Finetune Embeddings - LlamaIndex Example

```
# !pip install llama_index pypdf openai accelerate llama-cpp-python -q
!pip install llama-index langchain pypdf -q
```

```
import json

from llama_index import SimpleDirectoryReader
from llama_index.node_parser import SentenceSplitter
from llama_index.schema import MetadataMode
```

Download Data

```
!mkdir -p 'data/10k/'
!wget 'https://raw.githubusercontent.com/run-llama/llama_index/main/docs/examples/data/10k/uber_2021.pdf' -O 'da
!wget 'https://raw.githubusercontent.com/run-llama/llama_index/main/docs/examples/data/10k/lyft_2021.pdf' -O 'da
```

```
--2024-01-09 10:47:55--  https://raw.githubusercontent.com/run-llama/llama_index/main/docs/examples/data/10k
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1880483 (1.8M) [application/octet-stream]
Saving to: ‘data/10k/uber_2021.pdf’

data/10k/uber_2021. 100%[===================>]   1.79M  --.-KB/s    in 0.06s

2024-01-09 10:47:55 (31.0 MB/s) - ‘data/10k/uber_2021.pdf’ saved [1880483/1880483]

--2024-01-09 10:47:55--  https://raw.githubusercontent.com/run-llama/llama_index/main/docs/examples/data/10k
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1440303 (1.4M) [application/octet-stream]
Saving to: ‘data/10k/lyft_2021.pdf’

data/10k/lyft_2021. 100%[===================>]   1.37M  --.-KB/s    in 0.06s

2024-01-09 10:47:55 (24.1 MB/s) - ‘data/10k/lyft_2021.pdf’ saved [1440303/1440303]
```

```
TRAIN_FILES = ["./data/10k/lyft_2021.pdf"]
VAL_FILES = ["./data/10k/uber_2021.pdf"]

TRAIN_CORPUS_FPATH = "./data/train_corpus.json"
VAL_CORPUS_FPATH = "./data/val_corpus.json"
```

```
def load_corpus(files, verbose=False):
  if verbose:
    print(f"Loading files {files}")

  reader = SimpleDirectoryReader(input_files=files)
  docs = reader.load_data()

  if verbose:
    print(f"Loaded {len(docs)} docs")

  parser = SentenceSplitter()
  nodes = parser.get_nodes_from_documents(docs, show_progress=verbose)

  if verbose:
    print(f"Parsed {len(nodes)} nodes")

  return nodes
```

```
train_nodes = load_corpus(TRAIN_FILES, verbose=True)
val_nodes = load_corpus(VAL_FILES, verbose=True)
```

```
Loading files ['./data/10k/lyft_2021.pdf']
Loaded 238 docs
Parsing nodes: 100%                                    238/238 [00:01<00:00, 249.04it/s]
Parsed 344 nodes
Loading files ['./data/10k/uber_2021.pdf']
Loaded 307 docs
Parsing nodes: 100%                                    307/307 [00:02<00:00, 197.29it/s]
Parsed 410 nodes
```

Generate synthetic queries

```
from llama_index.finetuning import (
    generate_qa_embedding_pairs,
    EmbeddingQAFinetuneDataset,
)
```

```
from llama_index.llms import OpenAILike
```

```
import os
from getpass import getpass


os.environ["TOGETHER_API_KEY"] = getpass("TOGETHER_API_KEY")
api_key = os.environ["TOGETHER_API_KEY"]
```

```
TOGETHER_API_KEY··********
```

```
llm = OpenAILike(
     model = "mistralai/Mixtral-8x7B-Instruct-v0.1",
    api_base = "https://api.together.xyz/v1",
    api_key=api_key
     )
```

```
train_dataset = generate_qa_embedding_pairs(train_nodes, llm=llm)
val_dataset = generate_qa_embedding_pairs(val_nodes, llm=llm)

train_dataset.save_json("train_dataset.json")
val_dataset.save_json("val_dataset.json")
```

```
100%|██████████| 344/344 [10:07<00:00,  1.77s/it]
100%|██████████| 410/410 [09:12<00:00,  1.35s/it]
```

Run Embedding Finetuning

```
from llama_index.finetuning import SentenceTransformersFinetuneEngine
```

```
!pip install -U sentence-transformers -q
```

```
finetune_engine = SentenceTransformersFinetuneEngine(
    train_dataset,
    model_id="BAAI/bge-small-en",
    model_output_path="test_model",
    val_dataset=val_dataset
)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:72: 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/setti
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(
```

.gitattributes: 100% 1.52k/1.52k [00:00<00:00, 51.9kB/s]

1_Pooling/config.json: 100% 190/190 [00:00<00:00, 10.0kB/s]

README.md: 100% 90.8k/90.8k [00:00<00:00, 4.32MB/s]

config.json: 100% 684/684 [00:00<00:00, 27.6kB/s]

config_sentence_transformers.json: 100% 124/124 [00:00<00:00, 6.13kB/s]

model.safetensors: 100% 133M/133M [00:01<00:00, 143MB/s]

pytorch_model.bin: 100% 134M/134M [00:01<00:00, 135MB/s]

sentence_bert_config.json: 100% 52.0/52.0 [00:00<00:00, 2.16kB/s]

special_tokens_map.json: 100% 125/125 [00:00<00:00, 6.36kB/s]

tokenizer.json: 100% 711k/711k [00:00<00:00, 12.2MB/s]

tokenizer_config.json: 100% 366/366 [00:00<00:00, 12.7kB/s]

vocab.txt: 100% 232k/232k [00:00<00:00, 5.83MB/s]

modules.json: 100% 349/349 [00:00<00:00, 14.7kB/s]

```
finetune_engine.finetune()
```

Epoch: 100% 2/2 [1:53:16<00:00, 3394.27s/it]

Iteration: 100% 79/79 [49:39<00:00, 35.59s/it]

Iteration: 100% 79/79 [48:55<00:00, 35.65s/it]

```
embed_model = finetune_engine.get_finetuned_model()
```

```
embed_model
```

```
HuggingFaceEmbedding(model_name='test_model', embed_batch_size=10, callback_manager=
<llama_index.callbacks.base.CallbackManager object at 0x7a00d5db39a0>, tokenizer_name='test_model',
max_length=512, pooling=<Pooling.CLS: 'cls'>, normalize=True, query_instruction=None,
text_instruction=None, cache_folder=None)
```

Evaluate Finetuned Model

```
from llama_index.embeddings import OpenAIEmbedding
from llama_index import ServiceContext, VectorStoreIndex
from llama_index.schema import TextNode
from tqdm.notebook import tqdm
import pandas as pd
```

```
def evaluate(dataset, embed_model, top_k=5, verbose=False):
  corpus = dataset.corpus
  queries = dataset.queries
  relevant_docs = dataset.relevant_docs

  service_context= ServiceContext.from_defaults(embed_model=embed_model)
  nodes = [TextNode(id_=id_, text=text) for id_, text in corpus.items()]
  index = VectorStoreIndex(
      nodes,
      service_context=service_context,
      show_progress=True
  )
  retriever = index.as_retriever(similarity_top_k=top_k)

  eval_results = []
  for query_id, query in tqdm(queries.items()):
    retrieved_nodes = retriever.retrieve(query)
    retrieved_ids = [node.node.node_id for node in retrieved_nodes]
    expected_id = relevant_docs[query_id][0]
    is_hit = expected_id in retrieved_ids

    eval_result = {
        "is_hit": is_hit,
        "retrieved": retrieved_ids,
        "expected": expected_id,
        "query": query_id
    }
    eval_results.append(eval_result)
  return eval_results
```

```
from sentence_transformers.evaluation import InformationRetrievalEvaluator
from sentence_transformers import SentenceTransformer
from pathlib import Path


def evaluate_st(dataset, model_id, name):
  corpus = dataset.corpus
  queries = dataset.queries
  relevant_docs = dataset.relevant_docs

  evaluator = InformationRetrievalEvaluator(
      queries, corpus, relevant_docs, name=name
  )
  model = SentenceTransformer(model_id)
  output_path = "results/"
  Path(output_path).mkdir(exist_ok=True, parents=True)
  return evaluator(model, output_path=output_path)
```

Run Evals

```
# import openai

# os.environ["OPENAI_API_KEY"] = getpass("OPENAI_API_KEY")
# openai.api_key = os.environ["OPENAI_API_KEY"]

# ada = OpenAIEmbedding()
# ada_val_results = evaluate(val_dataset, ada)

# df_ada = pd.DataFrame(ada_val_results)

# hit_rate_ada = df_ada["is_hit"].mean()
# hit_rate_ada
```

```
bge = "local:BAAI/bge-small-en"
bge_val_results = evaluate(val_dataset, bge)
df_bge = pd.DataFrame(bge_val_results)
hit_rate_bge = df_bge["is_hit"].mean()
hit_rate_bge
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:72: 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/setti
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(
```

Generating embeddings: 100% 410/410 [07:39<00:00, 1.08s/it]

100% 1006/1006 [01:29<00:00, 10.87it/s]

```
0.6163021868787276
```

```
evaluate_st(val_dataset, "BAAI/bge-small-en", name="bge")
```

```
0.46355582695619196
```

```
finetuned = "local:test_model"
val_results_finetuned = evaluate(val_dataset, finetuned)
```

Generating embeddings: 100% 410/410 [07:20<00:00, 1.06s/it]

100% 1006/1006 [01:20<00:00, 11.81it/s]

```
df_finetuned = pd.DataFrame(val_results_finetuned)
```

```
hit_rate_finetuned = df_finetuned["is_hit"].mean()
hit_rate_finetuned
```

```
0.668986083499006
```

```
evaluate_st(val_dataset, "test_model", name="finetuned")
```

```
0.5478756097147947
```

Summary of Results

```
df_bge["model"] = "bge"
df_finetuned["model"] = "fine_tuned"
```

```
df_all = pd.concat([df_bge, df_finetuned])
df_all.groupby("model").mean("is_hit")
```

model	is_hit
bge	0.616302
fine_tuned	0.668986

Information Retrieval Evaluator

```
df_st_bge = pd.read_csv(
    "results/Information-Retrieval_evaluation_bge_results.csv"
)
df_st_finetuned = pd.read_csv(
    "results/Information-Retrieval_evaluation_finetuned_results.csv"
)
```

```
df_st_bge["model"] = "bge"
df_st_finetuned["model"] = "fine_tuned"
df_st_all = pd.concat([df_st_bge, df_st_finetuned])
df_st_all = df_st_all.set_index("model")
df_st_all
```

model	epoch	steps	cos_sim-Accuracy@1	cos_sim-Accuracy@3	cos_sim-Accuracy@5	cos_sim-Accuracy@10	cos_sim-Precision@1	cos_sim-Recall@1	cos_sim-Precision@3
bge	-1	-1	0.359841	0.525845	0.581511	0.650099	0.359841	0.359841	0.175282
fine_tuned	-1	-1	0.452286	0.605368	0.668986	0.720676	0.452286	0.452286	0.201789

2 rows × 32 columns