[![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)

Cross Encoder

Package Reference

* [Sentence Transformer](../../../docs/package_reference/sentence_transformer/index.html)
*
[SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html)
*
[SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#id1)
*
[SentenceTransformerModelCardData](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentencetransformermodelcarddata)
*
[SimilarityFunction](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#similarityfunction)
* [Trainer](../../../docs/package_reference/sentence_transformer/trainer.html)
*
[SentenceTransformerTrainer](../../../docs/package_reference/sentence_transformer/trainer.html#sentencetransformertrainer)
* [Training Arguments](../../../docs/package_reference/sentence_transformer/training_args.html)
*
[SentenceTransformerTrainingArguments](../../../docs/package_reference/sentence_transformer/training_args.html#sentencetransformertrainingarguments)
* [Losses](../../../docs/package_reference/sentence_transformer/losses.html)
*
[BatchAllTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchalltripletloss)
*

[BatchHardSoftMarginTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardsoftmargintripletloss)

*

[BatchHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardtripletloss)

*

[BatchSemiHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchsemihardtripletloss)

*

[ContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastiveloss)

*

[OnlineContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#onlinecontrastiveloss)

*

[ContrastiveTensionLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionloss)

*

[ContrastiveTensionLossInBatchNegatives](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionlossinbatchnegatives)
    * [CoSENTLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosentloss)
    * [AnglELoss](../../../docs/package_reference/sentence_transformer/losses.html#angleloss)

*

[CosineSimilarityLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosinesimilarityloss)

*

[DenoisingAutoEncoderLoss](../../../docs/package_reference/sentence_transformer/losses.html#denoisingautoencoderloss)

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

s.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_train

ing_available)

*

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

*

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

*

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

*

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

*

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

* [Model Optimization](../../../docs/package_reference/util.html#module-sentence_transformers.backend)

*

[`export_dynamic_quantized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)

*

[`export_optimized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model)

*

[`export_static_quantized_openvino_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_static_quantized_openvino_model)

* [Similarity Metrics](../../../docs/package_reference/util.html#module-sentence_transformers.util)
  * [`cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.cos_sim)
  * [`dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.dot_score)
  * [`euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.euclidean_sim)
  * [`manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.manhattan_sim)
  * [`pairwise_cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_cos_sim)
  * [`pairwise_dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_dot_score)
  * [`pairwise_euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_euclidean_sim)
  * [`pairwise_manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_manhattan_sim)

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

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

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

* * *

Training with Promptsïƒ•

What are Prompts?ïƒ•

Many modern embedding models are trained with â€œinstructionsâ€• or â€œpromptsâ€• following the [INSTRUCTOR paper](https://arxiv.org/abs/2212.09741). These prompts are strings, prefixed to each text to be embedded, allowing the model to distinguish between different types of text.

For example, the [mixedbread-ai/mxbai-embed-large-v1](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1) model was trained with `Represent this sentence for searching relevant passages: ` as the prompt for all queries. This prompt is stored in the [model configuration](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1/blob/main/config_sentence_transformers.json) under the prompt name `"query"`, so users can specify that `prompt_name` in `model.encode`:

```
from sentence_transformers import SentenceTransformer
```

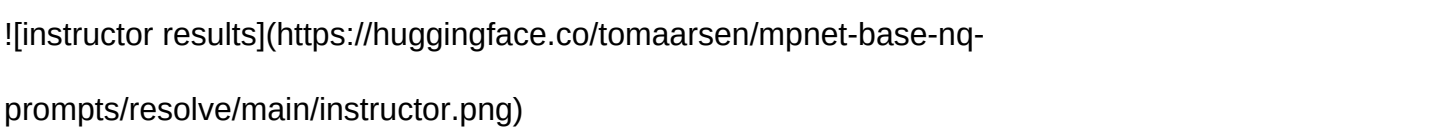
```
model = SentenceTransformer("mixedbread-ai/mxbai-embed-large-v1")
query_embedding = model.encode("What are Pandas?", prompt_name="query")
# or
# query_embedding = model.encode("What are Pandas?", prompt="Represent this sentence for searching relevant passages: ")
document_embeddings = model.encode([
    "Pandas is a software library written for the Python programming language for data manipulation and analysis.",
    "Pandas are a species of bear native to South Central China. They are also known as the giant panda or simply panda.",
    "Koala bears are not actually bears, they are marsupials native to Australia.",
])
similarity = model.similarity(query_embedding, document_embeddings)
print(similarity)
# => tensor([[0.7594, 0.7560, 0.4674]])
```

See [Prompt Templates](https://sbert.net/examples/applications/computing-embeddings/README.html#prompt-templates) for more information about inference with prompts.

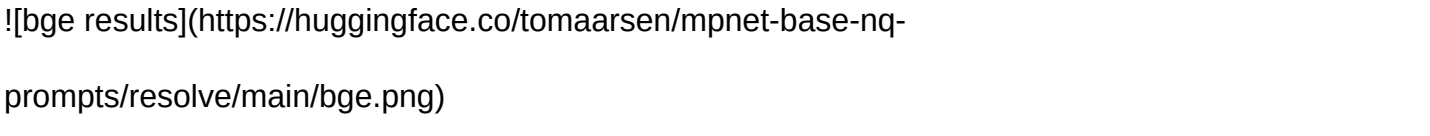
Why would we train with Prompts?ïƒ•

The [INSTRUCTOR paper](https://arxiv.org/abs/2212.09741) showed that adding prompts or instructions before each text could improve model performance by an average of ~6%, with major gains especially for classification, clustering,

and semantic textual similarity. Note that the performance improvements for retrieval are notably lower at 0.4% and 2.7% for small and large models, respectively.



More recently, the [BGE paper](https://arxiv.org/pdf/2309.07597) showed similar findings, showing about a 1.4% performance increase for retrieval if the query is prefixed with `Represent this sentence for searching relevant passages: `. The authors conclude that using instructions may substantially contribute to the quality of task-specific fine-tuning.



In essence, using instructions or prompts allows for improved performance as long as they are used both during training and inference.

How do we train with Prompts?ïƒ•

Since the v3.3.0 Sentence Transformers release, itâ€™s possible to finetune embedding models with prompts using the `prompts` argument in the [`SentenceTransformerTrainingArguments`](../../../docs/package_reference/sentence_transformer/training_args.html#sentence_transformers.training_args.SentenceTransformerTrainingArguments "sentence_transformers.training_args.SentenceTransformerTrainingArguments") class. There are 4 separate accepted formats for this argument:

1. `str`: A single prompt to use for all columns in all datasets. For example:

```
args = SentenceTransformerTrainingArguments(
...,
prompts="text: ",
...,
)
```

2. `Dict[str, str]`: A dictionary mapping column names to prompts, applied to all datasets. For example:

```
args = SentenceTransformerTrainingArguments(
...,
prompts={
    "query": "query: ",
    "answer": "document: ",
},
...,
)
```

3. `Dict[str, str]`: A dictionary mapping dataset names to prompts. This should only be used if your training/evaluation/test datasets are a [`DatasetDict`](https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.DatasetDict "\(in datasets vmain\)") or a dictionary of

[`Dataset`](https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.Dataset "\(in datasets vmain\)"). For example:

```
    args = SentenceTransformerTrainingArguments(
    ...,
    prompts={
        "stsb": "Represent this text for semantic similarity search: ",
        "nq": "Represent this text for retrieval: ",
    },
    ...,
)
```

4. `Dict[str, Dict[str, str]]`: A dictionary mapping dataset names to dictionaries mapping column names to prompts. This should only be used if your training/evaluation/test datasets are a [`DatasetDict`](https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.DatasetDict "\(in datasets vmain\)") or a dictionary of [`Dataset`](https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.Dataset "\(in datasets vmain\)"). For example:

```
    args = SentenceTransformerTrainingArguments(
    ...,
    prompts={
        "stsb": {
            "sentence1": "sts: ",
            "sentence2": "sts: ",
        },
```

```
        "nq": {
            "query": "query: ",
            "document": "document: ",
        },
    },
    ...,
)
```

Additionally, some research papers ([INSTRUCTOR](https://arxiv.org/abs/2212.09741), [NV-Embed](https://arxiv.org/pdf/2405.17428)) exclude the prompt from the mean pooling step, such that itâ€™s only used in the Transformer blocks. In Sentence Transformers, this can be configured with the `include_prompt` argument/attribute in the [`Pooling`](../../../docs/package_reference/sentence_transformer/models.html#sentence_transformers.models.Pooling "sentence_transformers.models.Pooling") module or via the [`SentenceTransformer.set_pooling_include_prompt()`](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.set_pooling_include_prompt "sentence_transformers.SentenceTransformer.set_pooling_include_prompt") method. In my personal experience, models that include the prompt in the pooling tend to perform better.

Training Scriptï•

See the following script as an example of how to train with prompts in practice:

*

[training_nq_prompts.py](https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/prompts/training_nq_prompts.py): This script finetunes [mpnet-base](https://huggingface.co/microsoft/mpnet-base) on 100k query-answer pairs from the [natural-questions](https://huggingface.co/datasets/sentence-transformers/natural-questions) dataset using the [`CachedMultipleNegativesRankingLoss`](../../../docs/package_reference/sentence_transformer/losses.html#sentence_transformers.losses.CachedMultipleNegativesRankingLoss "sentence_transformers.losses.CachedMultipleNegativesRankingLoss") loss. The model is evaluated during training using the [`NanoBEIREvaluator`](../../../docs/package_reference/sentence_transformer/evaluation.html#sentence_transformers.evaluation.NanoBEIREvaluator "sentence_transformers.evaluation.NanoBEIREvaluator").

This script has two variables that affect 1) whether prompts are used and 2) whether prompts are included in the pooling. I have finetuned both `mpnet-base` and `bert-base-uncased` under the various different settings, resulting in a 0.66% and 0.90% relative improvements on `NDCG@10` at no extra cost.

Experiments with `mpnet-base`

Running the script under various settings resulted in these checkpoints:

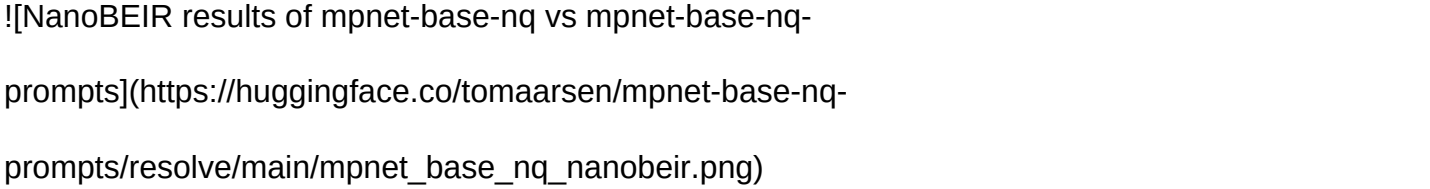
* [tomaarsen/mpnet-base-nq](https://huggingface.co/tomaarsen/mpnet-base-nq)

* [tomaarsen/mpnet-base-nq-prompts](https://huggingface.co/tomaarsen/mpnet-base-nq-prompts)

Note

`mpnet-base` seems to be a tad unstable when training with prompts and excluding those prompts in the pooling: the loss spikes at some point, an effect not observed with e.g. `bert-base-uncased`.

For these two models, the model trained with prompts consistently outperforms the baseline model all throughout training:



Additionally, the model trained with prompts includes these prompts in the training dataset details in the automatically generated model card: [tomaarsen/mpnet-base-nq-prompts#natural-questions](https://huggingface.co/tomaarsen/mpnet-base-nq-prompts#natural-questions).

Important

If you train with prompts, then itâ€™s heavily recommended to store prompts in the model configuration as a mapping of prompt names to prompt strings. You can do this by initializing the

[`SentenceTransformer`](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer "sentence_transformers.SentenceTransformer") with a `prompts` dictionary before saving it, updating the `model.prompts` of a loaded model before saving it, and/or updating the [config_sentence_transformers.json](https://huggingface.co/tomaarsen/mpnet-base-nq-prompts/blob/main/config_sentence_transformers.json) file of the saved model.

After adding the prompts in the model configuration, the final usage of the prompt-trained model becomes:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("tomaarsen/mpnet-base-nq-prompts")
query_embedding = model.encode("What are Pandas?", prompt_name="query")
document_embeddings = model.encode([
    "Pandas is a software library written for the Python programming language for data manipulation and analysis.",
    "Pandas are a species of bear native to South Central China. They are also known as the giant panda or simply panda.",
    "Koala bears are not actually bears, they are marsupials native to Australia.",
    ],
    prompt_name="document",
)
```

```
similarity = model.similarity(query_embedding, document_embeddings)
print(similarity)
# => tensor([[0.4725, 0.7339, 0.4369]])
```

Experiments with `bert-base-uncased`

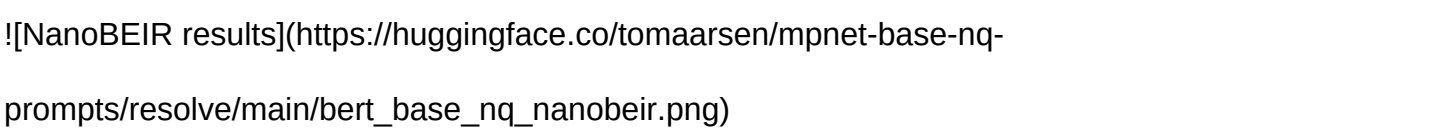
Running the script under various settings resulted in these checkpoints:

* [tomaarsen/bert-base-nq](https://huggingface.co/tomaarsen/bert-base-nq)

* [tomaarsen/bert-base-nq-prompts](https://huggingface.co/tomaarsen/bert-base-nq-prompts)

* [tomaarsen/bert-base-nq-prompts-exclude-pooling-prompts](https://huggingface.co/tomaarsen/bert-base-nq-prompts-exclude-pooling-prompts)

For these three models, the model trained with prompts consistently outperforms the baseline model all throughout training, except for the very first evaluation. The model that excludes the prompt in the mean pooling consistently performs notably worse than either of the other two.



Additionally, the model trained with prompts includes these prompts in the training dataset details in the automatically generated model card:

[tomaarsen/bert-base-nq-prompts#natural-
questions](https://huggingface.co/tomaarsen/bert-base-nq-prompts#natural-
questions).

Important

If you train with prompts, then itâ€™s heavily recommended to store prompts in
the model configuration as a mapping of prompt names to prompt strings. You
can do this by initializing the
[`SentenceTransformer`](../../../docs/package_reference/sentence_transformer/SentenceTransforme
r.html#sentence_transformers.SentenceTransformer
"sentence_transformers.SentenceTransformer") with a `prompts` dictionary
before saving it, updating the `model.prompts` of a loaded model before saving
it, and/or updating the
[config_sentence_transformers.json](https://huggingface.co/tomaarsen/mpnet-
base-nq-prompts/blob/main/config_sentence_transformers.json) file of the saved
model.

After adding the prompts in the model configuration, the final usage of the
prompt-trained model becomes:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("tomaarsen/bert-base-nq-prompts")
query_embedding = model.encode("What are Pandas?", prompt_name="query")
```

```
    document_embeddings = model.encode([
            "Pandas is a software library written for the Python programming language for data manipulation and analysis.",
        "Pandas are a species of bear native to South Central China. They are also known as the giant panda or simply panda.",
        "Koala bears are not actually bears, they are marsupials native to Australia.",
        ],
        prompt_name="document",
    )
    similarity = model.similarity(query_embedding, document_embeddings)
    print(similarity)
    # => tensor([[0.3955, 0.8226, 0.5706]])
```

[Previous](../data_augmentation/README.html "Augmented SBERT") [Next](../peft/README.html "Training with PEFT Adapters")

* * *