```
import pandas as pd
import random
from IPython.display import display, HTML
import math
from tqdm.auto import tqdm

import torch
from torch.utils.data import DataLoader, RandomSampler
from torch.optim import AdamW

from datasets import load_dataset, DatasetDict
from datasets import ClassLabel
import evaluate

from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, TrainingArguments, pipeline
import accelerate

from langchain.chains import RetrievalQA
# from langchain.chat_models import ChatOpenAI
from langchain.llms import HuggingFaceHub, HuggingFacePipeline
from langchain.document_loaders import CSVLoader, PyPDFDirectoryLoader, PyPDFLoader, TextLoader
from langchain.vectorstores import DocArrayInMemorySearch
from langchain.indexes import VectorstoreIndexCreator
from langchain.embeddings import SentenceTransformerEmbeddings

from IPython.display import display, Markdown


print(torch.__version__)
print(torch.version.cuda)
print(torch.cuda.device_count())
```

```
2.0.1+cu118
11.8
0
```

```
PDF_DIR = '/content/redditnews.csv'

pdf_loader = CSVLoader(PDF_DIR, encoding='latin1')


docs = pdf_loader.load()


print(len(docs))
print(docs[1])
```

```
41377
page_content='date: 2016-07-01\nnews: IMF chief backs Athens as permanent Olympic host' metadata={'source': '/content/redditnews.csv
```

```
model_name = "sentence-transformers/all-mpnet-base-v2"
# model_kwargs = {'device': 'cpu'}
encode_kwargs = {'normalize_embeddings': False}

st_embed = SentenceTransformerEmbeddings(
    model_name = model_name,
    encode_kwargs = encode_kwargs,
    model_kwargs = {'device': 'cuda'}
)
```

```
Downloading (…)a8e1d/.gitattributes: 100%|| 1.18k/1.18k [00:00<00:00, 67.4kB/s]
Downloading
```

```
embed = st_embed.embed_query("Hello world! How are you")
```

```
Traceback (most recent call last)
in <cell line: 1>:1

/usr/local/lib/python3.10/dist-packages/langchain/embeddings/huggingface.py:91 i

   88 |   |   |   Embeddings for the text.
   89 |   |   """
   90 |   |   text = text.replace("\n", " ")
❱  91 |   |   embedding = self.client.encode(text, **self.encode_kwargs)
   92 |   |   return embedding.tolist()
   93 |
   94 |

/usr/local/lib/python3.10/dist-packages/sentence_transformers/SentenceTransforme
encode

  150 |   |   if device is None:
```

```
len(embed)
```

```
Traceback (most recent call last)
in <cell line: 1>:1

NameError: name 'embed' is not defined
```

```
%time

db = DocArrayInMemorySearch.from_documents(
    docs,
    st_embed
)
```

```
CPU times: user 0 ns, sys: 5 µs, total: 5 µs
Wall time: 10 µs
```

```
---------------------------------------------------------------------------
Traceback (most recent call last)
in <cell line: 3>:3

/usr/local/lib/python3.10/dist-packages/langchain/vectorstores/base.py:332 in fr

  329 |   |   """Return VectorStore initialized from documents and embeddings.""
  330 |   |   texts = [d.page_content for d in documents]
  331 |   |   metadatas = [d.metadata for d in documents]
❱ 332 |   |   return cls.from_texts(texts, embedding, metadatas=metadatas, **kwa
  333 |
  334 |   @classmethod
  335 |   async def afrom_documents(

/usr/local/lib/python3.10/dist-packages/langchain/vectorstores/docarray/in_memor
from_texts

  65 |   |   |   DocArrayInMemorySearch Vector Store
  66 |   |   """
  67 |   |   store = cls.from_params(embedding, **kwargs)
❱ 68 |   |   store.add_texts(texts=texts, metadatas=metadatas)
  69 |   |   return store
  70

/usr/local/lib/python3.10/dist-packages/langchain/vectorstores/docarray/base.py:

   77 |   |   |   List of ids from adding the texts into the vectorstore.
   78 |   |   """
   79 |   |   ids: List[str] = []
❱  80 |   |   embeddings = self.embedding.embed_documents(list(texts))
   81 |   |   for i, (t, e) in enumerate(zip(texts, embeddings)):
   82 |   |   |   m = metadatas[i] if metadatas else {}
   83 |   |   |   doc = self.doc_cls(text=t, embedding=e, metadata=m)

/usr/local/lib/python3.10/dist-packages/langchain/embeddings/huggingface.py:78 i
embed_documents

   75 |   |   |   List of embeddings, one for each text.
   76 |   |   """
   77 |   |   texts = list(map(lambda x: x.replace("\n", " "), texts))
❱  78 |   |   embeddings = self.client.encode(texts, **self.encode_kwargs)
   79 |   |   return embeddings.tolist()
   80 |
   81 |   def embed_query(self, text: str) -> List[float]:

/usr/local/lib/python3.10/dist-packages/sentence_transformers/SentenceTransforme
encode

  150 |   |   if device is None:
  151 |   |   |   device = self._target_device
  152 |   |
❱ 153 |   |   self.to(device)
  154 |   |
  155 |   |   all_embeddings = []
  156 |   |   length_sorted_idx = np.argsort([-self._text_length(sen) for sen in

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/module.py:1145 in to
```

```
query = "Why did the oeprating profit increase"
num_articles = 10

context_docs = db.similarity_search_with_score(query, k = num_articles)


print(context_docs[1])
```

```
(Document(page_content='neutral: positive\nAccording to Gran , the company has no plans to move all production to Russia , although
```

```
retriever = db.as_retriever(device='cuda')
```