**[Hugging Face](https://huggingface.co/)**

**Model List**

bge is short for BAAI general embedding.

| **Model** | **Language** |  | **Description** | **query instruction for retrieval [1]** |
| --- | --- | --- | --- | --- |
| [BAAI/bge-m3](https://huggingface.co/BAAI/bge-m3) | Multilingual | [Inference](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3#usage) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3) | Multi-Functionality(dense retrieval, sparse retrieval, multi-vector(colbert)), Multi-Linguality, and Multi-Granularity(8192 tokens) |  |
| [BAAI/llm-embedder](https://huggingface.co/BAAI/llm-embedder) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5/blob/main/FlagEmbedding/llm_embedder/README.md) [Fine-tune](https://huggingface.co/BAAI/bge-large-zh-v1.5/blob/main/FlagEmbedding/llm_embedder/README.md) | a unified embedding model to support diverse retrieval augmentation needs for LLMs | See [README](https://huggingface.co/BAAI/bge-large-zh-v1.5/blob/main/FlagEmbedding/llm_embedder/README.md) |
| [BAAI/bge-reranker-large](https://huggingface.co/BAAI/bge-reranker-large) | Chinese and English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-reranker) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/reranker) | a cross-encoder model which is more accurate but less efficient [2] |  |
| [BAAI/bge-reranker-base](https://huggingface.co/BAAI/bge-reranker-base) | Chinese and English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-reranker) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/reranker) | a cross-encoder model which is more accurate but less efficient [2] |  |
| [BAAI/bge-large-en-v1.5](https://huggingface.co/BAAI/bge-large-en-v1.5) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | Represent this sentence for searching relevant passages: |
| [BAAI/bge-base-en-v1.5](https://huggingface.co/BAAI/bge-base-en-v1.5) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | Represent this sentence for searching relevant passages: |
| [BAAI/bge-small-en-v1.5](https://huggingface.co/BAAI/bge-small-en-v1.5) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | Represent this sentence for searching relevant passages: |
| [BAAI/bge-large-zh-v1.5](https://huggingface.co/BAAI/bge-large-zh-v1.5) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | 为这个句子生成表示以用于检索相关文章： |
| [BAAI/bge-base-zh-v1.5](https://huggingface.co/BAAI/bge-base-zh-v1.5) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | 为这个句子生成表示以用于检索相关文章： |
| [BAAI/bge-small-zh-v1.5](https://huggingface.co/BAAI/bge-small-zh-v1.5) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | version 1.5 with more reasonable similarity distribution | 为这个句子生成表示以用于检索相关文章： |
| [BAAI/bge-large-en](https://huggingface.co/BAAI/bge-large-en) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | :trophy: rank **1st** in [MTEB](https://huggingface.co/spaces/mteb/leaderboard) leaderboard | Represent this sentence for searching relevant passages: |
| [BAAI/bge-base-en](https://huggingface.co/BAAI/bge-base-en) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | a base-scale model but with similar ability to bge-large-en | Represent this sentence for searching relevant passages: |
| [BAAI/bge-small-en](https://huggingface.co/BAAI/bge-small-en) | English | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | a small-scale model but with competitive performance | Represent this sentence for searching relevant passages: |
| [BAAI/bge-large-zh](https://huggingface.co/BAAI/bge-large-zh) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | :trophy: rank **1st** in [C-MTEB](https://github.com/FlagOpen/FlagEmbedding/tree/master/C_MTEB) benchmark | 为这个句子生成表示以用于检索相关文章： |
| [BAAI/bge-base-zh](https://huggingface.co/BAAI/bge-base-zh) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | a base-scale model but with similar ability to bge-large-zh | 为这个句子生成表示以用于检索相关文章： |
| [BAAI/bge-small-zh](https://huggingface.co/BAAI/bge-small-zh) | Chinese | [Inference](https://huggingface.co/BAAI/bge-large-zh-v1.5#usage-for-embedding-model) [Fine-tune](https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/finetune) | a small-scale model but with competitive performance | 为这个句子生成表示以用于检索相关文章： |

[1]: If you need to search the relevant passages to a query, we suggest to add the instruction to the query; in other cases, no instruction is needed, just use the original query directly. In all cases, **no instruction** needs to be added to passages.

[2]: Different from embedding model, reranker uses question and document as input and directly output similarity instead of embedding. To balance the accuracy and time cost, cross-encoder is widely used to re-rank top-k documents retrieved by other simple models. For examples, use bge embedding model to retrieve top 100 relevant documents, and then use bge reranker to re-rank the top 100 document to get the final top-3 results.

at <https://huggingface.co/BAAI>. If you cannot open the Huggingface Hub, you also can download the models at <https://model.baai.ac.cn/models> .

**Frequently asked questions**

1. How to fine-tune bge embedding model?

2. The similarity score between two dissimilar sentences is higher than 0.5

3. When does the query instruction need to be used

**Usage**

**Usage for Embedding Model**

Here are some examples for using bge models with [FlagEmbedding](https://huggingface.co/BAAI/bge-large-zh-v1.5" \l "using-flagembedding), [Sentence-Transformers](https://huggingface.co/BAAI/bge-large-zh-v1.5#using-sentence-transformers), [Langchain](https://huggingface.co/BAAI/bge-large-zh-v1.5" \l "using-langchain), or [Huggingface Transformers](https://huggingface.co/BAAI/bge-large-zh-v1.5" \l "using-huggingface-transformers).

**Using FlagEmbedding**

pip install -U FlagEmbedding

If it doesn't work for you, you can see [FlagEmbedding](https://github.com/FlagOpen/FlagEmbedding/blob/master/FlagEmbedding/baai_general_embedding/README.md) for more methods to install FlagEmbedding.

from FlagEmbedding import FlagModel

sentences\_1 = ["样例数据-1", "样例数据-2"]

sentences\_2 = ["样例数据-3", "样例数据-4"]

model = FlagModel('BAAI/bge-large-zh-v1.5',

query\_instruction\_for\_retrieval="为这个句子生成表示以用于检索相关文章：",

use\_fp16=True) *# Setting use\_fp16 to True speeds up computation with a slight performance degradation*

embeddings\_1 = model.encode(sentences\_1)

embeddings\_2 = model.encode(sentences\_2)

similarity = embeddings\_1 @ embeddings\_2.T

print(similarity)

*# for s2p(short query to long passage) retrieval task, suggest to use encode\_queries() which will automatically add the instruction to each query*

*# corpus in retrieval task can still use encode() or encode\_corpus(), since they don't need instruction*

queries = ['query\_1', 'query\_2']

passages = ["样例文档-1", "样例文档-2"]

q\_embeddings = model.encode\_queries(queries)

p\_embeddings = model.encode(passages)

scores = q\_embeddings @ p\_embeddings.T

For the value of the argument query\_instruction\_for\_retrieval, see [Model List](https://github.com/FlagOpen/FlagEmbedding/tree/master#model-list).

By default, FlagModel will use all available GPUs when encoding. Please set os.environ["CUDA\_VISIBLE\_DEVICES"] to select specific GPUs. You also can set os.environ["CUDA\_VISIBLE\_DEVICES"]="" to make all GPUs unavailable.

**Using Sentence-Transformers**

You can also use the bge models with [sentence-transformers](https://www.sbert.net/):

pip install -U sentence-transformers

from sentence\_transformers import SentenceTransformer

sentences\_1 = ["样例数据-1", "样例数据-2"]

sentences\_2 = ["样例数据-3", "样例数据-4"]

model = SentenceTransformer('BAAI/bge-large-zh-v1.5')

embeddings\_1 = model.encode(sentences\_1, normalize\_embeddings=True)

embeddings\_2 = model.encode(sentences\_2, normalize\_embeddings=True)

similarity = embeddings\_1 @ embeddings\_2.T

print(similarity)

For s2p(short query to long passage) retrieval task, each short query should start with an instruction (instructions see [Model List](https://github.com/FlagOpen/FlagEmbedding/tree/master#model-list)). But the instruction is not needed for passages.

from sentence\_transformers import SentenceTransformer

queries = ['query\_1', 'query\_2']

passages = ["样例文档-1", "样例文档-2"]

instruction = "为这个句子生成表示以用于检索相关文章："

model = SentenceTransformer('BAAI/bge-large-zh-v1.5')

q\_embeddings = model.encode([instruction+q for q in queries], normalize\_embeddings=True)

p\_embeddings = model.encode(passages, normalize\_embeddings=True)

scores = q\_embeddings @ p\_embeddings.T

**Using Langchain**

You can use bge in langchain like this:

from langchain.embeddings import HuggingFaceBgeEmbeddings

model\_name = "BAAI/bge-large-en-v1.5"

model\_kwargs = {'device': 'cuda'}

encode\_kwargs = {'normalize\_embeddings': True} *# set True to compute cosine similarity*

model = HuggingFaceBgeEmbeddings(

model\_name=model\_name,

model\_kwargs=model\_kwargs,

encode\_kwargs=encode\_kwargs,

query\_instruction="为这个句子生成表示以用于检索相关文章："

)

model.query\_instruction = "为这个句子生成表示以用于检索相关文章："

**Using HuggingFace Transformers**

With the transformers package, you can use the model like this: First, you pass your input through the transformer model, then you select the last hidden state of the first token (i.e., [CLS]) as the sentence embedding.

from transformers import AutoTokenizer, AutoModel

import torch

*# Sentences we want sentence embeddings for*

sentences = ["样例数据-1", "样例数据-2"]

*# Load model from HuggingFace Hub*

tokenizer = AutoTokenizer.from\_pretrained('BAAI/bge-large-zh-v1.5')

model = AutoModel.from\_pretrained('BAAI/bge-large-zh-v1.5')

model.eval()

*# Tokenize sentences*

encoded\_input = tokenizer(sentences, padding=True, truncation=True, return\_tensors='pt')

*# for s2p(short query to long passage) retrieval task, add an instruction to query (not add instruction for passages)*

*# encoded\_input = tokenizer([instruction + q for q in queries], padding=True, truncation=True, return\_tensors='pt')*

*# Compute token embeddings*

with torch.no\_grad():

model\_output = model(\*\*encoded\_input)

*# Perform pooling. In this case, cls pooling.*

sentence\_embeddings = model\_output[0][:, 0]

*# normalize embeddings*

sentence\_embeddings = torch.nn.functional.normalize(sentence\_embeddings, p=2, dim=1)

print("Sentence embeddings:", sentence\_embeddings)

**Usage for Reranker**

Different from embedding model, reranker uses question and document as input and directly output similarity instead of embedding. You can get a relevance score by inputting query and passage to the reranker. The reranker is optimized based cross-entropy loss, so the relevance score is not bounded to a specific range.

**Using FlagEmbedding**

pip install -U FlagEmbedding

Get relevance scores (higher scores indicate more relevance):

from FlagEmbedding import FlagReranker

reranker = FlagReranker('BAAI/bge-reranker-large', use\_fp16=True) *# Setting use\_fp16 to True speeds up computation with a slight performance degradation*

score = reranker.compute\_score(['query', 'passage'])

print(score)

scores = reranker.compute\_score([['what is panda?', 'hi'], ['what is panda?', 'The giant panda (Ailuropoda melanoleuca), sometimes called a panda bear or simply panda, is a bear species endemic to China.']])

print(scores)

**Using Huggingface transformers**

import torch

from transformers import AutoModelForSequenceClassification, AutoTokenizer

tokenizer = AutoTokenizer.from\_pretrained('BAAI/bge-reranker-large')

model = AutoModelForSequenceClassification.from\_pretrained('BAAI/bge-reranker-large')

model.eval()

pairs = [['what is panda?', 'hi'], ['what is panda?', 'The giant panda (Ailuropoda melanoleuca), sometimes called a panda bear or simply panda, is a bear species endemic to China.']]

with torch.no\_grad():

inputs = tokenizer(pairs, padding=True, truncation=True, return\_tensors='pt', max\_length=512)

scores = model(\*\*inputs, return\_dict=True).logits.view(-1, ).float()

print(scores)