)

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#denoisingautoencoderloss)

*

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

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

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

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

* [MatryoshkaLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshkaloss)

* [Matryoshka2dLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshka2dloss)

* [AdaptiveLayerLoss](../../../docs/package_reference/sentence_transformer/losses.html#adaptivelayerloss)

* [MegaBatchMarginLoss](../../../docs/package_reference/sentence_transformer/losses.html#megabatchmarginloss)

* [MultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#multiplenegativesrankingloss)

* [CachedMultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativesrankingloss)

* [MultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#multiplenegativessymmetricrankingloss)

* [CachedMultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativessymmetricrankingloss)

    * [SoftmaxLoss](../../../docs/package_reference/sentence_transformer/losses.html#softmaxloss)

    * [TripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#tripletloss)

  * [Samplers](../../../docs/package_reference/sentence_transformer/sampler.html)

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

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

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

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

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

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

* [NanoBEIREvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#nanobe

irevaluator)

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

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

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

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

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

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

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

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

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

*

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

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

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

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

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

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

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

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

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

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

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

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

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

* [Training

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

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

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

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

    * [CECorrelationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#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_transformers.backend.export_static_quantized_openvino_model)

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

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

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

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

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

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

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

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

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

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

* [](../../../index.html)
* [Training Examples](../../../docs/sentence_transformer/training/examples.html)
* Quora Duplicate Questions
* [ Edit on

GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/quora_duplicate_questions/README.md)

* * *

# Quora Duplicate Questionsïƒ•

This folder contains scripts that demonstrate how to train SentenceTransformers for **Information Retrieval**. As a simple example, we will use the [Quora Duplicate Questions dataset](https://huggingface.co/datasets/sentence-transformers/quora-duplicates). It contains over 500,000 sentences with over 400,000 pairwise annotations whether two questions are a duplicate or not.

Models trained on this dataset can be used for mining duplicate questions, i.e., given a large set of sentences (in this case questions), identify all pairs that are duplicates. See [Paraphrase Mining](../../applications/paraphrase-mining/README.html) for an example how to use sentence transformers to mine for duplicate questions / paraphrases. This approach can be scaled to hundred thousands of sentences.

## Trainingïƒ•

Choosing the right loss function is crucial for finetuning useful models. For the given task, two loss functions are especially suitable: [`OnlineContrastiveLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.OnlineContrastiveLoss "sentence_transformers.losses.OnlineContrastiveLoss") and

[`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html

#sentence_transformers.losses.MultipleNegativesRankingLoss

"sentence_transformers.losses.MultipleNegativesRankingLoss").

### Contrastive Lossïƒ•

For the complete training example, see

[training_OnlineContrastiveLoss.py](https://github.com/UKPLab/sentence-

transformers/tree/master/examples/training/quora_duplicate_questions/training_OnlineContrastiveL

oss.py).

The Quora Duplicates dataset has a [pair-class

subset](https://huggingface.co/datasets/sentence-transformers/quora-

duplicates/viewer/pair-class) which consists of question pairs and labels: 1

for duplicate and 0 for different.

As shown by our [Loss

Overview](../../../docs/sentence_transformer/loss_overview.md), this allows us

to use

[`ContrastiveLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_tran

sformers.losses.ContrastiveLoss

"sentence_transformers.losses.ContrastiveLoss"). Similar pairs with label 1

are pulled together, so that they are close in vector space, while dissimilar

pairs that are closer than a defined margin are pushed away in vector space.

An improved version is

[`OnlineContrastiveLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.OnlineContrastiveLoss "sentence_transformers.losses.OnlineContrastiveLoss"). This loss looks which negative pairs have a lower distance than the largest positive pair and which positive pairs have a higher distance than the lowest distance of negative pairs. I.e., this loss automatically detects the hard cases in a batch and computes the loss only for these cases.

The loss can be used like this:

```
from datasets import load_dataset

train_dataset = load_dataset("sentence-transformers/quora-duplicates", "pair-class", split="train")
# => Dataset({
#     features: ['sentence1', 'sentence2', 'label'],
#     num_rows: 404290
# })
print(train_dataset[0])
# => {'sentence1': 'What is the step by step guide to invest in share market in india?', 'sentence2': 'What is the step by step guide to invest in share market?', 'label': 0}
train_loss = losses.OnlineContrastiveLoss(model=model, margin=0.5)
```

## MultipleNegativesRankingLossïƒ•

For the complete example, see [training_MultipleNegativesRankingLoss.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/quora_duplicate_questions/training_MultipleNegativesRankingLoss.py).

[`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss") is especially suitable for Information Retrieval / Semantic Search. A nice advantage is that it only requires positive pairs, i.e., we only need examples of duplicate questions. See [NLI > MultipleNegativesRankingLoss](../nli/README.html#multiplenegativesrankingloss) for more information on how the loss works.

Using the loss is easy and does not require tuning of any hyperparameters:

```
from datasets import load_dataset


train_dataset = load_dataset("sentence-transformers/quora-duplicates", "pair", split="train")
# => Dataset({
#     features: ['anchor', 'positive'],
#     num_rows: 149263
# })
print(train_dataset[0])
 # => {'anchor': 'Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say
about me?', 'positive': "I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?"}
    train_loss = losses.MultipleNegativesRankingLoss(model)
```

As â€˜is_duplicateâ€™ is a symmetric relation, we can use not just (anchor, positive) but also (positive, anchor) to our training sample set:

```
    from datasets import concatenate_datasets

    train_dataset = concatenate_datasets([
        train_dataset,
        train_dataset.rename_columns({"anchor": "positive", "positive": "anchor"})
    ])
    # Dataset({
    #     features: ['anchor', 'positive'],
    #     num_rows: 298526
    # })
```

Note

Increasing the batch sizes usually yields better results, as the task gets harder. It is more difficult to identify the correct duplicate question out of a set of 100 questions than out of a set of only 10 questions. So it is advisable to set the training batch size as large as possible. I trained it with a batch size of 350 on 32 GB GPU memory.

Note

[`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss") only works if _(a_i, b_j)_ with j != i is actually a negative, non-duplicate question pair.
In few instances, this assumption is wrong. But in the majority of cases, if we sample two random questions, they are not duplicates. If your dataset cannot fulfil this property, [`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss") might not work well.

### Multi-Task-Learningïƒ•

[`ContrastiveLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.ContrastiveLoss "sentence_transformers.losses.ContrastiveLoss") works well for pair classification, i.e., given two pairs, are these duplicates or not. It pushes negative pairs far away in vector space, so that the distinguishing between duplicate and non-duplicate pairs works good.

[`MultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html

#sentence_transformers.losses.MultipleNegativesRankingLoss "sentence_transformers.losses.MultipleNegativesRankingLoss") on the other sides mainly reduces the distance between positive pairs out of large set of possible candidates. However, the distance between non-duplicate questions is not so large, so that this loss does not work that well for pair classification.

In [training_multi-task-learning.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/quora_duplicate_questions/training_multi-task-learning.py) I demonstrate how we can train the network with both losses. The essential code is to define both losses and to pass it to the fit method.

```
from datasets import load_dataset
from sentence_transformers.losses import ContrastiveLoss, MultipleNegativesRankingLoss
from sentence_transformers import SentenceTransformerTrainer, SentenceTransformer

model_name = "stsb-distilbert-base"
model = SentenceTransformer(model_name)

# https://huggingface.co/datasets/sentence-transformers/quora-duplicates
mnrl_dataset = load_dataset(
    "sentence-transformers/quora-duplicates", "triplet", split="train"
)  # The "pair" subset also works
mnrl_train_dataset = mnrl_dataset.select(range(100000))
mnrl_eval_dataset = mnrl_dataset.select(range(100000, 101000))
```

```
mnrl_train_loss = MultipleNegativesRankingLoss(model=model)


# https://huggingface.co/datasets/sentence-transformers/quora-duplicates
cl_dataset = load_dataset("sentence-transformers/quora-duplicates", "pair-class", split="train")
cl_train_dataset = cl_dataset.select(range(100000))
cl_eval_dataset = cl_dataset.select(range(100000, 101000))


cl_train_loss = ContrastiveLoss(model=model, margin=0.5)


# Create the trainer & start training
trainer = SentenceTransformerTrainer(
    model=model,
    train_dataset={
        "mnrl": mnrl_train_dataset,
        "cl": cl_train_dataset,
    },
    eval_dataset={
        "mnrl": mnrl_eval_dataset,
        "cl": cl_eval_dataset,
    },
    loss={
        "mnrl": mnrl_train_loss,
        "cl": cl_train_loss,
    },
)
trainer.train()
```

## Pretrained Modelsïƒ•

Currently the following models trained on Quora Duplicate Questions are available:

* [distilbert-base-nli-stsb-quora-ranking](https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-quora-ranking): We extended the [distilbert-base-nli-stsb-mean-tokens](https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-mean-tokens) model and trained it with _OnlineContrastiveLoss_ and with _MultipleNegativesRankingLoss_ on the Quora Duplicate questions dataset. For the code, see [training_multi-task-learning.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/quora_duplicate_questions/training_multi-task-learning.py)

* [distilbert-multilingual-nli-stsb-quora-ranking](https://huggingface.co/sentence-transformers/distilbert-multilingual-nli-stsb-quora-ranking): Extension of _distilbert-base-nli-stsb-quora-ranking_ to be multi-lingual. Trained on parallel data for 50 languages.

You can load & use pre-trained models like this:

```
    from sentence_transformers import SentenceTransformer
    model = SentenceTransformer("distilbert-base-nli-stsb-quora-ranking")
```

[ Previous](../paraphrases/README.html "Paraphrase Data") [Next ](../ms_marco/README.html "MS MARCO")

* * *

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).