```
pip install sentence-transformers datasets transformers
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (3.4.1)
Collecting datasets
  Downloading datasets-3.4.1-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.50.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch ≥ 1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.14.1)
Requirement already satisfied: huggingface-hub ≥ 0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transf
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.1.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets) (3.18.0)
Requirement already satisfied: numpy ≥ 1.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.0.2)
Requirement already satisfied: pyarrow ≥ 15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9, ≥ 0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests ≥ 2.32.2 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec ≤ 2024.12.0, ≥ 2023.1.0 (from fsspec[http] ≤ 2024.12.0, ≥ 2023.1.0→datasets)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.14)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml ≥ 5.1 in /usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: regex≠2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11
Requirement already satisfied: tokenizers<0.22, ≥ 0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0
Requirement already satisfied: safetensors ≥ 0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: aiohappyeyeballs ≥ 2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp→dataset
Requirement already satisfied: aiosignal ≥ 1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (1.3
Requirement already satisfied: attrs ≥ 17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (25.3.0
Requirement already satisfied: frozenlist ≥ 1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (1
Requirement already satisfied: multidict<7.0, ≥ 4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (
Requirement already satisfied: propcache ≥ 0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (0.3
Requirement already satisfied: yarl<2.0, ≥ 1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp→datasets) (1
Requirement already satisfied: typing-extensions ≥ 3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-h
Requirement already satisfied: charset-normalizer<4, ≥ 2 in /usr/local/lib/python3.11/dist-packages (from requests ≥ 2.32
Requirement already satisfied: idna<4, ≥ 2.5 in /usr/local/lib/python3.11/dist-packages (from requests ≥ 2.32.2→datasets)
Requirement already satisfied: urllib3<3, ≥ 1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests ≥ 2.32.2→dat
Requirement already satisfied: certifi ≥ 2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests ≥ 2.32.2→dat
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch ≥ 1.11.0→sentence-transf
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch ≥ 1.11.0→sentence-transform
Collecting nvidia-cuda-nvrtc-cu12 == 12.4.127 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12 == 12.4.127 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12 == 12.4.127 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12 == 9.1.0.70 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12 == 12.4.5.8 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12 == 11.2.1.3 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12 == 10.3.5.147 (from torch ≥ 1.11.0→sentence-transformers)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
```

```
from huggingface_hub import login

login(token=token, add_to_git_credential=True)
```

```
from datasets import load_dataset
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_excel("test_data_mid.xlsx")

# Rename columns
df = df.rename(columns={"Question": "anchor", "Answer": "positive"})

# Add an "id" column
df["id"] = range(len(df))

# Split dataset into 90% train and 10% test
train_df, test_df = train_test_split(df, test_size=0.1, random_state=42)
```

```
# Save datasets to JSON files
train_df.to_json("train_dataset.json", orient="records", lines=True)
test_df.to_json("test_dataset.json", orient="records", lines=True)


import torch
from sentence_transformers import SentenceTransformer
from sentence_transformers.evaluation import (
    InformationRetrievalEvaluator,
    SequentialEvaluator,
)
from sentence_transformers.util import cos_sim
from datasets import load_dataset, concatenate_datasets



model_id = "BAAI/bge-base-en-v1.5" # Hugging Face model ID
matryoshka_dimensions = [768, 512, 256, 128, 64] # Important: large to small

# Load a model
model = SentenceTransformer(
    model_id, device="cuda" if torch.cuda.is_available() else "cpu"
)

# load test dataset
test_dataset = load_dataset("json", data_files="test_dataset.json", split="train")
train_dataset = load_dataset("json", data_files="train_dataset.json", split="train")

corpus_dataset = concatenate_datasets([train_dataset, test_dataset])

# Convert the datasets to dictionaries
corpus = dict(
    zip(corpus_dataset["id"], corpus_dataset["positive"])
)  # Our corpus (cid ⇒ document)
queries = dict(
    zip(test_dataset["id"], test_dataset["anchor"])
)  # Our queries (qid ⇒ question)

# Create a mapping of relevant document (1 in our case) for each query
relevant_docs = {}  # Query ID to relevant documents (qid ⇒ set([relevant_cids])
for q_id in queries:
    relevant_docs[q_id] = [q_id]


matryoshka_evaluators = []
# Iterate over the different dimensions
for dim in matryoshka_dimensions:
    ir_evaluator = InformationRetrievalEvaluator(
        queries=queries,
        corpus=corpus,
        relevant_docs=relevant_docs,
        name=f"dim_{dim}",
        truncate_dim=dim,  # Truncate the embeddings to a certain dimension
        score_functions={"cosine": cos_sim},

    )
    matryoshka_evaluators.append(ir_evaluator)

# Create a sequential evaluator
evaluator = SequentialEvaluator(matryoshka_evaluators)


# Evaluate the model
results = evaluator(model)

# # COMMENT IN for full results
# print(results)

# Print the main score
for dim in matryoshka_dimensions:
    key = f"dim_{dim}_cosine_ndcg@10"
    print
    print(f"{key}: {results[key]}")
```

```
dim_768_cosine_ndcg@10: 0.6536132266728631
dim_512_cosine_ndcg@10: 0.6457622727443029
dim_256_cosine_ndcg@10: 0.6456678361673469
dim_128_cosine_ndcg@10: 0.6262083333221699
```

```
dim_64_cosine_ndcg@10: 0.5660617625472383
```

Double-click (or enter) to edit

```
corpus_dataset
```

```
Dataset({
    features: ['Topic', 'Title', 'anchor', 'positive', 'bge-large-en-v1.5-correlation', 'id'],
    num_rows: 206461
})
```

```
from sentence_transformers import SentenceTransformerModelCardData, SentenceTransformer

# Hugging Face model ID: https://huggingface.co/BAAI/bge-base-en-v1.5

model_id = "BAAI/bge-base-en-v1.5"

# load model with SDPA for using Flash Attention 2
model = SentenceTransformer(
    model_id,
    model_kwargs={"attn_implementation": "sdpa"},
    model_card_data=SentenceTransformerModelCardData(
        language="en",
        license="apache-2.0",
        model_name="Regulatory Financial Matryoshka",
    ),
)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

```
from sentence_transformers.losses import MatryoshkaLoss, MultipleNegativesRankingLoss

matryoshka_dimensions = [768, 512, 256, 128, 64]  # Important: large to small
inner_train_loss = MultipleNegativesRankingLoss(model)
train_loss = MatryoshkaLoss(
    model, inner_train_loss, matryoshka_dims=matryoshka_dimensions
)
```

```
from sentence_transformers import SentenceTransformerTrainingArguments
from sentence_transformers.training_args import BatchSamplers

# load train dataset again
train_dataset = load_dataset("json", data_files="train_dataset.json", split="train")

# define training arguments
args = SentenceTransformerTrainingArguments(
    output_dir="bge-base-financial-matryoshka", # output directory and hugging face model ID
    num_train_epochs=4,                         # number of epochs
    per_device_train_batch_size=32,             # train batch size
    gradient_accumulation_steps=16,             # for a global batch size of 512
    per_device_eval_batch_size=16,              # evaluation batch size
    warmup_ratio=0.1,                           # warmup ratio
    learning_rate=2e-5,                         # learning rate, 2e-5 is a good value
    lr_scheduler_type="cosine",                 # use constant learning rate scheduler
    optim="adamw_torch_fused",                  # use fused adamw optimizer                                  # use tf32 preci
    bf16=True,                                  # use bf16 precision
    batch_sampler=BatchSamplers.NO_DUPLICATES,  # MultipleNegativesRankingLoss benefits from no duplicate samples in a batch
    eval_strategy="epoch",                      # evaluate after each epoch
    save_strategy="epoch",                      # save after each epoch
    logging_steps=10,                           # log every 10 steps
    save_total_limit=3,                         # save only the last 3 models
    load_best_model_at_end=True,                # load the best model when training ends
    metric_for_best_model="eval_dim_128_cosine_ndcg@10",  # Optimizing for the best ndcg@10 score for the 128 dimension
)
```

```
Generating train split: 1641/0 [00:00<00:00, 74336.89 examples/s]
```

```
train_dataset = train_dataset.filter(lambda x: x["positive"] is not None and x["anchor"] is not None)
```

Filter: 100% 2432/2432 [00:00<00:00, 80752.92 examples/s]

```
from sentence_transformers import SentenceTransformerTrainer

trainer = SentenceTransformerTrainer(
    model=model, # bg-base-en-v1
    args=args,  # training arguments
    train_dataset=train_dataset.select_columns(
        ["positive", "anchor"]
    ),  # training dataset
    loss=train_loss,
    evaluator=evaluator,
)
```

```
trainer.train()

# save the best model
trainer.save_model()
```

```
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not inten
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ··· ··· ··· ···
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: hshashank06 (hshashank06-work) to https://api.wandb.ai. Use `wandb login --relogin` to for
Tracking run with wandb version 0.19.8
Run data is saved locally in /content/wandb/run-20250324_181808-otbdig66
Syncing run bge-base-financial-matryoshka to Weights & Biases (docs)
View project at https://wandb.ai/hshashank06-work/sentence-transformers
View run at https://wandb.ai/hshashank06-work/sentence-transformers/runs/otbdig66
WARNING:sentence_transformers.data_collator:Column 'anchor' is at index 1, whereas a column with this name is usually exp
dataset = dataset.select_columns(['anchor', 'positive', 'negative'])
[12/12 01:59, Epoch 3/4]
```

Epoch	Training Loss	Validation Loss	Dim 768 Cosine Accuracy@1	Dim 768 Cosine Accuracy@3	Dim 768 Cosine Accuracy@5	Dim 768 Cosine Accuracy@10	Dim 768 Cosine Precision@1	Dim 768 Cosine Precision@3	Dim 768 Cosine Precision@5	Dim 768 Cosine Precisi
1	No log	No log	0.819672	0.885246	0.907104	0.918033	0.819672	0.295082	0.181421	0.0
2	No log	No log	0.819672	0.885246	0.907104	0.918033	0.819672	0.295082	0.181421	0.0
3	14.920600	No log	0.819672	0.885246	0.907104	0.918033	0.819672	0.295082	0.181421	0.0

Start coding or generate with AI.

37

```
from huggingface_hub import login

login(token=token, add_to_git_credential=True)  # ADD YOUR TOKEN HERE
```

```
import torch
from sentence_transformers import SentenceTransformer
from sentence_transformers.evaluation import (
    InformationRetrievalEvaluator,
    SequentialEvaluator,
)
from sentence_transformers.util import cos_sim
from datasets import load_dataset, concatenate_datasets
```

```
model_id = "hshashank06/final-regulatory-policy" # Hugging Face model ID
matryoshka_dimensions = [768, 512, 256, 128, 64] # Important: large to small
```

```
# Load a model
model = SentenceTransformer(
    model_id, device="cuda" if torch.cuda.is_available() else "cpu"
)

# load test dataset
test_dataset = load_dataset("json", data_files="test_dataset.json", split="train")
train_dataset = load_dataset("json", data_files="train_dataset.json", split="train")

corpus_dataset = concatenate_datasets([train_dataset, test_dataset])

# Convert the datasets to dictionaries
corpus = dict(
    zip(corpus_dataset["id"], corpus_dataset["positive"])
)  # Our corpus (cid ⇒ document)
queries = dict(
    zip(test_dataset["id"], test_dataset["anchor"])
)  # Our queries (qid ⇒ question)

# Create a mapping of relevant document (1 in our case) for each query
relevant_docs = {}  # Query ID to relevant documents (qid ⇒ set([relevant_cids])
for q_id in queries:
    relevant_docs[q_id] = [q_id]


matryoshka_evaluators = []
# Iterate over the different dimensions
for dim in matryoshka_dimensions:
    ir_evaluator = InformationRetrievalEvaluator(
        queries=queries,
        corpus=corpus,
        relevant_docs=relevant_docs,
        name=f"dim_{dim}",
        truncate_dim=dim,  # Truncate the embeddings to a certain dimension
        score_functions={"cosine": cos_sim},

    )
    matryoshka_evaluators.append(ir_evaluator)

# Create a sequential evaluator
evaluator = SequentialEvaluator(matryoshka_evaluators)


# Evaluate the model
results = evaluator(model)

# # COMMENT IN for full results
# print(results)

# Print the main score
for dim in matryoshka_dimensions:
    key = f"dim_{dim}_cosine_ndcg@10"
    print
    print(f"{key}: {results[key]}")
```

```
dim_768_cosine_ndcg@10: 0.7077523192798166
dim_512_cosine_ndcg@10: 0.7088071753087704
dim_256_cosine_ndcg@10: 0.7018734131882189
dim_128_cosine_ndcg@10: 0.6879293185871078
dim_64_cosine_ndcg@10: 0.6633536328298336
```

```
trainer.model.push_to_hub("hshashank06/final-regulatory-policy")
```

```
model.safetensors: 100%                                    438M/438M [00:23<00:00, 34.0MB/s]
'https://huggingface.co/hshashank06/final-regulatory-policy/commit/cd213010af4389c004b5782eb49396e9e0be7c26'
```