```
!pip install langchain bitsandbytes accelerate langchain_community sentence-transformers faiss-gpu
```

```
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-json<0.7,>=0.5.7->langchain_community)
  Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub->accelerate) (3
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub->accele
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-h
Collecting jsonpatch<2.0,>=1.33 (from langchain-core<0.3.0,>=0.2.10->langchain)
  Downloading jsonpatch-1.33-py2.py3-none-any.whl (12 kB)
Collecting orjson<4.0.0,>=3.9.14 (from langsmith<0.2.0,>=0.1.17->langchain)
  Downloading orjson-3.10.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (144 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 145.0/145.0 kB 3.5 MB/s eta 0:00:00
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->l
Requirement already satisfied: pydantic-core==2.18.4 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->la
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langc
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langc
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->lan
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->bitsandbytes) (1.12.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->bitsandbytes) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->bitsandbytes) (3.1.4)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch->bitsandbytes)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch->bitsandbytes)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch->bitsandbytes)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch->bitsandbytes)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch->bitsandbytes)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch->bitsandbytes)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch->bitsandbytes)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch->bitsandbytes)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch->bitsandbytes)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch->bitsandbytes)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch->bitsandbytes)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/dist-packages (from torch->bitsandbytes) (2.3.
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch->bitsandbytes)
  Downloading nvidia_nvjitlink_cu12-12.5.40-py3-none-manylinux2014_x86_64.whl (21.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 49.5 MB/s eta 0:00:00
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sentence-tra
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sente
Collecting jsonpointer>=1.9 (from jsonpatch<2.0,>=1.33->langchain-core<0.3.0,>=0.2.10->langchain)
  Downloading jsonpointer-3.0.0-py2.py3-none-any.whl (7.6 kB)
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain_community)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch->bitsandby
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch->bitsa
Installing collected packages: faiss-gpu, orjson, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-cura
Successfully installed accelerate-0.31.0 bitsandbytes-0.43.1 dataclasses-json-0.6.7 faiss-gpu-1.7.2 jsonpatch-1.33 jsonp
```

```
!pip install pypdf
```

```
Collecting pypdf
  Downloading pypdf-4.2.0-py3-none-any.whl (290 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 290.4/290.4 kB 6.1 MB/s eta 0:00:00
Requirement already satisfied: typing_extensions>=4.0 in /usr/local/lib/python3.10/dist-packages (from pypdf) (4.12.2)
Installing collected packages: pypdf
Successfully installed pypdf-4.2.0
```

```python
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.prompts import ChatPromptTemplate
from langchain.vectorstores import FAISS
from langchain import HuggingFaceHub
from langchain_community.llms.huggingface_pipeline import HuggingFacePipeline
from transformers import AutoModelForCausalLM, AutoTokenizer
from langchain.chains import RetrievalQA

import torch
import os
import warnings

warnings.filterwarnings("ignore")
```

```
pdf_file = PyPDFLoader("/content/1706.03762v7.pdf")
pages = pdf_file.load_and_split()
```

```
pages[2]
```

```
Document(page_content='Most competitive neural sequence transduction models have an encoder-decoder structure [
5,2,35].\nHere, the encoder maps an input sequence of symbol representations (x1, ..., x n)to a sequence\nof continuous
representations z= (z1, ..., z n). Given z, the decoder then generates an output\nsequence (y1, ..., y m)of symbols one
element at a time. At each step the model is auto-regressive\n[10], consuming the previously generated symbols as
additional input when generating the next.\n2', metadata={'source': '/content/1706.03762v7.pdf', 'page': 1})
```

```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1024,
    chunk_overlap=32
)
```

```
chunks = text_splitter.split_documents(pages)
```

```
chunks[23]
```

```
Document(page_content='recurrent layers, by a factor of k. Separable convolutions [ 6], however, decrease the
complexity\nconsiderably, to O(k·n·d+n·d2). Even with k=n, however, the complexity of a separable\nconvolution is equal
to the combination of a self-attention layer and a point-wise feed-forward layer,\nthe approach we take in our
model.\nAs side benefit, self-attention could yield more interpretable models. We inspect attention distributions\nfrom
our models and present and discuss examples in the appendix. Not only do individual attention\nheads clearly learn to
perform different tasks, many appear to exhibit behavior related to the syntactic\nand semantic structure of the
sentences.\n5 Training\nThis section describes the training regime for our models.\n5.1 Training Data and Batching\nWe
trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million\nsentence pairs. Sentences were
encoded using byte-pair encoding [ 3], which has a shared source-', metadata={'source': '/content/1706.03762v7.pdf',
'page': 6})
```

```
Embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2",
                                   model_kwargs={"device": "cuda"})

vector_db = FAISS.from_texts([str(chunk) for chunk in chunks], Embeddings)
```

```
modules.json: 100%                                         349/349 [00:00<00:00, 10.6kB/s]
config_sentence_transformers.json: 100%                    116/116 [00:00<00:00, 3.51kB/s]
README.md: 100%                                            10.7k/10.7k [00:00<00:00, 182kB/s]
sentence_bert_config.json: 100%                            53.0/53.0 [00:00<00:00, 1.40kB/s]
config.json: 100%                                          612/612 [00:00<00:00, 16.4kB/s]
model.safetensors: 100%                                    90.9M/90.9M [00:00<00:00, 256MB/s]
tokenizer_config.json: 100%                                350/350 [00:00<00:00, 24.4kB/s]
vocab.txt: 100%                                            232k/232k [00:00<00:00, 3.43MB/s]
tokenizer.json: 100%                                       466k/466k [00:00<00:00, 6.71MB/s]
special_tokens_map.json: 100%                              112/112 [00:00<00:00, 7.82kB/s]
1_Pooling/config.json: 100%                                190/190 [00:00<00:00, 11.4kB/s]
```

```
question = """
what is the purpose of the decoder?
"""

relevant_results = vector_db.similarity_search(question, k=2)
relevant_results[0]
```

```
Document(page_content="page_content='and the memory keys and values come from the output of the encoder. This allows
every\\nposition in the decoder to attend over all positions in the input sequence. This mimics the\\ntypical encoder-
decoder attention mechanisms in sequence-to-sequence models such as\\n[38, 2, 9].\\n•The encoder contains self-
attention layers. In a self-attention layer all of the keys, values\\nand queries come from the same place, in this
case, the output of the previous layer in the\\nencoder. Each position in the encoder can attend to all positions in
the previous layer of the\\nencoder.\\n•Similarly, self-attention layers in the decoder allow each position in the
decoder to attend to\\nall positions in the decoder up to and including that position. We need to prevent
leftward\\ninformation flow in the decoder to preserve the auto-regressive property. We implement this\\ninside of
scaled dot-product attention by masking out (setting to −∞) all values in the input\\nof the softmax which correspond
to illegal connections. See Figure 2.' metadata={'source': '/content/1706.03762v7.pdf', 'page': 4}")
```

```
prompt = """
Using this piece of information:
\n
{context}
\n
Answer the following question:
```

```
Answer the following question:
\n
{question}
\n
Answer:
\n
"""

prompt = ChatPromptTemplate.from_template(prompt)


from transformers import BitsAndBytesConfig


bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4"
)


from transformers import AutoTokenizer, AutoModelForCausalLM

# Authenticate using your Hugging Face token
huggingface_token = "sample_api_key"

# Load the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-Instruct-v0.2", use_auth_token=huggingface_token)
model = AutoModelForCausalLM.from_pretrained("mistralai/Mistral-7B-Instruct-v0.2", use_auth_token=huggingface_token, quantiza
tokenizer_config.json: 100%                                   1.47k/1.47k [00:00<00:00, 31.5kB/s]

tokenizer.model: 100%                                   493k/493k [00:00<00:00, 4.79MB/s]

tokenizer.json: 100%                                   1.80M/1.80M [00:00<00:00, 7.67MB/s]

special_tokens_map.json: 100%                                   72.0/72.0 [00:00<00:00, 2.59kB/s]

loading file tokenizer.model from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct-v0.2/snap
loading file tokenizer.json from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct-v0.2/snaps
loading file added_tokens.json from cache at None
loading file special_tokens_map.json from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct-v
loading file tokenizer_config.json from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct-v0.
config.json: 100%                                   596/596 [00:00<00:00, 11.6kB/s]
loading configuration file config.json from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct
Model config MistralConfig {
  "_name_or_path": "mistralai/Mistral-7B-Instruct-v0.2",
  "architectures": [
    "MistralForCausalLM"
  ],
  "attention_dropout": 0.0,
  "bos_token_id": 1,
  "eos_token_id": 2,
  "hidden_act": "silu",
  "hidden_size": 4096,
  "initializer_range": 0.02,
  "intermediate_size": 14336,
  "max_position_embeddings": 32768,
  "model_type": "mistral",
  "num_attention_heads": 32,
  "num_hidden_layers": 32,
  "num_key_value_heads": 8,
  "rms_norm_eps": 1e-05,
  "rope_theta": 1000000.0,
  "sliding_window": null,
  "tie_word_embeddings": false,
  "torch_dtype": "bfloat16",
  "transformers_version": "4.41.2",
  "use_cache": true,
  "vocab_size": 32000
}

Overriding torch_dtype=None with `torch_dtype=torch.float16` due to requirements of `bitsandbytes` to enable model loadi
The device_map was not initialized. Setting device_map to {'':torch.cuda.current_device()}. If you want to use the model
`low_cpu_mem_usage` was None, now set to True since model is quantized.
model.safetensors.index.json: 100%                                   25.1k/25.1k [00:00<00:00, 1.04MB/s]
loading weights file model.safetensors from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-7B-Instruct
Downloading shards: 100%                                   3/3 [02:24<00:00, 47.62s/it]

model-00001-of-00003.safetensors: 100%                                   4.94G/4.94G [00:52<00:00, 15.8MB/s]

model-00002-of-00003.safetensors: 100%                                   5.00G/5.00G [00:44<00:00, 172MB/s]

model-00003-of-00003.safetensors: 100%                                   4.54G/4.54G [00:46<00:00, 158MB/s]
Instantiating MistralForCausalLM model under default dtype torch.float16.
Generate config GenerationConfig {
  "bos_token_id": 1,
  "eos_token_id": 2
}

Loading checkpoint shards: 100%                                   3/3 [01:05<00:00, 21.76s/it]
All model checkpoint weights were used when initializing MistralForCausalLM.

All the weights of MistralForCausalLM were initialized from the model checkpoint at mistralai/Mistral-7B-Instruct-v0.2.
If your task is similar to the task the model of the checkpoint was trained on, you can already use MistralForCausalLM f
generation_config.json: 100%                                   111/111 [00:00<00:00, 5.80kB/s]
loading configuration file generation_config.json from cache at /root/.cache/huggingface/hub/models--mistralai--Mistral-
Generate config GenerationConfig {
  "bos_token_id": 1,
  "eos_token_id": 2
}
```

```
from transformers import pipeline

pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=128)
lc_pipeline = HuggingFacePipeline(pipeline=pipe)
```

```
qa_chain = RetrievalQA.from_chain_type(
    llm=lc_pipeline,
    retriever=vector_db.as_retriever(search_kwargs = {"k": 3}),
```

```
result = qa_chain({"query": question})
```

```
Disabling tokenizer parallelism, we're using DataLoader multithreading already
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
```

```
print(result["result"].split("Answer:")[1])
```

```
e next token in the sequence based on the attention mechanism. The decoder also contains self-attention layers, but it is
```

```
print(result["source_documents"])
```

```
l values in the input\\nof the softmax which correspond to illegal connections. See Figure 2.' metadata={'source': '/cont
```

```
question = """
What is the attention function?
"""
```

```
result = qa_chain({"query": question})
```

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
```

```
print(result["result"].split("Answer:")[1])
```

```
omputes a weighted sum of the input sequence based on the query and the input sequence itself. The weights are determined
```

```
question = """
What are the three ways of using multi-head attention?
"""
```

```
result = qa_chain({"query": question})
```

```
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
```

```
print(result["result"].split("Answer:")[1])
```

```
The Transformer uses multi-head attention in three different ways:

1. In "encoder-decoder attention" layers, the queries come from the previous decoder layer.
2. In "self-attention" layers, the queries, keys, and values come from the same input sequence.
3. In "multi-head attention with different input sequences", the queries, keys, and values come from different input seq



Reference:


page_content='4 Why Self-Attention\nIn this section we compare various aspects of self-attention layers to the recurrent
```

```
print(result["source_documents"])
```

```
minimum number of sequential operations required.\\nThe third is the path length between long-range dependencies in the
```

```
question = """
Why did the authors use self-attention
"""
```