[![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)

    * [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)

* Multilingual Models

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

* * *

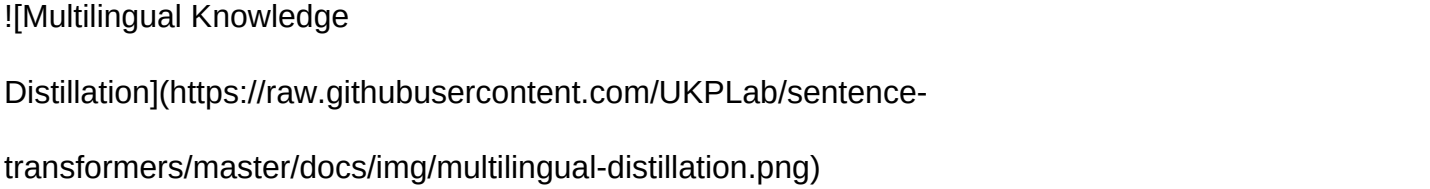
Multilingual Modelsïƒ•

The issue with multilingual BERT (mBERT) as well as with XLM-RoBERTa is that those produce rather bad sentence representation out-of-the-box. Further, the vectors spaces between languages are not aligned, i.e., the sentences with the same content in different languages would be mapped to different locations in the vector space.

In my publication [Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation](https://arxiv.org/abs/2004.09813) I describe an easy approach to extend sentence embeddings to further languages.

Chien Vu also wrote a nice blog article on this technique: [A complete guide to transfer learning from English to other Languages using Sentence Embeddings BERT Models](https://towardsdatascience.com/a-complete-guide-to-transfer-learning-from-english-to-other-languages-using-sentence-embeddings-8c427f8804a9)

Extend your own modelsïƒ•



The idea is based on a fixed (monolingual) **teacher model** that produces sentence embeddings with our desired properties in one language (e.g. English). The **student model** is supposed to mimic the teacher model, i.e., the same English sentence should be mapped to the same vector by the teacher and by the student model. Additionally, in order to make the student model work for other languages, we train the student model on parallel (translated) sentences. The translation of each sentence should also be mapped to the same vector as the original sentence.

In the above figure, the student model should map _Hello World_ and the German translation _Hallo Welt_ to the vector of `teacher_model('Hello World')`. We achieve this by training the student model using mean squared error (MSE) loss.

In our experiments we initialized the student model with the multilingual [XLM-RoBERTa model](https://huggingface.co/FacebookAI/xlm-roberta-base).

Trainingïƒ•

For a **fully automatic code example** , see [make_multilingual.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/multilingual/make_multilingual.py).

This scripts downloads the parallel sentences corpus, a corpus with
transcripts and translations from talks. It than extends a monolingual model
to several languages (en, de, es, it, fr, ar, tr). This corpus contains
parallel data for more than 100 languages, hence, you can simple change the
script and train a multilingual model in your favorite languages.

Datasetsïƒ•

As training data we require parallel sentences, i.e., sentences translated in
various languages. In particular, we will use
[`Dataset`](https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.Dataset
"\(in datasets vmain\)") instances with `"english"` and `"non_english"`
columns. We have prepared a large collection of such datasets in our [Parallel
Sentences dataset collection](https://huggingface.co/collections/sentence-transformers/parallel-sentences-datasets-6644d644123d31ba5b1c8785).

The training script will take the `"english"` column and add a `"label"`
column containing the embeddings of the english texts. Then, the student model
`"english"` and `"non_english"` will be trained to be similar to this
`"label"`. You can load such a training dataset like so:

```
from datasets import load_dataset

train_dataset = load_dataset("sentence-transformers/parallel-sentences-talks", "en-de",
```

split="train")

print(train_dataset[0])

{"english": "So I think practicality is one case where it's worth teaching people by hand.", "non_english": "Ich denke, dass es sich aus diesem Grund lohnt, den Leuten das Rechnen von Hand beizubringen."}

Sources for Training Dataïƒ•

A great website for a vast number of parallel (translated) datasets is [OPUS](http://opus.nlpl.eu/). There, you find parallel datasets for more than 400 languages. You can use these to create your own parallel sentence datasets, if you wish.

Evaluationïƒ•

Training can be evaluated in different ways. For an example how to use these evaluation methods, see [make_multilingual.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/multilingual/make_multilingual.py).

MSE Evaluationïƒ•

You can measure the mean squared error (MSE) between the student embeddings and teacher embeddings.

```
from datasets import load_dataset

eval_dataset = load_dataset("sentence-transformers/parallel-sentences-talks", "en-fr", split="dev")

dev_mse = MSEEvaluator(
    source_sentences=eval_dataset["english"],
    target_sentences=eval_dataset["non_english"],
    name="en-fr-dev",
    teacher_model=teacher_model,
    batch_size=32,
)
```

This evaluator computes the teacher embeddings for the `source_sentences`, for example, for English. During training, the student model is used to compute embeddings for the `target_sentences`, for example, for French. The distance between teacher and student embeddings is measures. Lower scores indicate a better performance.

Translation Accuracyïƒ•

You can also measure the translation accuracy. As inputs, this evaluator accepts a list of `source_sentences` (e.g. English), and a list of `target_sentences` (e.g. Spanish), such that `target_sentences[i]` is a translation of `source_sentences[i]`.

For each sentence pair, we check if `source_sentences[i]` we check if
`target_sentences[i]` has the highest similarity out of all target sentences.
If this is the case, we have a hit, otherwise an error. This evaluator reports
accuracy (higher = better).

```
from datasets import load_dataset

eval_dataset = load_dataset("sentence-transformers/parallel-sentences-talks", "en-fr", split="dev")

dev_trans_acc = TranslationEvaluator(
    source_sentences=eval_dataset["english"],
    target_sentences=eval_dataset["non_english"],
    name="en-fr-dev",
    batch_size=32,
)
```

Multilingual Semantic Textual Similarityïƒ•

You can also measure the semantic textual similarity (STS) between sentence
pairs in different languages:

```
from datasets import load_dataset
```

```
    test_dataset = load_dataset("mteb/sts17-crosslingual-sts", "nl-en", split="test")

    test_emb_similarity = EmbeddingSimilarityEvaluator(
        sentences1=test_dataset["sentence1"],
        sentences2=test_dataset["sentence2"],
        scores=[score / 5.0 for score in test_dataset["score"]],  # Convert 0-5 scores to 0-1 scores
        batch_size=32,
        name=f"sts17-nl-en-test",
        show_progress_bar=False,
    )
```

Where `sentences1` and `sentences2` are lists of sentences and score is numeric value indicating the semantic similarity between `sentences1[i]` and `sentences2[i]`.

Available Pre-trained Modelsïƒ•

For a list of available models, see [Pretrained Models](../../../docs/sentence_transformer/pretrained_models.html#multilingual-models).

Usageïƒ•

You can use the models in the following way:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")
embeddings = model.encode(["Hello World", "Hallo Welt", "Hola mundo", "Bye, Moon!"])
similarities = model.similarity(embeddings, embeddings)
# tensor([[1.0000, 0.9429, 0.8880, 0.4558],
#         [0.9429, 1.0000, 0.9680, 0.5307],
#         [0.8880, 0.9680, 1.0000, 0.4933],
#         [0.4558, 0.5307, 0.4933, 1.0000]])
```

Performanceïƒ•

The performance was evaluated on the [Semantic Textual Similarity (STS) 2017 dataset](http://ixa2.si.ehu.es/stswiki/index.php/Main_Page). The task is to predict the semantic similarity (on a scale 0-5) of two given sentences.
STS2017 has monolingual test data for English, Arabic, and Spanish, and cross-lingual test data for English-Arabic, -Spanish and -Turkish.

We extended the STS2017 and added cross-lingual test data for English-German, French-English, Italian-English, and Dutch-English ([STS2017-extended.zip](https://public.ukp.informatik.tu-darmstadt.de/reimers/sentence-transformers/datasets/STS2017-extended.zip)).
The performance is measured using Spearman correlation between the predicted similarity score and the gold score.

Model | AR-AR | AR-EN | ES-ES | ES-EN | EN-EN | TR-EN | EN-DE | FR-EN | IT-EN | NL-EN | Average

---|---|---|---|---|---|---|---|---|---|---|---

XLM-RoBERTa mean pooling | 25.7 | 17.4 | 51.8 | 10.9 | 50.7 | 9.2 | 21.3 | 16.6 | 22.9 | 26.0 | 25.2

mBERT mean pooling | 50.9 | 16.7 | 56.7 | 21.5 | 54.4 | 16.0 | 33.9 | 33.0 | 34.0 | 35.6 | 35.3

LASER | 68.9 | 66.5 | 79.7 | 57.9 | 77.6 | 72.0 | 64.2 | 69.1 | 70.8 | 68.5 | 69.5

Sentence Transformer Models

distiluse-base-multilingual-cased | 75.9 | 77.6 | 85.3 | 78.7 | 85.4 | 75.5 | 80.3 | 80.2 | 80.5 | 81.7 | 80.1

Citationï ƒ•

If you use the code for multilingual models, feel free to cite our publication [Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation](https://arxiv.org/abs/2004.09813):

```
@article{reimers-2020-multilingual-sentence-bert,
    title = "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation",
    author = "Reimers, Nils and Gurevych, Iryna",
    journal= "arXiv preprint arXiv:2004.09813",
    month = "04",
    year = "2020",
    url = "http://arxiv.org/abs/2004.09813",
}
```

[Previous](../adaptive_layer/README.html "Adaptive Layers") [Next](../distillation/README.html "Model Distillation")

* * *