[![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)

* [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)

* [Re-Ranker: Cross-Encoder](../../applications/retrieve_rerank/README.html#re-ranker-cross-encoder)
  * [Example Scripts](../../applications/retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../../applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../../applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
* [Clustering](../../applications/clustering/README.html)
  * [k-Means](../../applications/clustering/README.html#k-means)
  * [Agglomerative Clustering](../../applications/clustering/README.html#agglomerative-clustering)
  * [Fast Clustering](../../applications/clustering/README.html#fast-clustering)
  * [Topic Modeling](../../applications/clustering/README.html#topic-modeling)
* [Paraphrase Mining](../../applications/paraphrase-mining/README.html)
* [`paraphrase_mining()`](../../applications/paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
* [Translated Sentence Mining](../../applications/parallel-sentence-mining/README.html)
* [Margin Based Mining](../../applications/parallel-sentence-mining/README.html#margin-based-mining)
  * [Examples](../../applications/parallel-sentence-mining/README.html#examples)
* [Image Search](../../applications/image-search/README.html)
  * [Installation](../../applications/image-search/README.html#installation)
  * [Usage](../../applications/image-search/README.html#usage)
  * [Examples](../../applications/image-search/README.html#examples)
* [Embedding Quantization](../../applications/embedding-quantization/README.html)
* [Binary Quantization](../../applications/embedding-quantization/README.html#binary-quantization)

* [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)

* [Dataset Overview](../../../docs/sentence_transformer/dataset_overview.html)
* [Datasets on the Hugging Face Hub](../../../docs/sentence_transformer/dataset_overview.html#datasets-on-the-hugging-face-hub)
* [Pre-existing Datasets](../../../docs/sentence_transformer/dataset_overview.html#pre-existing-datasets)
* [Loss Overview](../../../docs/sentence_transformer/loss_overview.html)
* [Loss modifiers](../../../docs/sentence_transformer/loss_overview.html#loss-modifiers)
* [Distillation](../../../docs/sentence_transformer/loss_overview.html#distillation)
* [Commonly used Loss Functions](../../../docs/sentence_transformer/loss_overview.html#commonly-used-loss-functions)
* [Custom Loss Functions](../../../docs/sentence_transformer/loss_overview.html#custom-loss-functions)
* [Training Examples](../../../docs/sentence_transformer/training/examples.html)
* [Semantic Textual Similarity](../sts/README.html)
* [Training data](../sts/README.html#training-data)
* [Loss Function](../sts/README.html#loss-function)
* [Natural Language Inference](../nli/README.html)
* [Data](../nli/README.html#data)
* [SoftmaxLoss](../nli/README.html#softmaxloss)
* [MultipleNegativesRankingLoss](../nli/README.html#multiplenegativesrankingloss)
* [Paraphrase Data](../paraphrases/README.html)
* [Pre-Trained Models](../paraphrases/README.html#pre-trained-models)
* [Quora Duplicate Questions](../quora_duplicate_questions/README.html)
* [Training](../quora_duplicate_questions/README.html#training)
* [MultipleNegativesRankingLoss](../quora_duplicate_questions/README.html#multiplenegativesrankingloss)

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

    * [Speed - Performance Trade-Off](../distillation/README.html#speed-performance-trade-off)
    * [Dimensionality Reduction](../distillation/README.html#dimensionality-reduction)
    * [Quantization](../distillation/README.html#quantization)
  * [Augmented SBERT](../data_augmentation/README.html)
    * [Motivation](../data_augmentation/README.html#motivation)
    * [Extend to your own datasets](../data_augmentation/README.html#extend-to-your-own-datasets)
    * [Methodology](../data_augmentation/README.html#methodology)
      * [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../data_augmentation/README.html#scenario-1-limited-or-small-annotated-datasets-few-labeled-sentence-pairs)
      * [Scenario 2: No annotated datasets (Only unlabeled sentence-pairs)](../data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)
    * [Training](../data_augmentation/README.html#training)
    * [Citation](../data_augmentation/README.html#citation)
  * [Training with Prompts](../prompts/README.html)
    * [What are Prompts?](../prompts/README.html#what-are-prompts)
    * [Why would we train with Prompts?](../prompts/README.html#why-would-we-train-with-prompts)
    * [How do we train with Prompts?](../prompts/README.html#how-do-we-train-with-prompts)
  * [Training with PEFT Adapters](../peft/README.html)
    * [Compatibility Methods](../peft/README.html#compatibility-methods)
    * [Adding a New Adapter](../peft/README.html#adding-a-new-adapter)
    * [Loading a Pretrained Adapter](../peft/README.html#loading-a-pretrained-adapter)
    * [Training Script](../peft/README.html#training-script)
  * [Unsupervised Learning](../../unsupervised_learning/README.html)

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#cache

dgistembedloss)

* [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#matryoshka2

dloss)

*

[AdaptiveLayerLoss](../../../docs/package_reference/sentence_transformer/losses.html#adaptivelaye

rloss)

*

[MegaBatchMarginLoss](../../../docs/package_reference/sentence_transformer/losses.html#megabat

chmarginloss)

*

[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)

* Hyperparameter Optimization

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

* * *

Hyperparameter Optimizationïƒ•

The [`SentenceTransformerTrainer`](../../../docs/package_reference/sentence_transformer/trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer "sentence_transformers.trainer.SentenceTransformerTrainer") supports hyperparameter optimization using `transformers`, which in turn supports four hyperparameter search backends: [optuna](https://optuna.org/), [sigopt](https://sigopt.org/), [raytune](https://docs.ray.io/en/latest/tune/index.html), and [wandb](https://wandb.ai/site/sweeps). You should install your backend of choice before using it:

```
pip install optuna/sigopt/wandb/ray[tune]
```

On this page, weâ€™ll show you how to use the hyperparameter optimization feature with the optuna backend. The other backends are similar to use, but

you should refer to their respective documentation or the [transformers HPO documentation](https://huggingface.co/docs/transformers/en/hpo_train) for more information.

HPO Componentsïƒ•

The hyperparameter optimization process consists of the following components:

Hyperparameter Search Space Specify ranges for hyperparameter values. Model Initialization Initialize a SentenceTransformer model for a trial. Loss Initialization Initialize a loss function given a model. Compute Objective Determines the value to be minimized or maximized.

Hyperparameter Search Spaceïƒ•

The hyperparameter search space is defined by a function that returns a dictionary of hyperparameters and their respective search spaces. Hereâ€™s an example using `optuna` of a search space function that defines the hyperparameters for a SentenceTransformer model:

```
    def hpo_search_space(trial):
        return {
            "num_train_epochs": trial.suggest_int("num_train_epochs", 1, 2),
```

```
        "per_device_train_batch_size": trial.suggest_int("per_device_train_batch_size", 32, 128),
        "warmup_ratio": trial.suggest_float("warmup_ratio", 0, 0.3),
        "learning_rate": trial.suggest_float("learning_rate", 1e-6, 1e-4, log=True),
    }
```

Model Initializationïƒ•

The model initialization function is a function that takes the hyperparameters of the current â€œtrialâ€• as input and returns a SentenceTransformer model. Generally, this function is quite simple. Hereâ€™s an example of a model initialization function:

```
def hpo_model_init(trial):
    return SentenceTransformer("distilbert-base-uncased")
```

Loss Initializationïƒ•

The loss initialization function is a function that takes the model initialized for the current trial and returns a loss function. Hereâ€™s an example of a loss initialization function:

```
    def hpo_loss_init(model):
        return losses.CosineSimilarityLoss(model)
```

Compute Objectiveïƒ•

The compute objective function is a function that takes the evaluation
`metrics` and returns the float value to be minimized or maximized. Hereâ€™s
an example of a compute objective function:

```
    def hpo_compute_objective(metrics):
        return metrics["eval_sts-dev_spearman_cosine"]
```

Putting It All Togetherïƒ•

You can perform HPO on any regular training loop, the only difference being
that you donâ€™t call
[`SentenceTransformerTrainer.train`](../../../docs/package_reference/sentence_transformer/trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer.train
"sentence_transformers.trainer.SentenceTransformerTrainer.train"), but
[`SentenceTransformerTrainer.hyperparameter_search`](../../../docs/package_reference/sentence_transformer/trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer.hyperparameter_search
"sentence_transformers.trainer.SentenceTransformerTrainer.hyperparameter_search")

instead. Hereâ€™s an example of how to put it all together:

Documentation

1. [sentence-transformers/all-nli](https://huggingface.co/datasets/sentence-transformers/all-nli)

2. [`EmbeddingSimilarityEvaluator`](../../../docs/package_reference/sentence_transformer/evaluation.html#sentence_transformers.evaluation.EmbeddingSimilarityEvaluator "sentence_transformers.evaluation.EmbeddingSimilarityEvaluator")

3. Hyperparameter Search Space

4. Model Initialization

5. Loss Initialization

6. Compute Objective

7. [`SentenceTransformerTrainingArguments`](../../../docs/package_reference/sentence_transformer/training_args.html#sentence_transformers.training_args.SentenceTransformerTrainingArguments "sentence_transformers.training_args.SentenceTransformerTrainingArguments")

8. [`SentenceTransformerTrainer`](../../../docs/package_reference/sentence_transformer/trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer

"sentence_transformers.trainer.SentenceTransformerTrainer")

9.

[`hyperparameter_search()`](../../../docs/package_reference/sentence_transformer/trainer.html#sentence_transformers.trainer.SentenceTransformerTrainer.hyperparameter_search "sentence_transformers.trainer.SentenceTransformerTrainer.hyperparameter_search")

```
from sentence_transformers import losses
from sentence_transformers import SentenceTransformer, SentenceTransformerTrainer, SentenceTransformerTrainingArguments
from sentence_transformers.evaluation import EmbeddingSimilarityEvaluator, SimilarityFunction
from sentence_transformers.training_args import BatchSamplers
from datasets import load_dataset

# 1. Load the AllNLI dataset: https://huggingface.co/datasets/sentence-transformers/all-nli, only 10k train and 1k dev
train_dataset = load_dataset("sentence-transformers/all-nli", "triplet", split="train[:10000]")
eval_dataset = load_dataset("sentence-transformers/all-nli", "triplet", split="dev[:1000]")

# 2. Create an evaluator to perform useful HPO
stsb_eval_dataset = load_dataset("sentence-transformers/stsb", split="validation")
dev_evaluator = EmbeddingSimilarityEvaluator(
    sentences1=stsb_eval_dataset["sentence1"],
    sentences2=stsb_eval_dataset["sentence2"],
    scores=stsb_eval_dataset["score"],
```

```
    main_similarity=SimilarityFunction.COSINE,
    name="sts-dev",
)

# 3. Define the Hyperparameter Search Space
def hpo_search_space(trial):
    return {
        "num_train_epochs": trial.suggest_int("num_train_epochs", 1, 2),
        "per_device_train_batch_size": trial.suggest_int("per_device_train_batch_size", 32, 128),
        "warmup_ratio": trial.suggest_float("warmup_ratio", 0, 0.3),
        "learning_rate": trial.suggest_float("learning_rate", 1e-6, 1e-4, log=True),
    }

# 4. Define the Model Initialization
def hpo_model_init(trial):
    return SentenceTransformer("distilbert-base-uncased")

# 5. Define the Loss Initialization
def hpo_loss_init(model):
    return losses.MultipleNegativesRankingLoss(model)

# 6. Define the Objective Function
def hpo_compute_objective(metrics):
    """
    Valid keys are: 'eval_loss', 'eval_sts-dev_pearson_cosine', 'eval_sts-dev_spearman_cosine',
                    'eval_sts-dev_pearson_manhattan',     'eval_sts-dev_spearman_manhattan',
'eval_sts-dev_pearson_euclidean',
```

```
                                'eval_sts-dev_spearman_euclidean',      'eval_sts-dev_pearson_dot',
'eval_sts-dev_spearman_dot',
                            'eval_sts-dev_pearson_max',    'eval_sts-dev_spearman_max',    'eval_runtime',
'eval_samples_per_second',
        'eval_steps_per_second', 'epoch'


        due to the evaluator that we're using.
        """
        return metrics["eval_sts-dev_spearman_cosine"]


    # 7. Define the training arguments
    args = SentenceTransformerTrainingArguments(
        # Required parameter:
        output_dir="checkpoints",
        # Optional training parameters:
        # max_steps=10000, # We might want to limit the number of steps for HPO
        fp16=True,  # Set to False if you get an error that your GPU can't run on FP16
        bf16=False,  # Set to True if you have a GPU that supports BF16
         batch_sampler=BatchSamplers.NO_DUPLICATES,  # MultipleNegativesRankingLoss benefits
from no duplicate samples in a batch
        # Optional tracking/debugging parameters:
        eval_strategy="no", # We don't need to evaluate/save during HPO
        save_strategy="no",
        logging_steps=10,
        run_name="hpo",  # Will be used in W&B if `wandb` is installed
    )
```

```
# 8. Create the trainer with model_init rather than model
trainer = SentenceTransformerTrainer(
    model=None,
    args=args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    evaluator=dev_evaluator,
    model_init=hpo_model_init,
    loss=hpo_loss_init,
)

# 9. Perform the HPO
best_trial = trainer.hyperparameter_search(
    hp_space=hpo_search_space,
    compute_objective=hpo_compute_objective,
    n_trials=20,
    direction="maximize",
    backend="optuna",
)
print(best_trial)
```

[I 2024-05-17 15:10:47,844] Trial 0 finished with value: 0.7889856589698055 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 123, 'warmup_ratio': 0.07380948785410107, 'learning_rate': 2.686331417509812e-06}. Best is trial 0 with value: 0.7889856589698055.

[I 2024-05-17 15:12:13,283] Trial 1 finished with value: 0.7927780672090986 and parameters:

{'num_train_epochs': 2, 'per_device_train_batch_size': 69, 'warmup_ratio': 0.2927897848007451, 'learning_rate': 5.885372118095137e-06}. Best is trial 1 with value: 0.7927780672090986.

[I 2024-05-17 15:12:43,896] Trial 2 finished with value: 0.7684829743509601 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 114, 'warmup_ratio': 0.0739429232666916, 'learning_rate': 7.344415188959276e-05}. Best is trial 1 with value: 0.7927780672090986.

[I 2024-05-17 15:14:49,730] Trial 3 finished with value: 0.7873032743147989 and parameters: {'num_train_epochs': 2, 'per_device_train_batch_size': 43, 'warmup_ratio': 0.15184370143796674, 'learning_rate': 9.703232080395476e-06}. Best is trial 1 with value: 0.7927780672090986.

[I 2024-05-17 15:15:39,597] Trial 4 finished with value: 0.7759251781929949 and parameters: {'num_train_epochs': 2, 'per_device_train_batch_size': 127, 'warmup_ratio': 0.263946220093495, 'learning_rate': 1.231454337152625e-06}. Best is trial 1 with value: 0.7927780672090986.

[I 2024-05-17 15:17:02,191] Trial 5 finished with value: 0.7964580509886684 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 34, 'warmup_ratio': 0.2276865359631089, 'learning_rate': 7.889007438884571e-06}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:18:55,559] Trial 6 finished with value: 0.7901878917859169 and parameters: {'num_train_epochs': 2, 'per_device_train_batch_size': 48, 'warmup_ratio': 0.23228838664572948, 'learning_rate': 2.883013292682523e-06}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:20:27,027] Trial 7 finished with value: 0.7935671067660925 and parameters: {'num_train_epochs': 2, 'per_device_train_batch_size': 62, 'warmup_ratio': 0.22061123927198237, 'learning_rate': 2.95413457610349e-06}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:22:23,147] Trial 8 finished with value: 0.7848123114933252 and parameters: {'num_train_epochs': 2, 'per_device_train_batch_size': 45, 'warmup_ratio': 0.23071701022961139, 'learning_rate': 9.793681667449783e-06}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:22:52,826] Trial 9 finished with value: 0.7909708416168918 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 121, 'warmup_ratio': 0.22440506724181647, 'learning_rate': 4.0744671365843346e-05}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:23:30,395] Trial 10 finished with value: 0.7928991732385567 and parameters:

{'num_train_epochs': 1, 'per_device_train_batch_size': 89, 'warmup_ratio': 0.14607293301068847, 'learning_rate': 2.5557492055039498e-05}. Best is trial 5 with value: 0.7964580509886684.

[I 2024-05-17 15:24:18,024] Trial 11 finished with value: 0.7991870087507459 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 66, 'warmup_ratio': 0.16886154348739527, 'learning_rate': 3.705926066938032e-06}. Best is trial 11 with value: 0.7991870087507459.

[I 2024-05-17 15:25:44,198] Trial 12 finished with value: 0.7923304174306207 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 33, 'warmup_ratio': 0.15953772535423974, 'learning_rate': 1.8076298025704224e-05}. Best is trial 11 with value: 0.7991870087507459.

[I 2024-05-17 15:26:20,739] Trial 13 finished with value: 0.8020260244040395 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 90, 'warmup_ratio': 0.18105202625281253, 'learning_rate': 5.513908793512551e-06}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:26:57,783] Trial 14 finished with value: 0.7571110256860063 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 95, 'warmup_ratio': 0.00122391151793258, 'learning_rate': 1.0432486633629492e-06}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:27:32,581] Trial 15 finished with value: 0.8009013936824717 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 101, 'warmup_ratio': 0.1761274711346081, 'learning_rate': 4.5918293464430035e-06}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:28:05,850] Trial 16 finished with value: 0.8017668050806169 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 103, 'warmup_ratio': 0.10766501647726355, 'learning_rate': 5.0309795522333e-06}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:28:37,393] Trial 17 finished with value: 0.7769412380909586 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 108, 'warmup_ratio': 0.1036610178950246, 'learning_rate': 1.7747598626081271e-06}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:29:19,340] Trial 18 finished with value: 0.8011921300048339 and parameters: {'num_train_epochs': 1, 'per_device_train_batch_size': 80, 'warmup_ratio': 0.117014165550441, 'learning_rate': 1.238558867958792e-05}. Best is trial 13 with value: 0.8020260244040395.

[I 2024-05-17 15:29:59,508] Trial 19 finished with value: 0.8027501854704168 and parameters:

{'num_train_epochs': 1, 'per_device_train_batch_size': 84, 'warmup_ratio': 0.014601112207929548, 'learning_rate': 5.627813947769514e-06}. Best is trial 19 with value: 0.8027501854704168.

BestRun(run_id='19', objective=0.8027501854704168, hyperparameters={'num_train_epochs': 1, 'per_device_train_batch_size': 84, 'warmup_ratio': 0.014601112207929548, 'learning_rate': 5.627813947769514e-06}, run_summary=None)

As you can see, the strongest hyperparameters reached **0.802** Spearman correlation on the STS (dev) benchmark. For context, training with the default training arguments (`per_device_train_batch_size=8`, `learning_rate=5e-5`) results in **0.736** , and hyperparameters chosen based on experience (`per_device_train_batch_size=64`, `learning_rate=2e-5`) results in **0.783** Spearman correlation. Consequently, HPO proved quite effective here in improving the model performance.

Example Scriptsïƒ•

* [hpo_nli.py](hpo_nli.py) \- An example script that performs hyperparameter optimization on the AllNLI dataset.

[Previous](../../domain_adaptation/README.html "Domain Adaptation") [Next](../../../docs/sentence_transformer/training/distributed.html "Distributed Training")

* * *

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).