[![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](../computing-embeddings/README.html)
    * [Initializing a Sentence Transformer Model](../computing-embeddings/README.html#initializing-a-sentence-transformer-model)
    * [Calculating Embeddings](../computing-embeddings/README.html#calculating-embeddings)
    * [Prompt Templates](../computing-embeddings/README.html#prompt-templates)
    * [Input Sequence Length](../computing-embeddings/README.html#id1)
    * [Multi-Process / Multi-GPU Encoding](../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](../semantic-search/README.html)
* [Background](../semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../semantic-search/README.html#speed-optimization)
* [Elasticsearch](../semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../semantic-search/README.html#retrieve-re-rank)
* [Examples](../semantic-search/README.html#examples)
* [Retrieve & Re-Rank](../retrieve_rerank/README.html)
* [Retrieve & Re-Rank Pipeline](../retrieve_rerank/README.html#retrieve-re-rank-pipeline)
* [Retrieval: Bi-Encoder](../retrieve_rerank/README.html#retrieval-bi-encoder)
* [Re-Ranker: Cross-Encoder](../retrieve_rerank/README.html#re-ranker-cross-encoder)
* [Example Scripts](../retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
* [Clustering](../clustering/README.html)

    * [k-Means](../clustering/README.html#k-means)
    * [Agglomerative Clustering](../clustering/README.html#agglomerative-clustering)
    * [Fast Clustering](../clustering/README.html#fast-clustering)
    * [Topic Modeling](../clustering/README.html#topic-modeling)
  * [Paraphrase Mining](../paraphrase-mining/README.html)
    * [`paraphrase_mining()`](../paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
  * [Translated Sentence Mining](../parallel-sentence-mining/README.html)
    * [Margin Based Mining](../parallel-sentence-mining/README.html#margin-based-mining)
    * [Examples](../parallel-sentence-mining/README.html#examples)
  * [Image Search](../image-search/README.html)
    * [Installation](../image-search/README.html#installation)
    * [Usage](../image-search/README.html#usage)
    * [Examples](../image-search/README.html#examples)
  * Embedding Quantization
    * Binary Quantization
    * Scalar (int8) Quantization
    * Additional extensions
    * Demo
    * 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)

* [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](../../training/sts/README.html)
    * [Training data](../../training/sts/README.html#training-data)
    * [Loss Function](../../training/sts/README.html#loss-function)
  * [Natural Language Inference](../../training/nli/README.html)
    * [Data](../../training/nli/README.html#data)
    * [SoftmaxLoss](../../training/nli/README.html#softmaxloss)
    * [MultipleNegativesRankingLoss](../../training/nli/README.html#multiplenegativesrankingloss)
  * [Paraphrase Data](../../training/paraphrases/README.html)
    * [Pre-Trained Models](../../training/paraphrases/README.html#pre-trained-models)
  * [Quora Duplicate Questions](../../training/quora_duplicate_questions/README.html)
    * [Training](../../training/quora_duplicate_questions/README.html#training)
    * [MultipleNegativesRankingLoss](../../training/quora_duplicate_questions/README.html#multiplenegativesrankingloss)
    * [Pretrained Models](../../training/quora_duplicate_questions/README.html#pretrained-models)
  * [MS MARCO](../../training/ms_marco/README.html)
    * [Bi-Encoder](../../training/ms_marco/README.html#bi-encoder)
  * [Matryoshka Embeddings](../../training/matryoshka/README.html)
    * [Use Cases](../../training/matryoshka/README.html#use-cases)
    * [Results](../../training/matryoshka/README.html#results)
    * [Training](../../training/matryoshka/README.html#training)
    * [Inference](../../training/matryoshka/README.html#inference)
    * [Code Examples](../../training/matryoshka/README.html#code-examples)
  * [Adaptive Layers](../../training/adaptive_layer/README.html)
    * [Use Cases](../../training/adaptive_layer/README.html#use-cases)
    * [Results](../../training/adaptive_layer/README.html#results)

    * [Training](../../training/adaptive_layer/README.html#training)
    * [Inference](../../training/adaptive_layer/README.html#inference)
    * [Code Examples](../../training/adaptive_layer/README.html#code-examples)
  * [Multilingual Models](../../training/multilingual/README.html)
    * [Extend your own models](../../training/multilingual/README.html#extend-your-own-models)
    * [Training](../../training/multilingual/README.html#training)
    * [Datasets](../../training/multilingual/README.html#datasets)
    * [Sources for Training Data](../../training/multilingual/README.html#sources-for-training-data)
    * [Evaluation](../../training/multilingual/README.html#evaluation)
      * [Available Pre-trained Models](../../training/multilingual/README.html#available-pre-trained-models)
    * [Usage](../../training/multilingual/README.html#usage)
    * [Performance](../../training/multilingual/README.html#performance)
    * [Citation](../../training/multilingual/README.html#citation)
  * [Model Distillation](../../training/distillation/README.html)
    * [Knowledge Distillation](../../training/distillation/README.html#knowledge-distillation)
      * [Speed - Performance Trade-Off](../../training/distillation/README.html#speed-performance-trade-off)
    * [Dimensionality Reduction](../../training/distillation/README.html#dimensionality-reduction)
    * [Quantization](../../training/distillation/README.html#quantization)
  * [Augmented SBERT](../../training/data_augmentation/README.html)
    * [Motivation](../../training/data_augmentation/README.html#motivation)
      * [Extend to your own datasets](../../training/data_augmentation/README.html#extend-to-your-own-datasets)
    * [Methodology](../../training/data_augmentation/README.html#methodology)
      * [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../../training/data_augmentation/README.html#scenario-1-limited-or-small-annotat

ed-datasets-few-labeled-sentence-pairs)

* [Scenario 2: No annotated datasets (Only unlabeled sentence-pairs)](../../training/data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)

* [Training](../../training/data_augmentation/README.html#training)

* [Citation](../../training/data_augmentation/README.html#citation)

* [Training with Prompts](../../training/prompts/README.html)

* [What are Prompts?](../../training/prompts/README.html#what-are-prompts)

* [Why would we train with Prompts?](../../training/prompts/README.html#why-would-we-train-with-prompts)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

* [Performance Comparison](../../unsupervised_learning/README.html#performance-comparison)
  * [Domain Adaptation](../../domain_adaptation/README.html)
    * [Domain Adaptation vs. Unsupervised Learning](../../domain_adaptation/README.html#domain-adaptation-vs-unsupervised-learning)
      * [Adaptive Pre-Training](../../domain_adaptation/README.html#adaptive-pre-training)
    * [GPL: Generative Pseudo-Labeling](../../domain_adaptation/README.html#gpl-generative-pseudo-labeling)
  * [Hyperparameter Optimization](../../training/hpo/README.html)
    * [HPO Components](../../training/hpo/README.html#hpo-components)
    * [Putting It All Together](../../training/hpo/README.html#putting-it-all-together)
    * [Example Scripts](../../training/hpo/README.html#example-scripts)
  * [Distributed Training](../../../docs/sentence_transformer/training/distributed.html)
    * [Comparison](../../../docs/sentence_transformer/training/distributed.html#comparison)
    * [FSDP](../../../docs/sentence_transformer/training/distributed.html#fsdp)

Cross Encoder

* [Usage](../../../docs/cross_encoder/usage/usage.html)
  * [Retrieve & Re-Rank](../retrieve_rerank/README.html)
    * [Retrieve & Re-Rank Pipeline](../retrieve_rerank/README.html#retrieve-re-rank-pipeline)
    * [Retrieval: Bi-Encoder](../retrieve_rerank/README.html#retrieval-bi-encoder)
    * [Re-Ranker: Cross-Encoder](../retrieve_rerank/README.html#re-ranker-cross-encoder)
    * [Example Scripts](../retrieve_rerank/README.html#example-scripts)
    * [Pre-trained Bi-Encoders (Retrieval)](../retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
    * [Pre-trained Cross-Encoders

(Re-Ranker)](../retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)

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

* [MS MARCO](../../../docs/cross_encoder/pretrained_models.html#ms-marco)

* [SQuAD (QNLI)](../../../docs/cross_encoder/pretrained_models.html#squad-qnli)

* [STSbenchmark](../../../docs/cross_encoder/pretrained_models.html#stsbenchmark)

* [Quora Duplicate Questions](../../../docs/cross_encoder/pretrained_models.html#quora-duplicate-questions)

* [NLI](../../../docs/cross_encoder/pretrained_models.html#nli)

* [Community Models](../../../docs/cross_encoder/pretrained_models.html#community-models)

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

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

* [MS MARCO](../../training/ms_marco/cross_encoder_README.html)

* [Cross-Encoder](../../training/ms_marco/cross_encoder_README.html#cross-encoder)

* [Cross-Encoder Knowledge Distillation](../../training/ms_marco/cross_encoder_README.html#cross-encoder-knowledge-distillation)

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#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#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.paraphr

ase_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)
* [Usage](../../../docs/sentence_transformer/usage/usage.html)
* Embedding Quantization
* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/examples/applications/embedding-quantization/README.md)

* * *

Embedding Quantizationïƒ•

Embeddings may be challenging to scale up, which leads to expensive solutions

and high latencies. Currently, many state-of-the-art models produce embeddings with 1024 dimensions, each of which is encoded in `float32`, i.e., they require 4 bytes per dimension. To perform retrieval over 50 million vectors, you would therefore need around 200GB of memory. This tends to require complex and costly solutions at scale.

However, there is a new approach to counter this problem; it entails reducing the size of each of the individual values in the embedding: **Quantization**. Experiments on quantization have shown that we can maintain a large amount of performance while significantly speeding up computation and saving on memory, storage, and costs.

To learn more about Embedding Quantization and their performance, please read the [blogpost](https://huggingface.co/blog/embedding-quantization) by Sentence Transformers and mixedbread.ai.

Binary Quantizationïƒ•

Binary quantization refers to the conversion of the `float32` values in an embedding to 1-bit values, resulting in a 32x reduction in memory and storage usage. To quantize `float32` embeddings to binary, we simply threshold normalized embeddings at 0: if the value is larger than 0, we make it 1, otherwise we convert it to 0. We can use the Hamming Distance to efficiently perform retrieval with these binary embeddings. This is simply the number of positions at which the bits of two binary embeddings differ. The lower the Hamming Distance, the closer the embeddings, and thus the more relevant the document. A huge advantage of the Hamming Distance is that it can be easily

calculated with 2 CPU cycles, allowing for blazingly fast performance.

[Yamada et al. (2021)](https://arxiv.org/abs/2106.00882) introduced a rescore step, which they called _rerank_ , to boost the performance. They proposed that the `float32` query embedding could be compared with the binary document embeddings using dot-product. In practice, we first retrieve `rescore_multiplier * top_k` results with the binary query embedding and the binary document embeddings â€“ i.e., the list of the first k results of the double-binary retrieval â€“ and then rescore that list of binary document embeddings with the `float32` query embedding.

By applying this novel rescoring step, we are able to preserve up to ~96% of the total retrieval performance, while reducing the memory and disk space usage by 32x and improving the retrieval speed by up to 32x as well.

Binary Quantization in Sentence Transformersï*ƒ*•

Quantizing an embedding with a dimensionality of 1024 to binary would result in 1024 bits. In practice, it is much more common to store bits as bytes instead, so when we quantize to binary embeddings, we pack the bits into bytes using `np.packbits`.

As a result, in practice quantizing a `float32` embedding with a dimensionality of 1024 yields an `int8` or `uint8` embedding with a dimensionality of 128. See two approaches of how you can produce quantized embeddings using Sentence Transformers below:

References

1. [mixedbread-ai/mxbai-embed-large-v1](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1)

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

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

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

```
from sentence_transformers import SentenceTransformer
from sentence_transformers.quantization import quantize_embeddings

# 1. Load an embedding model
```

```
model = SentenceTransformer("mixedbread-ai/mxbai-embed-large-v1")

# 2a. Encode some text using "binary" quantization
binary_embeddings = model.encode(
    ["I am driving to the lake.", "It is a beautiful day."],
    precision="binary",
)

# 2b. or, encode some text without quantization & apply quantization afterwards
embeddings = model.encode(["I am driving to the lake.", "It is a beautiful day."])
binary_embeddings = quantize_embeddings(embeddings, precision="binary")
```

Here you can see the differences between default `float32` embeddings and binary embeddings in terms of shape, size, and `numpy` dtype:

```
>>> embeddings.shape
(2, 1024)
>>> embeddings.nbytes
8192
>>> embeddings.dtype
float32
>>> binary_embeddings.shape
(2, 128)
>>> binary_embeddings.nbytes
```

```
256
>>> binary_embeddings.dtype
int8
```

Note that you can also choose `"ubinary"` to quantize to binary using the unsigned `uint8` data format. This may be a requirement for your vector library/database.

Scalar (int8) Quantizationïƒ•

To convert the `float32` embeddings into `int8`, we use a process called scalar quantization. This involves mapping the continuous range of `float32` values to the discrete set of `int8` values, which can represent 256 distinct levels (from -128 to 127). This is done by using a large calibration dataset of embeddings. We compute the range of these embeddings, i.e. the `min` and `max` of each of the embedding dimensions. From there, we calculate the steps (buckets) in which we categorize each value.

To further boost the retrieval performance, you can optionally apply the same rescoring step as for the binary embeddings. It is important to note here that the calibration dataset has a large influence on the performance, since it defines the buckets.

Scalar Quantization in Sentence Transformersïƒ•

Quantizing an embedding with a dimensionality of 1024 to `int8` results in

1024 bytes. In practice, we can choose either `uint8` or `int8`. This choice is usually made depending on what your vector library/database supports.

In practice, it is recommended to provide the scalar quantization with either:

1. a large set of embeddings to quantize all at once, or

2. `min` and `max` ranges for each of the embedding dimensions, or

3. a large calibration dataset of embeddings from which the `min` and `max` ranges can be computed.

If none of these are the case, you will be given a warning like this:

Computing int8 quantization buckets based on 2 embeddings. int8 quantization is more stable with 'ranges' calculated from more embeddings or a 'calibration_embeddings' that can be used to calculate the buckets.

See how you can produce scalar quantized embeddings using Sentence Transformers below:

References

[mixedbread-ai/mxbai-embed-large-v1](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1
)

2.

[`SentenceTransformer`](../../../docs/package_reference/sentence_transformer/SentenceTransforme
r.html#sentence_transformers.SentenceTransformer
"sentence_transformers.SentenceTransformer")

3.

[`SentenceTransformer.encode`](../../../docs/package_reference/sentence_transformer/SentenceTra
nsformer.html#sentence_transformers.SentenceTransformer.encode
"sentence_transformers.SentenceTransformer.encode")

4.

[`quantize_embeddings()`](../../../docs/package_reference/sentence_transformer/quantization.html#s
entence_transformers.quantization.quantize_embeddings
"sentence_transformers.quantization.quantize_embeddings")

```
from sentence_transformers import SentenceTransformer
from sentence_transformers.quantization import quantize_embeddings
from datasets import load_dataset

# 1. Load an embedding model
model = SentenceTransformer("mixedbread-ai/mxbai-embed-large-v1")
```

```
# 2. Prepare an example calibration dataset
corpus = load_dataset("nq_open", split="train[:1000]")["question"]
calibration_embeddings = model.encode(corpus)

# 3. Encode some text without quantization & apply quantization afterwards
embeddings = model.encode(["I am driving to the lake.", "It is a beautiful day."])
int8_embeddings = quantize_embeddings(
    embeddings,
    precision="int8",
    calibration_embeddings=calibration_embeddings,
)
```

Here you can see the differences between default `float32` embeddings and `int8` scalar embeddings in terms of shape, size, and `numpy` dtype:

```
>>> embeddings.shape
(2, 1024)
>>> embeddings.nbytes
8192
>>> embeddings.dtype
float32
>>> int8_embeddings.shape
(2, 1024)
>>> int8_embeddings.nbytes
```

2048

>>> int8_embeddings.dtype

int8

Combining Binary and Scalar Quantizationïƒ•

It is possible to combine binary and scalar quantization to get the best of both worlds: the extreme speed from binary embeddings and the great performance preservation of scalar embeddings with rescoring. See the demo below for a real-life implementation of this approach involving 41 million texts from Wikipedia. The pipeline for that setup is as follows:

1. The query is embedded using the [`mixedbread-ai/mxbai-embed-large-v1`](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1) SentenceTransformer model.

2. The query is quantized to binary using the [`quantize_embeddings`](../../../docs/package_reference/quantization.html#sentence_transformers.quantization.quantize_embeddings) function from the `sentence-transformers` library.

3. A binary index (41M binary embeddings; 5.2GB of memory/disk space) is searched using the quantized query for the top 40 documents.

4. The top 40 documents are loaded on the fly from an int8 index on disk (41M int8 embeddings; 0 bytes of memory, 47.5GB of disk space).

5. The top 40 documents are rescored using the float32 query and the int8 embeddings to get the top 10 documents.

6. The top 10 documents are sorted by score and displayed.

Through this approach, we use 5.2GB of memory and 52GB of disk space for the indices. This is considerably less than normal retrieval, for which we would require 200GB of memory and 200GB of disk space. Especially as you scale up even further, this will result in notable reductions in both latency and costs.

Additional extensionsïƒ•

Note that embedding quantization can be combined with other approaches to improve retrieval efficiency, such as [Matryoshka Embeddings](../../training/matryoshka/README.html). Additionally, the [Retrieve & Re-Rank](../retrieve_rerank/README.html) also works very well with quantized embeddings, i.e. you can still use a Cross-Encoder to rerank.

Demoïƒ•

The following demo showcases the retrieval efficiency using `exact` search through combining binary search with scalar (`int8`) rescoring. The solution requires 5GB of memory for the binary index and 50GB of disk space for the binary and scalar indices, considerably less than the 200GB of memory and disk space which would be required for regular `float32` retrieval. Additionally, retrieval is much faster.

Try it yourself!

The following scripts can be used to experiment with embedding quantization for retrieval & beyond. There are three categories:

* **Recommended Retrieval** :

*

[semantic_search_recommended.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/embedding-quantization/semantic_search_recommended.py): This script combines binary search with scalar rescoring, much like the above demo, for cheap, efficient, and performant retrieval.

* **Usage** :

*

[semantic_search_faiss.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/embedding-quantization/semantic_search_faiss.py): This script showcases regular usage of binary or scalar quantization, retrieval, and rescoring using FAISS, by using the [`semantic_search_faiss`](../../../docs/package_reference/quantization.html#sentence_transformers.quantization.semantic_search_faiss) utility function.

*

[semantic_search_usearch.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/embedding-quantization/semantic_search_usearch.py): This script showcases regular usage of binary or scalar quantization, retrieval, and rescoring using USearch, by using the

[`semantic_search_usearch`](../../../docs/package_reference/quantization.html#sentence_transformers.quantization.semantic_search_usearch) utility function.

* **Benchmarks** :

*

[semantic_search_faiss_benchmark.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/embedding-quantization/semantic_search_faiss_benchmark.py): This script includes a retrieval speed benchmark of `float32` retrieval, binary retrieval + rescoring, and scalar retrieval + rescoring, using FAISS. It uses the [`semantic_search_faiss`](../../../docs/package_reference/quantization.html#sentence_transformers.quantization.semantic_search_faiss) utility function. Our benchmarks especially show show speedups for `ubinary`.

*

[semantic_search_usearch_benchmark.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/applications/embedding-quantization/semantic_search_usearch_benchmark.py): This script includes a retrieval speed benchmark of `float32` retrieval, binary retrieval + rescoring, and scalar retrieval + rescoring, using USearch. It uses the [`semantic_search_usearch`](../../../docs/package_reference/quantization.html#sentence_transformers.quantization.semantic_search_usearch) utility function. Our experiments show large speedups on newer hardware, particularly for `int8`.

[Previous](../image-search/README.html "Image Search") [Next](../../../docs/sentence_transformer/usage/efficiency.html "Speeding up Inference")

* * *

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).