

1 Start coding or generate with AI.

1 Start coding or generate with AI.

1 !pip install PyPDF2 langchain faiss-cpu

1 !pip install -U langchain-community

```
→ Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Collecting langchain-text-splitters<1.0.0,>=0.3.8 (from langchain<1.0.0,>=0.3.23->langchain-community)
  Downloading langchain_text_splitters-0.3.8-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from langchain<1.0.0,>=0.3.23->langchain-community)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages
Collecting python-dotenv>=0.21.0 (from pydantic-settings<3.0.0,>=2.4.0->langchain-community)
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from pydantic-settings<3.0.0,>=2.4.0->langchain-community)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from pydantic-settings<3.0.0,>=2.4.0->langchain-community)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from pydantic-settings<3.0.0,>=2.4.0->langchain-community)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages (from SQLAlchemy<2.0.0,>=1.4.22->pydantic-settings<3.0.0,>=2.4.0->langchain-community)
```

```
Successfully uninstalled langchain-core-0.3.50
Attempting uninstall: langchain-text-splitters
  Found existing installation: langchain-text-splitters 0.3.7
  Uninstalling langchain-text-splitters-0.3.7:
    Successfully uninstalled langchain-text-splitters-0.3.7
Attempting uninstall: langchain
  Found existing installation: langchain 0.3.22
  Uninstalling langchain-0.3.22:
    Successfully uninstalled langchain-0.3.22
Successfully installed dataclasses-isone-0.6.7 httpx-sse-0.4.0 langchain-0.3.23 langchain-commun
```

```
1 import os
2 import torch
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7 from transformers import AutoModelForCausalLM, AutoTokenizer
8 from google.colab import drive
9
10 # تثبيت الاعتمادات الالزامية
11 !pip install PyPDF2 langchain faiss-cpu transformers torch
12
13 # تهيئة النماذج
14 def initialize_models():
15     # تحميل نموذج التضمين من Hugging Face
16     print("جاری تحميل نموذج التضمين...")
17     embedding_model = HuggingFaceEmbeddings(model_name="sentence-t"
18
19     # تحميل نموذج اللغة من Hugging Face
20     print("جاری تحميل نموذج اللغة...")
21     model_id = "bartowski/Phi-3.5-mini-instruct-ex12..."
22     tokenizer = AutoTokenizer.from_pretrained(model_id)
23     model = AutoModelForCausalLM.from_pretrained(
24         model_id,
25         torch_dtype=torch.float16,
26         device_map="auto"
27     )
28
29     return embedding_model, model, tokenizer
30
31 # استخراج النص من ملف PDF
32 def extract_text_from_pdf(pdf_path):
33     print(f": جاري استخراج النص من الملف {pdf_path}")
34     text = ""
35     try:
```

```

36     pdf_reader = PdfReader(pdf_path)
37     for page in pdf_reader.pages:
38         text += page.extract_text() + "\n"
39     print(f". حرف من النص {len(text)} تم استخراج")
40     return text
41 except Exception as e:
42     print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
43     return None
44
45 # تقسيم النص إلى أجزاء صغيرة
46 def split_text(text):
47     print("جارٍ تقسيم النص إلى أجزاء...")
48     text_splitter = RecursiveCharacterTextSplitter(
49         chunk_size=1000,
50         chunk_overlap=200,
51         length_function=len
52     )
53     chunks = text_splitter.split_text(text)
54     print(f". تم تقسيم النص إلى {len(chunks)} جزء")
55     return chunks
56
57 # إنشاء قاعدة المعرفة (قاعدة البيانات المضمنة)
58 def create_knowledge_base(chunks, embedding_model):
59     print("جارٍ إنشاء قاعدة المعرفة...")
60     knowledge_base = FAISS.from_texts(chunks, embedding_model)
61     print("تم إنشاء قاعدة المعرفة بنجاح")
62     return knowledge_base
63
64 # البحث عن المعلومات ذات الصلة
65 def search_knowledge_base(knowledge_base, query, k=4):
66     print(f": جاري البحث عن {query}")
67     docs = knowledge_base.similarity_search(query, k=k)
68     context = "\n\n".join(doc.page_content for doc in docs)
69     return context
70
71 # توليد الإجابة باستخدام نموذج اللغة
72 def generate_answer(model, tokenizer, context, query):
73     print("جارٍ توليد الإجابة")
74
75     prompt = f"""
76 {context}
77
78 : السؤال
    """

```

```
79 {query}
80
81 : الإجابة
82 """
83
84     inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
85
86     outputs = model.generate(
87         **inputs,
88         max_new_tokens=64, # 512 سياق بحجم
89         num_return_sequences=1,
90         pad_token_id=tokenizer.eos_token_id,
91         do_sample=True,
92         temperature=0.7,
93         top_p=0.9,
94         max_length=inputs.input_ids.shape[1] + 22 # المخرجات 22
95     )
96
97     answer = tokenizer.decode(outputs[0][inputs.input_ids.shape[1]:])
98     return answer
99
100 # الدالة الرئيسية
101 def main():
102     print("RAG مع كتاب PDF")
103     print("=" * 50)
104
105     # الاتصال بجوجل درايف (اختياري)
106     try:
107         drive.mount('/content/drive')
108         print("تم الاتصال بجوجل درايف بنجاح")
109     except:
110         print("تصال بجوجل درايف، سيتم استخدام المسارات المحلية")
111
112     # تهيئة النماذج
113     embedding_model, llm_model, tokenizer = initialize_models()
114
115     # من المستخدم PDF طلب مسار الكتاب
116     pdf_path = input("PDF على سبيل المثال) /content/drive/1Bhc6IBjkHzwRFMzQCjDRnlg7BIDQcmI#scrollTo=01_nkrGzASbb&printMode=true")
117
118     # استخراج النص وإنشاء قاعدة المعرفة
119     text = extract_text_from_pdf(pdf_path)
120     if text:
121         chunks = chunk_text(text, 1024)
```

```
chunks = split_chunks(chunks)
knowledge_base = create_knowledge_base(chunks, embedding_model)

حلقة استعلام المستخدم #  
while True:  
    query = input("\nالسؤال (أو اكتب 'خروج' للخروج): ")  
    if query.lower() == 'خروج' or query.lower() == 'exit':  
        break  
  
    # استرجاع السياق ذي الصلة  
    context = search_knowledge_base(knowledge_base, query)  
  
    # توليد الإجابة  
    answer = generate_answer(llm_model, tokenizer, context)  
  
    print("\nالإجابة:")  
    print("-" * 50)  
    print(answer)  
    print("-" * 50)  
  
print("تم إنتهاء البرنامج")
```

```
if __name__ == "__main__":  
    main()
```

```
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.11/dist-packages (3.0.1)
Requirement already satisfied: langchain in /usr/local/lib/python3.11/dist-packages (0.3.23)
Requirement already satisfied: faiss-cpu in /usr/local/lib/python3.11/dist-packages (1.10.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.50.3)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (2.6.0+cu124)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.51 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.8 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: langsmith<0.4,>=0.1.17 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.11/dist-packages (1.4.35)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/dist-packages (from langchain)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dist-packages (from langchain)
Requirement already satisfied: numpy<3.0,>=1.25.0 in /usr/local/lib/python3.11/dist-packages (1.25.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from faiss-cpu)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers)
Requirement already satisfied: huggingface-hub<1.0,>=0.26.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch) (2.2.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch)
  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)
  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)
  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)
  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)
  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)
  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)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparseelt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from torch)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (1
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages (from SQL
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from httpx<1,>
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from h
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.11/dist-packages (fr
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from ar
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.9 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
    ━━━━━━━━━━━━━━━━ 13.8/13.8 MB 76.9 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
    ━━━━━━━━━━━━━━ 24.6/24.6 MB 55.6 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
    ━━━━━━━━ 883.7/883.7 kB 36.9 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
    ━━━━━━━━ 664.8/664.8 MB 1.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
    ━━━━━━ 211.5/211.5 MB 2.1 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
    ━━━━ 56.3/56.3 MB 16.0 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
    ━━━━ 127.9/127.9 MB 7.9 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
    ━━━━ 207.5/207.5 MB 6.5 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
    ━━━━ 21.1/21.1 MB 20.6 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, n
Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
        Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
        Successfully uninstalled nvidia-curand-cu12-10.3.6.82
Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
        Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
        Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
        Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
        Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
```

```

Attempting uninstall: nvidia-cudnn-cu12
  Found existing installation: nvidia-cudnn-cu12 9.3.0.75
  Uninstalling nvidia-cudnn-cu12-9.3.0.75:
    Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
Attempting uninstall: nvidia-cusolver-cu12
  Found existing installation: nvidia-cusolver-cu12 11.6.3.83
  Uninstalling nvidia-cusolver-cu12-11.6.3.83:
    Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-
مع كتاب RAG بـPDF
=====
Mounted at /content/drive
تم الاتصال بجوجل درايف بنجاح.
جارٍ تحميل نموذج التضمين...
<ipython-input-5-c7c2fd247fe4>:17: LangChainDeprecationWarning: The class `HuggingFaceEmbedding`  

embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")  

/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(  

modules.json: 100%                                         349/349 [00:00<00:00, 36.8kB/s]  

config_sentence_transformers.json: 100%                      116/116 [00:00<00:00, 12.2kB/s]  

README.md: 100%                                         10.5k/10.5k [00:00<00:00, 1.07MB/s]  

sentence_bert_config.json: 100%                           53.0/53.0 [00:00<00:00, 5.58kB/s]  

config.json: 100%                                         612/612 [00:00<00:00, 48.9kB/s]  

Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to Xet Storage.  

WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.  

model.safetensors: 100%                                     90.9M/90.9M [00:00<00:00, 169MB/s]  

tokenizer_config.json: 100%                               350/350 [00:00<00:00, 40.4kB/s]  

vocab.txt: 100%                                         232k/232k [00:00<00:00, 12.0MB/s]  

tokenizer.json: 100%                                       466k/466k [00:00<00:00, 7.18MB/s]  

special_tokens_map.json: 100%                            112/112 [00:00<00:00, 7.22kB/s]  

config.json: 100%                                         190/190 [00:00<00:00, 20.7kB/s]  

جارٍ تحميل نموذج اللغة Phi-3.5-mini-instruct-ex12...
-----
OSError                                                 Traceback (most recent call last)
<ipython-input-5-c7c2fd247fe4> in <cell line: 0>()
  142
  143 if __name__ == "__main__":
--> 144     main()

----- 3 frames -----
/usr/local/lib/python3.11/dist-packages/transformers/tokenization_utils_base.py in from_pretrained(cls, pretrained_model_name_or_path, cache_dir, force_download, local_files_only, token, revision, trust_remote_code, *init_inputs, **kwargs)
  2044         # loaded directly from the GGUF file.
  2045         if all(full_file_name is None for full_file_name in
resolved_vocab_files.values()) and not gguf_file:
-> 2046             raise EnvironmentError(
  2047                 f"Can't load tokenizer for '{pretrained_model_name_or_path}'. If you


```

```
were trying to load it from "
2048           "'https://huggingface.co/models', make sure you don't have a local
directory with the same name. "
```

OSError: Can't load tokenizer for 'bartowski/Phi-3.5-mini-instruct-ex12'. If you were trying to load it from '<https://huggingface.co/models>', make sure you don't have a local directory with the same name. Otherwise, make sure 'bartowski/Phi-3.5-mini-instruct-ex12' is the correct path to a directory containing all relevant files for a BartTokenizerFast tokenizer.

Next steps:

[Explain error](#)

```
1 import os
2 import subprocess
3 import torch
4 from PyPDF2 import PdfReader
5 from langchain.text_splitter import RecursiveCharacterTextSplitter
6 from langchain.embeddings import HuggingFaceEmbeddings
7 from langchain.vectorstores import FAISS
8
9 إذا كنت تعمل في بيئة مثل (Colab) تثبيت الاعتمادات #
10 # !pip install -q PyPDF2 langchain faiss-cpu
11
12 دالة لاستخراج النص من ملف #
13 def extract_text_from_pdf(pdf_path):
14     print(f"جارٍ استخراج النص من الملف {pdf_path}")
15     text = ""
16     try:
17         pdf_reader = PdfReader(pdf_path)
18         for page in pdf_reader.pages:
19             page_text = page.extract_text()
20             if page_text:
21                 text += page_text + "\n"
22         print(f"تم استخراج {len(text)} حرف من النص")
23     return text
24 except Exception as e:
25     print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
26     return None
27
28 دالة لتقسيم النص إلى أجزاء صغيرة #
29 def split_text(text):
30     print("جارٍ تقسيم النص إلى أجزاء")
31     text_splitter = RecursiveCharacterTextSplitter(
32         chunk_size=1000,
33         chunk_overlap=200,
34         length_function=len
35     )
36     chunks = text_splitter.split_text(text)
37     print(f"تم تقسيم النص إلى {len(chunks)} جزء")
38     return chunks
39
40 إنشاء قاعدة المعرفة باستخدام #
41 def create_knowledge_base(chunks, embedding_model):
42     print("جارٍ إنشاء قاعدة المعرفة...")
```

```

43     knowledge_base = FAISS.from_texts(chunks, embedding_model)
44     print("تم إنشاء قاعدة المعرفة بنجاح")
45     return knowledge_base
46
47 دالة لاسترجاع السياق من قاعدة المعرفة بناءً على استعلام المستخدم #
48 def search_knowledge_base(knowledge_base, query, k=4):
49     print(f"جارٍ البحث عن: {query}")
50     docs = knowledge_base.similarity_search(query, k=k)
51     context = "\n\n".join(doc.page_content for doc in docs)
52     return context
53
54 # دالة لاستدعاء نموذج Exllama عبر subprocess
55 def generate_answer_exllama(model_path, prompt):
56     """
57         يقبل المعلمات exllama_inference.py: نفترض وجود سكريبت
58         --model : مسار النموذج
59         --prompt : النص الكامل (السياق والسؤال)
60         حسب الحاجة max_new_tokens وـtemperature تعديل الخيارات مثل .
61     """
62     command = [
63         "python", "exllama_inference.py",
64         "--model", model_path,
65         "--prompt", prompt,
66         # يمكنك إضافة معلمات أخرى مثل:
67         # "--max_new_tokens", "64",
68         # "--temperature", "0.7"
69     ]
70
71     try:
72         # تشغيل الأمر واستلام النتيجة
73         result = subprocess.run(command, capture_output=True, text=True)
74         output = result.stdout.strip()
75         return output
76     except subprocess.CalledProcessError as e:
77         print(f"حدث خطأ أثناء استدعاء Exllama:", e)
78         return None
79
80 def main():
81     print("RAG بـExllama يستخدم نموذج PDF مع كتاب برنامج")
82     print("==" * 50)
83
84     # خطوات إعداد قاعدة المعرفة

```

```
85 pdf_path = input("أدخل مسار ملف PDF: /content/drive/MyDrive/")
86 text = extract_text_from_pdf(pdf_path)
87 if not text:
88     return
89
90 # يمكنك استخدام أي نموذج مناسب مثل) استخدم نموذج التضمين المناسب
91 print("...جارى تحميل نموذج التضمين")
92 embedding_model = HuggingFaceEmbeddings(model_name="sentence-t5")
93 chunks = split_text(text)
94 knowledge_base = create_knowledge_base(chunks, embedding_model)
95
96 # (عدل المسار حسب مكان النموذج المحوّل) مسار النموذج الخاص بـ Exllama
97 model_path = input("أدخل مسار نموذج Exllama: /path/to/exllama")
98
99 # حلقة استعلام المستخدم
100 while True:
101     query = input("\nأدخل سؤالك (أو اكتب 'خروج' للخروج).strip()")
102     if query.lower() in ['خروج', 'exit']:
103         break
104
105     # استرجاع السياق ذي الصلة
106     context = search_knowledge_base(knowledge_base, query)
107
108     # يحتوي على السياق والسؤال
109     prompt = f"""
110 {context}
111
112 السؤال:
113 {query}
114
115 """": الإجابة
116
117     # لتوليد الإجابة Exllama استدعاء نموذج
118     answer = generate_answer_exllama(model_path, prompt)
119
120     print("\nالإجابة:")
121     print("-" * 50)
122     print(answer if answer else "لم يتم توليد إجابة")
123     print("-" * 50)
124
125     print("تم إنتهاء البرنامج")
126
```

```
127 if __name__ == "__main__":
128     main()
129
```

→ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```
=====
PDF مسار أدخل (مثلاً): /content/drive/MyDrive/book.pdf: /content/The_Little_Prince_Antoine_de_Sair
جارى استخراج النص من الملف
تم استخراج 76733 حرف من النص.
جارى تحميل نموذج التضمين
جارى تقسيم النص إلى أجزاء
تم تقسيم النص إلى 98 جزء
جارى إنشاء قاعدة المعرفة
تم إنشاء قاعدة المعرفة بنجاح
جارى إنشاء قاعدة المعرفة بنجاح
Exllama (مثلاً: /path/to/exllama_model): /content/Phi-3.5-mini-instruct-ex12
```

أدخل سؤالك (أو اكتب 'خروج' للخروج)

جارى البحث عن What is the topic of the book?

Exllama: Command '['python', 'exllama_inference.py', '--model', '/content/Phi-3.

الإجابة:

لم يتم توليد إجابة.

أدخل سؤالك (أو اكتب 'خروج' للخروج) : خروج

تم إنهاء البرنامج.

```
1 !git clone --single-branch --branch 3_5 https://huggingface.co/bart1
2
```

→ Cloning into 'Phi-3.5-mini-instruct-ex12'...

```
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 3 (from 1)
Unpacking objects: 100% (26/26), 566.80 KiB | 3.31 MiB/s, done.
```

```
1 !rm -rf /content/Phi-3.5-mini-instruct-ex12
```

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f"جارى استخراج النص من الملف {pdf_path}")
10    text = ""
```

```

11     try:
12         pdf_reader = PdfReader(pdf_path)
13         for page in pdf_reader.pages:
14             page_text = page.extract_text()
15             if page_text:
16                 # تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
17                 page_text = page_text.replace("\r", " ").replace(
18                     "\n", "\n")
19                 print(f"تم استخراج {len(page_text)} حرفاً من النص")
20             return page_text
21     except Exception as e:
22         print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
23     return None
24
25 def split_text(text):
26     print("جارٍ تقسيم النص إلى أجزاء...")
27     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chun
28     chunks = splitter.split_text(text)
29     print(f"تم تقسيم النص إلى {len(chunks)} جزء.")
30     return chunks
31
32 def create_knowledge_base(chunks, embedding_model):
33     print("جارٍ إنشاء قاعدة المعرفة...")
34     kb = FAISS.from_texts(chunks, embedding_model)
35     print("تم إنشاء قاعدة المعرفة بنجاح!")
36     return kb
37
38 def search_knowledge_base(kb, query, k=4):
39     print(f"جارٍ البحث عن '{query}'")
40     docs = kb.similarity_search(query, k=k)
41     context = "\n\n".join(doc.page_content for doc in docs)
42     return context
43
44 def generate_answer_exllama(model_path, prompt):
45     """
46     مع تمرير البرومبت والنموذج exllama_inference.py يقوم هذا الدالة باستدعاء سكريبت
47     (max_new_tokens, temperature, top_p).
48     """
49     command = [
50         "python", "exllama_inference.py",
51         "--model", model_path,
52         "--prompt", prompt,

```

```
53     "--max_new_tokens", "64",
54     "--temperature", "0.7",
55     "--top_p", "0.9"
56 ]
57 print("Executing command:", " ".join(command))
58 try:
59     result = subprocess.run(command, capture_output=True, text=True)
60     output = result.stdout.strip()
61     return output
62 except subprocess.CalledProcessError as e:
63     print("حدث خطأ أثناء استدعاء Exllama:", e)
64     print("stderr:", e.stderr)
65     return None
66
67 def main():
68     print(" باستخدام نموذج PDF مع كتاب RAG برنامج Exllama")
69     print("=" * 50)
70
71     # طلب مسار ملف PDF
72     pdf_path = input("أدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/")
73     text = extract_text_from_pdf(pdf_path)
74     if not text:
75         return
76
77     # تقطيع النص وإنشاء قاعدة المعرفة
78     chunks = split_text(text)
79
80     print("...جارٍ تحميل نموذج التضمين")
81     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
82     kb = create_knowledge_base(chunks, embedding_model)
83
84     # طلب مسار نموذج Exllama
85     model_path = input("أدخل مسار نموذج Exllama (مثلاً: /content/Phi-3.5/)")
86
87     # حلقة استعلام المستخدم
88     while True:
89         query = input("\nالسؤال (أو اكتب 'خروج' للخروج)\n").strip()
90         if query.lower() in ['خروج', 'exit']:
91             break
92
93         context = search_knowledge_base(kb, query)
94         prompt = f"\nالسؤال: {query}\nالإجابة: {context}\n\n" + "-"*50
```

```

95
96     answer = generate_answer_exllama(model_path, prompt)
97
98     print("\nالإجابة:")
99     print("-" * 50)
100    if answer:
101        print(answer)
102    else:
103        print("لم يتم توليد إجابة")
104    print("-" * 50)
105
106    print("تم إنتهاء البرنامج")
107
108 if __name__ == "__main__":
109     main()
110

```

➡️ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```

=====
PDF: /content/drive/MyDrive/book.pdf): /content/g.pdf
Jarvi استخراج النص من الملف
تم استخراج 21 حرف من النص.
Jarvi تقسيم النص إلى أجزاء
...
تم تقسيم النص إلى 1 جزء.
Jarvi تحميل نموذج التضمين
...
Jarvi إنشاء قاعدة المعرفة
...
تم إنشاء قاعدة المعرفة بنجاح.
Exllama (مثال): /content/Phi-3.5-mini-instruct-ex12): What is my name?

```

أدخل سؤالك (أو اكتب 'خروج' للخروج): خروج
تم إنتهاء البرنامج.

```

1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f"Jarvi استخراج النص من الملف {pdf_path}")
10    text = ""
11    try:
12        pdf_reader = PdfReader(pdf_path)
13        for page in pdf_reader.pages:
14            page_text = page.extract_text()
15            if page_text:

```

```

تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة.
16 ..... # حرف من النص تم استخراج {len(text)} تم
17 ..... page_text = page_text.replace("\r", "").replace("",
18 ..... text += page_text + "\n"
19 ..... print(f" حرف من النص تم استخراج {len(text)} تم")
20 ..... return text
21 ..... except Exception as e:
22 .....     print(f" حدث خطأ أثناء قراءة ملف PDF: {e}")
23 ..... return None
24
25 def split_text(text):
26 ..... print("جارٍ تقسيم النص إلى أجزاء...")
27 ..... splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chui
28 ..... chunks = splitter.split_text(text)
29 ..... print(f" تم تقسيم النص إلى {len(chunks)} جزء")
30 ..... return chunks
31
32 def create_knowledge_base(chunks, embedding_model):
33 ..... print("جارٍ إنشاء قاعدة المعرفة...")
34 ..... kb = FAISS.from_texts(chunks, embedding_model)
35 ..... print("تم إنشاء قاعدة المعرفة بنجاح")
36 ..... return kb
37
38 def search_knowledge_base(kb, query, k=4):
39 ..... print(f"جارٍ البحث عن {query}")
40 ..... docs = kb.similarity_search(query, k=k)
41 ..... context = "\n\n".join(doc.page_content for doc in docs)
42 ..... return context
43
44 def generate_answer_exllama(model_path, prompt):
45 ..... """
46 ..... يقوم هذا الدالة باستدعاء سكريبت exllama_inference.py
47 ..... يتم إضافة معلمات إضافية للتوليد (max_new_tokens, temperature)
48 ..... """
49 ..... command = [
50 .....     "python", "exllama_inference.py",
51 .....     "--model", model_path,
52 .....     "--prompt", prompt,
53 .....     "--max_new_tokens", "64",
54 .....     "--temperature", "0.7",
55 .....     "--top_p", "0.9"
56 ..... ]
57 ..... print("Executing command:", "\n".join(command))
58 ..... try:

```

```
59 ..... result = subprocess.run(command, capture_output=True, text=True)
60 ..... output = result.stdout.strip()
61 ..... return output
62 ..... except subprocess.CalledProcessError as e:
63 .....     print("حدث خطأ أثناء استدعاء Exllama:", e)
64 .....     print("stderr:", e.stderr)
65 ..... return None
66
67 def main():
68 ..... print(" باستخدام نموذج PDF مع كتاب RAG ببرنامج Exllama")
69 ..... print("*" * 50)
70 .....
71 ..... # طلب مسار ملف PDF
72 ..... pdf_path = input("أدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/...)")
73 ..... text = extract_text_from_pdf(pdf_path)
74 ..... if not text:
75 .....     return
76 .....
77 ..... # تقسيم النص وإنشاء قاعدة المعرفة
78 ..... chunks = split_text(text)
79 .....
80 ..... print("جارٍ تحميل نموذج التضمين...")
81 ..... embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
82 ..... kb = create_knowledge_base(chunks, embedding_model)
83 .....
84 ..... # طلب مسار نموذج Exllama
85 ..... model_path = input("أدخل مسار نموذج Exllama (مثلاً: /content/FastChat/...).")
86 .....
87 ..... # حلقة استعلام المستخدم
88 ..... while True:
89 .....     query = input("\nالسؤال (أو اكتب 'خروج' للخروج):\n")
90 .....     if query.lower() in ['خروج', 'exit']:
91 .....         break
92 .....
93 .....     context = search_knowledge_base(kb, query)
94 .....     prompt = f":السؤال\n\nالسياق\n\n{n}{query}\n\nإجابة"
95 .....
96 .....     answer = generate_answer_exllama(model_path, prompt)
97 .....
98 .....     print("\nإجابة:")
99 .....     print("-" * 50)
100 .....    if answer:
101 .....        print(answer)
```

```

102 .....else:
103 .....print("لم يتم توليد إجابة")
104 .....print("-" * 50)
105 .....
106 ....print("تم إنتهاء البرنامج")
107
108 if __name__ == "__main__":
109 ....main()
110

```

→ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```

=====
PDF (مثلاً) /content/drive/MyDrive/book.pdf: /content/g.pdf
جارى استخراج النص من الملف
تم استخراج 21 حرف من النص.
جارى تقسيم النص إلى أجزاء
...
تم تقسيم النص إلى 1 جزء.
جارى تحميل نموذج التضمين
...
جارى إنشاء قاعدة المعرفة
تم إنشاء قاعدة المعرفة بنجاح.
Exllama (مثلاً /content/Phi-3.5-mini-instruct-ex12): /content/Phi-3.5-mini-instruct-ex

```

أدخل سؤالك (أو اكتب 'خروج' للخروج)

جارى البحث عن "What is my name?"

Executing command: python exllama_inference.py --model /content/Phi-3.5-mini-instruct-ex12 --prompt my name is very good

السؤال:

"What is my name?"

--max_new_tokens 64 --temperature 0.7 --top_p 0.9 الإجابة

حدث خطأ أثناء استدعاء Exllama: Command '['python', 'exllama_inference.py', '--model', '/content/Phi-3.5-mini-instruct-ex12']' stderr: python3: can't open file '/content/exllama_inference.py': [Errno 2] No such file or directory

الإجابة:

لم يتم توليد إجابة.

أدخل سؤالك (أو اكتب 'خروج' للخروج): خروج
تم إنتهاء البرنامج.

```

1 git clone https://github.com/turboderp/exllamav2
2 cd exllamav2
3 pip install -r requirements.txt
4 pip install .
5
6 python test_inference.py -m <path_to_model> -p "Once upon a time,"
7 # Append the '--gpu_split auto' flag for multi-GPU inference

```

```
1 !git clone https://github.com/turboderp/exllamav2
2 %cd exllamav2
3 !pip install -r requirements.txt
4 !pip install .
5
6
```



```
Created wheel for exllamav2: filename=exllamav2-0.2.8-cp311-cp311-linux_x86_64.whl size=54493
Stored in directory: /tmp/pip-ephem-wheel-cache-akcrcvjk/wheels/b4/4b/d2/15c8ae14306a022cd079
Successfully built exllamav2
Installing collected packages: exllamav2
Successfully installed exllamav2-0.2.8
```

1 Start coding or generate with AI.

1 Start coding or generate with AI.

1 Start coding or generate with AI.

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f"جاری استخراج النص من الملف {pdf_path}")
10    text = ""
11    try:
12        pdf_reader = PdfReader(pdf_path)
13        for page in pdf_reader.pages:
14            page_text = page.extract_text()
15            if page_text:
16                # تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
17                page_text = page_text.replace("\r", " ").replace(
18                    "\t", " ")
19            text += page_text + "\n"
20    except Exception as e:
21        print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
22    return text
23
24
25 def split_text(text):
26     print("...جاری تقسیم النص إلى أجزاء")
27     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, ch
28     chunks = splitter.split_text(text)
29     print(f"تم تقسیم النص إلى {len(chunks)} جزء")
30     return chunks
```

31

```
32 def create_knowledge_base(chunks, embedding_model):  
33     print("جارٍ إنشاء قاعدة المعرفة...")  
34     kb = FAISS.from_texts(chunks, embedding_model)  
35     print("تم إنشاء قاعدة المعرفة بنجاح.")  
36     return kb  
37
```

```
38 def search_knowledge_base(kb, query, k=4):  
39     print(f"جارٍ البحث عن '{query}'")  
40     docs = kb.similarity_search(query, k=k)  
41     context = "\n\n".join(doc.page_content for doc in docs)  
42     return context  
43
```

```
44 def generate_answer_exllama(model_path, prompt):  
45     """
```

مع تمرير البرومبت والنموذج `exllama_inference.py` يقوم هذا الدالة باستدعاء سكريبت `exllama_inference.py` (يتم إضافة معلمات إضافية للتوليد).

```
46     مع تمرير البرومبت والنموذج exllama_inference.py يقوم هذا الدالة باستدعاء سكريبت  
47     exllama_inference.py يتم إضافة معلمات إضافية للتوليد.  
48     """
```

```
49     command = [  
50         "python", "exllama_inference.py",  
51         "--model", model_path,  
52         "--prompt", prompt,  
53         "--max_new_tokens", "64",  
54         "--temperature", "0.7",  
55         "--top_p", "0.9"  
56     ]  
57     print("Executing command:", " ".join(command))
```

```
58     try:
```

```
59         result = subprocess.run(command, capture_output=True, text=True)  
60         output = result.stdout.strip()  
61         return output  
62     except subprocess.CalledProcessError as e:  
63         print("حدث خطأ أثناء استدعاء Exllama:", e)  
64         print("stderr:", e.stderr)  
65         return None  
66
```

```
67 def main():
```

```
68     print("يُستخدم نموذج RAG مع كتاب PDF في Exllama")  
69     print("—" * 50)
```

```
70
```

```
71     # طلب مسار ملف PDF
```

```
72     pdf_path = input("أدخل مسار ملف PDF مثلاً: /content/drive/MyDrive/
```

```
73     text = extract_text_from_pdf(pdf_path)
74     if not text:
75         return
76
77     # تقسيم النص وإنشاء قاعدة المعرفة
78     chunks = split_text(text)
79
80     print("...جارٍ تحميل نموذج التضمين")
81     embedding_model = HuggingFaceEmbeddings(model_name="sentence-t5")
82     kb = create_knowledge_base(chunks, embedding_model)
83
84     # طلب مسار نموذج Exllama
85     model_path = input("أدخل مسار نموذج Exllama (مثلاً: /content/Phi-3.5")
86
87     # حلقة استعلام المستخدم
88     while True:
89         query = input("\nأدخل سؤالك (أو اكتب 'خروج' للخروج).strip()")
90         if query.lower() in ['خروج', 'exit']:
91             break
92
93         context = search_knowledge_base(kb, query)
94         prompt = f":السؤال\n{n{context}}\n{n{query}}\n\n{n{answer}}:الإجابة"
95
96         answer = generate_answer_exllama(model_path, prompt)
97
98         print("\nالإجابة:")
99         print("-" * 50)
100        if answer:
101            print(answer)
102        else:
103            print("لم يتم توليد إجابة")
104            print("-" * 50)
105
106        print("تم إنتهاء البرنامج")
107
108 if __name__ == "__main__":
109     main()
110
```

☞ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```
KeyboardInterrupt                                     Traceback (most recent call last)
<ipython-input-15-75bcb7a402b0> in <cell line: 0>()
      107
      108 if __name__ == "__main__":
--> 109     main()

----- 2 frames -----
/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
1217         except KeyboardInterrupt:
1218             # re-raise KeyboardInterrupt, to truncate traceback
-> 1219             raise KeyboardInterrupt("Interrupted by user") from None
1220         except Exception:
1221             self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

/content/exllamav2/test_inference.py

شغل

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f"جارٍ استخراج النص من الملف {pdf_path}")
10    text = ""
11    try:
12        pdf_reader = PdfReader(pdf_path)
13        for page in pdf_reader.pages:
14            page_text = page.extract_text()
15            if page_text:
16                # يمكنك تعديل ذلك حسب الحاجة
17                page_text = page_text.replace("\r", "\n").replace(
18                    "\t", " ")
19                text += page_text + "\n"
20
21    except Exception as e:
22        print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
23
تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
(".حرف من النص تم استخراج تم انتزاعه من الملف")
20    return text
21
22
23
تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
(".حرف من النص تم استخراج تم انتزاعه من الملف")
20    return text
21
22
23
```

```

24
25 def split_text(text):
26     ... جاري تقسيم النص إلى أجزاء...")
27     ... splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chun...
28     ... chunks = splitter.split_text(text)
29     ... print(f"تم تقسيم النص إلى {len(chunks)} جزء")
30     ... return chunks
31
32 def create_knowledge_base(chunks, embedding_model):
33     ... جاري إنشاء قاعدة المعرفة...")
34     ... kb = FAISS.from_texts(chunks, embedding_model)
35     ... print("تم إنشاء قاعدة المعرفة بنجاح")
36     ... return kb
37
38 def search_knowledge_base(kb, query, k=4):
39     ... جاري البحث عن {query}")
40     ... docs = kb.similarity_search(query, k=k)
41     ... context = "\n\n".join(doc.page_content for doc in docs)
42     ... return context
43
44 def generate_answer_exllama(model_path, prompt):
45     """
46     ... يقوم هذا الدالة باستدعاء سكريبت الاستدلال الموجود في ...
47     ... /content/exllamav2/test_inference.py
48     ... مع تمرير البرومبت والنموذج ومعلمات التوليد ...
49     """
50     command = [
51         ... "python", ... /content/exllamav2/test_inference.py,
52         ... "--model", ... model_path,
53         ... "--prompt", ... prompt,
54         ... "--max_new_tokens", ... "64",
55         ... "--temperature", ... "0.7",
56         ... "--top_p", ... "0.9"
57     ]
58     print("Executing command:", ... .join(command))
59     try:
60         ... result = subprocess.run(command, capture_output=True, text...
61         ... output = result.stdout.strip()
62         ... return output
63     except subprocess.CalledProcessError as e:
64         ... print("حدث خطأ أثناء استدعاء Exllama:", ... e)
65         ... print("stderr:", ... e.stderr)
66         ... return None

```

```
67
68 def main():
69     print("RAG مع كتاب PDF بستخدام نموذج Exllama")
70     print("-" * 50)
71
72     # طلب مسار ملف PDF
73     pdf_path = input("أدخل مسار ملف PDF مثال: /content/drive/MyDrive")
74     text = extract_text_from_pdf(pdf_path)
75     if not text:
76         return
77
78     # تقسيم النص وإنشاء قاعدة المعرفة
79     chunks = split_text(text)
80
81     print("جارٍ تحميل نموذج التضمين")
82     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
83     kb = create_knowledge_base(chunks, embedding_model)
84
85     # طلب مسار نموذج Exllama
86     model_path = input("أدخل مسار نموذج Exllama مثال: /content/FineTunedModels")
87
88     # حلقة استعلام المستخدم
89     while True:
90         query = input("\nأدخل سؤالك (أو اكتب 'خروج' للخروج)\n")
91         if query.lower() in ['خروج', 'exit']:
92             break
93
94         context = search_knowledge_base(kb, query)
95         prompt = f"\n{context}\n\n{n{query}}\n\n"
96
97         answer = generate_answer_exllama(model_path, prompt)
98
99         print("\nإجابة:")
100        print("-" * 50)
101        if answer:
102            print(answer)
103        else:
104            print("لم يتم توليد إجابة")
105            print("-" * 50)
106
107        print("تم إنتهاء البرنامج")
108
109 if __name__ == "__main__":
110     main()
```

```
110     main()
111
```

→ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

=====
أدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/book.pdf): /content/g.pdf
جارٍ استخراج النص من الملف: /content/g.pdf

تم استخراج 21 حرف من النص.

جارٍ تقسيم النص إلى أجزاء...

تم تقسيم النص إلى 1 جزء.

جارٍ تحميل نموذج التضمين...

```
<ipython-input-1-2e6a132644de>:82: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings`  
embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")  
/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>). 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 data
warnings.warn(

جارٍ إنشاء قاعدة المعرفة...

تم إنشاء قاعدة المعرفة بنجاح.

جارٍ إدخال مسار نموذج Exllama (مثلاً: /content/Phi-3.5-mini-instruct-ex12): /content/Phi-3.5-mini-instruct-ex

أدخل سؤالك (أو اكتب 'خروج' للخروج): "What is my name?"

جارٍ البحث عن: "What is my name?"

Executing command: python /content/exllamav2/test_inference.py --model /content/Phi-3.5-mini-instantiation
my name is very good

السؤال:

"What is my name?"

الإجابة: --max_new_tokens 64 --temperature 0.7 --top_p 0.9

Exllama: Command '['python', '/content/exllamav2/test_inference.py', '--model',
stderr: usage: test_inference.py [-h] [-ed EVAL_DATASET] [-er EVAL_ROWS] [-el EVAL_LENGTH] [-et]

[-eq4] [-eq6] [-eq8] [-ec1] [-p PROMPT] [-pnb] [-t TOKENS] [-ps] [-s]
[-mix MIX_LAYERS] [-nwu] [-sl] [-sp {wiki2}] [-rr RANK_REDUCE]
[-mol MAX_OUTPUT_LEN] [-m MODEL_DIR] [-gs GPU_SPLIT] [-tp] [-l LENGTH]
[-rs ROPE_SCALE] [-ra ROPE_ALPHA] [-ry ROPE_YARN] [-nfa] [-nxr] [-nsdpa]
[-ng] [-lm] [-ept EXPERTS_PER_TOKEN] [-lq4] [-fst] [-ic]
[-chunk CHUNK_SIZE]

test_inference.py: error: unrecognized arguments: --max_new_tokens 64 --temperature 0.7 --top_p 0.9

الإجابة:

لم يتم توليد إجابة.

أدخل سؤالك (أو اكتب 'خروج' للخروج): خروج
تم إنهاء البرنامج.

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
```

```

6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f": جاري استخراج النص من الملف {pdf_path}")
10    text = ""
11    try:
12        pdf_reader = PdfReader(pdf_path)
13        for page in pdf_reader.pages:
14            page_text = page.extract_text()
15            if page_text:
16                # تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
17                page_text = page_text.replace("\r", " ").replace("``", "")
18                text += page_text + "\n"
19        print(f": تم استخراج {len(text)} حرف من النص")
20    return text
21 except Exception as e:
22     print(f": حدث خطأ أثناء قراءة ملف PDF: {e}")
23     return None
24
25 def split_text(text):
26     print("...جارى تقسيم النص إلى أجزاء...")
27     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
28     chunks = splitter.split_text(text)
29     print(f": تم تقسيم النص إلى {len(chunks)} جزء")
30     return chunks
31
32 def create_knowledge_base(chunks, embedding_model):
33     print("...جارى إنشاء قاعدة المعرفة...")
34     kb = FAISS.from_texts(chunks, embedding_model)
35     print("تم إنشاء قاعدة المعرفة بنجاح")
36     return kb
37
38 def search_knowledge_base(kb, query, k=4):
39     print(f": جاري البحث عن {query}")
40     docs = kb.similarity_search(query, k=k)
41     context = "\n\n".join(doc.page_content for doc in docs)
42     return context
43
44 def generate_answer_exllama(model_path, prompt):
45     """
46     يقوم هذا الدالة باستدعاء سكريبت الاستدلال الموجود في
47     /content/exllamav2/test_inference.py
48     مع تمرير البرومبت والنموذج ومعلمات التوليد

```

```
49 """
50     command = [
51         "python", "/content/exllamav2/test_inference.py",
52         "--model", model_path,
53         "--prompt", prompt,
54         "--max_new_tokens", "64",
55         "--temperature", "0.7",
56         "--top_p", "0.9"
57     ]
58     print("Executing command:", " ".join(command))
59     try:
60         result = subprocess.run(command, capture_output=True, text=True)
61         output = result.stdout.strip()
62         return output
63     except subprocess.CalledProcessError as e:
64         print("حدث خطأ أثناء استدعاء Exllama:", e)
65         print("stderr:", e.stderr)
66         return None
67
68 def main():
69     print(" باستخدام نموذج PDF مع كتاب RAG برنامج Exllama")
70     print("=" * 50)
71
72     # طلب مسار ملف PDF
73     pdf_path = input("أدخل مسار ملف PDF مثال: /content/drive/MyDrive")
74     text = extract_text_from_pdf(pdf_path)
75     if not text:
76         return
77
78     # تقسيم النص وإنشاء قاعدة المعرفة
79     chunks = split_text(text)
80
81     print("جارٍ تحميل نموذج التضمين...")
82     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
83     kb = create_knowledge_base(chunks, embedding_model)
84
85     # طلب مسار نموذج Exllama
86     model_path = input("أدخل مسار نموذج Exllama مثال: /content/FineTunedModels")
87
88     # حلقة استعلام المستخدم
89     while True:
90         query = input("\n退出 للخروج\n")
91         if query.lower() in ['退出', 'exit']:
```

```

92         break
93
94     context = search_knowledge_base(kb, query)
95     prompt = f"\n\n{n{context}}\n{n{query}}\n\nجابة"
96
97     answer = generate_answer_exllama(model_path, prompt)
98
99     print("\nإجابة:")
100    print("-" * 50)
101    if answer:
102        print(answer)
103    else:
104        print("لم يتم توليد إجابة")
105    print("-" * 50)
106
107    print("تم إنتهاء البرنامج")
108
109 if __name__ == "__main__":
110     main()
111

```

➡️ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```

=====
فأدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/book.pdf): /content/g.pdf
جارٍ استخراج النص من الملف /content/g.pdf
تم استخراج 21 حرف من النص.
جارٍ تقسيم النص إلى أجزاء...
تم تقسيم النص إلى 1 جزء.
جارٍ تحميل نموذج التضمين...
جارٍ إنشاء قاعدة المعرفة...
تم إنشاء قاعدة المعرفة بنجاح.
تم إنشاء قاعدة المعرفة بنجاح
Exllama (مثلاً: /content/Phi-3.5-mini-instruct-ex12): /content/Phi-3.5-mini-instruct-ex12

```

أدخل سؤالك (أو اكتب 'خروج' للخروج)

جارٍ البحث عن: What is my name?

Executing command: python /content/exllamav2/test_inference.py --model /content/Phi-3.5-mini-instruct-ex12 --max_new_tokens 64 --temperature 0.7 --top_p 0.9
my name is very good

سؤال:

What is my name?

```

الإجابة: --max_new_tokens 64 --temperature 0.7 --top_p 0.9
Exllama: Command '['python', '/content/exllamav2/test_inference.py', '--model',
stderr: usage: test_inference.py [-h] [-ed EVAL_DATASET] [-er EVAL_ROWS] [-el EVAL_LENGTH] [-et
                                [-eq4] [-eq6] [-eq8] [-ecl] [-p PROMPT] [-pn] [-t TOKENS] [-ps] [-s]
                                [-mix MIX_LAYERS] [-nwu] [-sl] [-sp {wiki2}] [-rr RANK_REDUCE]
                                [-mol MAX_OUTPUT_LEN] [-m MODEL_DIR] [-gs GPU_SPLIT] [-tp] [-l LENGTH]
                                [-rs ROPE_SCALE] [-ra ROPE_ALPHA] [-ry ROPE_YARN] [-nfa] [-nx] [-nsdp]
                                [-ng] [-lm] [-ept EXPERTS_PER_TOKEN] [-lq4] [-fst] [-ic]
                                [-chunk CHUNK_SIZE]
test_inference.py: error: unrecognized arguments: --max_new_tokens 64 --temperature 0.7 --top_p 0.9

```

الإجابة:

لم يتم توليد إجابة.

أدخل سؤالك (أو اكتب 'خروج' للخروج) : خروج
تم إنتهاء البرنامج.

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 def extract_text_from_pdf(pdf_path):
9     print(f"جارٍ استخراج النص من الملف {pdf_path}")
10    text = ""
11    try:
12        pdf_reader = PdfReader(pdf_path)
13        for page in pdf_reader.pages:
14            page_text = page.extract_text()
15            if page_text:
16                # تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
17                page_text = page_text.replace("\r", " ").replace(
18                    "\t", " ")
19                text += page_text + "\n"
20        print(f"تم استخراج {len(text)} حرف من النص")
21        return text
22    except Exception as e:
23        print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
24        return None
25
26 def split_text(text):
27     print("جارٍ تقسيم النص إلى أجزاء...")
28     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, ch
29    unks = splitter.split_text(text)
30     print(f"تم تقسيم النص إلى {len(chunks)} جزء")
31     return chunks
32
33 def create_knowledge_base(chunks, embedding_model):
34     print("جارٍ إنشاء قاعدة المعرفة...")
35     kb = FAISS.from_texts(chunks, embedding_model)
36     print("تم إنشاء قاعدة المعرفة بنجاح")
```

```
36     return kb
37
38 def search_knowledge_base(kb, query, k=4):
39     print(f"جاری البحث عن: {query}")
40     docs = kb.similarity_search(query, k=k)
41     context = "\n\n".join(doc.page_content for doc in docs)
42     return context
43
44 def generate_answer_exllama(model_path, prompt):
45     """
46     الموجود في هذا الدالة باستدعاء سكريبت test_inference.py
47     /content/exllamav2/test_inference.py
48     مع تمرير نص الاستعلام والنموذج ومعامل الحد الأقصى لطول الناتج.
49     """
50     command = [
51         "python", "/content/exllamav2/test_inference.py",
52         "-m", model_path,
53         "-p", prompt,
54         "-mol", "64"
55     ]
56     print("Executing command:", " ".join(command))
57     try:
58         result = subprocess.run(command, capture_output=True, text=True)
59         output = result.stdout.strip()
60         return output
61     except subprocess.CalledProcessError as e:
62         print("حدث خطأ أثناء استدعاء Exllama:", e)
63         print("stderr:", e.stderr)
64     return None
65
66 def main():
67     print(" باستخدام نموذج PDF مع كتاب RAG")
68     print("==" * 50)
69
70     # طلب مسار ملف PDF
71     pdf_path = input("أدخل مسار ملف PDF مثل: /content/drive/MyDrive/")
72     text = extract_text_from_pdf(pdf_path)
73     if not text:
74         return
75
76     # تقسيم النص وإنشاء قاعدة المعرفة
77     chunks = split_text(text)
```

```

78     print("...جارٍ تحميل نموذج التضمين")
79     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
80     kb = create_knowledge_base(chunks, embedding_model)
81
82
83     # طلب مسار نموذج Exllama
84     model_path = input("أدخل مسار نموذج (مثلاً: /content/Phi-3.5")
85
86     # حلقة استعلام المستخدم
87     while True:
88         query = input("\nأدخل سؤالك (أو اكتب 'خروج' للخروج)\n").strip()
89         if query.lower() in ['خروج', 'exit']:
90             break
91
92         context = search_knowledge_base(kb, query)
93         prompt = f":السؤال\n{n{context}}\n{n{query}}\n\nالإجابة"
94
95         answer = generate_answer_exllama(model_path, prompt)
96
97         print("\nالإجابة:")
98         print("-" * 50)
99         if answer:
100            print(answer)
101        else:
102            print("لم يتم توليد إجابة")
103            print("-" * 50)
104
105        print("\nتم إنتهاء البرنامج")
106
107 if __name__ == "__main__":
108     main()
109

```

➡️ باستخراج النص من ملف PDF، تم توليد إجابة بـ RAG.

أدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/book.pdf): /content/g.pdf

جارٍ استخراج النص من الملف

تم استخراج 21 حرف من النص.

جارٍ تقسيم النص إلى أجزاء

تم تقسيم النص إلى 1 جزء.

جارٍ تحميل نموذج التضمين...

<ipython-input-1-e585a4836fd7>:80: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings`

embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")

/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>)

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 data
warnings.warn(
    ...جاری إنشاء قاعدة المعرفة...
    ...تم إنشاء قاعدة المعرفة بنجاح.
    ...أدخل مسار نموذج Exllama (مثلاً: /content/Phi-3.5-mini-instruct-ex12): /content/Phi-3.5-mini-instruct-e
```

ادخل سؤالك (أو اكتب 'خروج' للخروج)?

جاری البحث عن: what is my name?

```
Executing command: python /content/exllamav2/test_inference.py -m /content/Phi-3.5-mini-instruct-
my name is very good
```

السؤال:

what is my name?

الإجابة: -mol 64

```
Exllama: Command '['python', '/content/exllamav2/test_inference.py', '-m', '/content/Phi-3.5-mini-instruct-e']'
stderr: Traceback (most recent call last):
```

```
File "/content/exllamav2/test_inference.py", line 192, in <module>
    cache = ExLlamaV2Cache(model) if not model.tp_context else ExLlamaV2Cache_TP(model)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^

File "/content/exllamav2/exllamav2/cache.py", line 256, in __init__
    self.create_state_tensors(copy_from, lazy)
File "/content/exllamav2/exllamav2/cache.py", line 91, in create_state_tensors
    p_key_states = torch.zeros(self.shape_wk, dtype = self.dtype, device = device).contiguous()
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

`torch.OutOfMemoryError: CUDA out of memory. Tried to allocate 768.00 MiB. GPU 0 has a total capacity of 12.00 GiB.`

الإجابة:

لم يتم توليد إجابة.

ادخل سؤالك (أو اكتب 'خروج' للخروج): خروج
تم إنتهاء البرنامج.

```
1 import os
2 os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"
3
```

```
1 import os
2 import subprocess
3 from PyPDF2 import PdfReader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5 from langchain.embeddings import HuggingFaceEmbeddings
6 from langchain.vectorstores import FAISS
7
8 #تعيين متغير البيئة لتفادي تجزئة الذاكرة
9 os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"
10
11 def extract_text_from_pdf(pdf_path):
12     print(f"جارٍ استخراج النص من الملف {pdf_path}")
```

```

13     text = ""
14     try:
15         pdf_reader = PdfReader(pdf_path)
16         for page in pdf_reader.pages:
17             page_text = page.extract_text()
18             if page_text:
19                 # تنظيف بسيط للنص؛ يمكنك تعديل ذلك حسب الحاجة
20                 page_text = page_text.replace("\r", " ").replace(
21                     "\n", "\n")
22                 print(f"تم استخراج {len(text)} حرف من النص")
23             return text
24     except Exception as e:
25         print(f"حدث خطأ أثناء قراءة ملف PDF: {e}")
26     return None
27
28 def split_text(text):
29     print("جارٍ تقسيم النص إلى أجزاء...")
30     splitter = RecursiveCharacterTextSplitter(chunk_size=1000, ch
31    unks = splitter.split_text(text)
32     print(f"تم تقسيم النص إلى {len(chunks)} جزء")
33     return chunks
34
35 def create_knowledge_base(chunks, embedding_model):
36     print("جارٍ إنشاء قاعدة المعرفة...")
37     kb = FAISS.from_texts(chunks, embedding_model)
38     print("تم إنشاء قاعدة المعرفة بنجاح")
39     return kb
40
41 def search_knowledge_base(kb, query, k=4):
42     print(f"جارٍ البحث عن {query}")
43     docs = kb.similarity_search(query, k=k)
44     context = "\n\n".join(doc.page_content for doc in docs)
45     return context
46
47 def generate_answer_exllama(model_path, prompt):
48     """
49     الموجود في هذا الدالة باستدعاء سكريبت
50     /content/exllamav2/test_inference.py
51     مع تمرير نص الاستعلام والنموذج ومعامل الحد الأقصى لطول الناتج.
52     """
53     command = [
54         "python", "/content/exllamav2/test_inference.py",

```

```
55     "-m", model_path,
56     "-p", prompt,
57     "-l", "32",
58     "-mol", "32"
59 ]
60 print("Executing command:", " ".join(command))
61 try:
62     result = subprocess.run(command, capture_output=True, text=True)
63     output = result.stdout.strip()
64     return output
65 except subprocess.CalledProcessError as e:
66     print("حدث خطأ أثناء استدعاء Exllama:", e)
67     print("stderr:", e.stderr)
68     return None
69
70 def main():
71     print(" باستخدام نموذج PDF مع كتاب RAG")
72     print("=". * 50)
73
74     # طلب مسار ملف PDF
75     pdf_path = input("أدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/")
76     text = extract_text_from_pdf(pdf_path)
77     if not text:
78         return
79
80     # تقسيم النص وإنشاء قاعدة المعرفة
81     chunks = split_text(text)
82
83     print("جارٍ تحميل نموذج التضمين...")
84     embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
85     kb = create_knowledge_base(chunks, embedding_model)
86
87     # طلب مسار نموذج Exllama
88     model_path = input("أدخل مسار نموذج Exllama (مثلاً: /content/Phi-3.5/)")
89
90     # حلقة استعلام المستخدم
91     while True:
92         query = input("\nالآن: أدخل سؤالك (أو اكتب 'خروج' للخروج).strip()")
93         if query.lower() in ['خروج', 'exit']:
94             break
95
96         context = search_knowledge_base(kb, query)
```

```

97     prompt = f"\nالسؤال:{context}\nالسؤال:{query}\n\n"
98
99     answer = generate_answer_exllama(model_path, prompt)
100
101    print("\nالإجابة:")
102    print("-" * 50)
103    if answer:
104        print(answer)
105    else:
106        print("لم يتم توليد إجابة")
107    print("-" * 50)
108
109    print("تم إنتهاء البرنامج")
110
111 if __name__ == "__main__":
112     main()
113

```

→ باستخدام نموذج PDF مع كتاب RAG برنامج Exllama

```

=====
فأدخل مسار ملف PDF (مثلاً: /content/drive/MyDrive/book.pdf): /content/g.pdf
جارٍ استخراج النص من الملف /content/g.pdf
تم استخراج 21 حرف من النص.
جارٍ تقسيم النص إلى أجزاء...
تم تقسيم النص إلى 1 جزء.
جارٍ تحميل نموذج التضمين...
<ipython-input-1-e72c2285d504>:84: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings` embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
/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)
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 data
warnings.warn(
جارٍ إنشاء قاعدة المعرفة...
تم إنشاء قاعدة المعرفة بنجاح.
جارٍ تحميل نموذج Exllama (مثلاً: /content/Phi-3.5-mini-instruct-ex12): /content/Phi-3.5-mini-instruct-ex12

```

أدخل سؤالك (أو اكتب 'خروج' للخروج)

جارٍ البحث عن: what is my name?

Executing command: python /content/exllamav2/test_inference.py -m /content/Phi-3.5-mini-instruct-my name is very good

السؤال:

what is my name?

-1 32 -mol 32 : الإجابة

Exllama: Command '['python', '/content/exllamav2/test_inference.py', '-m', '/content/Phi-3.5-mini-instruct-my name is very good']' stderr: Traceback (most recent call last):

```

File "/content/exllamav2/test_inference.py", line 214, in <module>
    output = generator.generate_simple(args.prompt, settings, args.tokens, token_healing = True,
                                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

```

File "/content/exllamav2/exllamav2/generator/base.py", line 265, in generate_simple
    indexed_embeddings = input_embeddings.float().cpu()

```

^^^^^

```
AttributeError: 'NoneType' object has no attribute 'float'
```

:الإجابة

لم يتم توليد إجابة.

أدخل سؤالك (أو اكتب 'خروج' للخروج) : خروج
تم إنهاء البرنامج.

```
1 command = [  
2     "python", "/content/exllamav2/test_inference.py",  
3     "-m", model_path,  
4     "-p", prompt,  
5     "-l", "128",  
6     "-mol", "32"  
7 ]  
8
```

```
1  
2 !python /content/exllamav2/test_inference.py -http  
3
```

```
→ -eq6, --eval_token_q6          Evaluate perplexity on token-by-token inference using Q4 cache  
→ -eq8, --eval_token_q8          Evaluate perplexity on token-by-token inference using Q6 cache  
→ -ecl, --eval_context_lens    Evaluate perplexity at range of context lengths  
→ -eq16, --eval_token_q16        Evaluate perplexity on token-by-token inference using Q8 cache
```

```

-gs GPU_SPLIT, --gpu_split GPU_SPLIT
    "auto", or VRAM allocation per GPU in GB. "auto" is implied by default
    tensor-parallel mode.

-tp, --tensor_parallel
    Load in tensor-parallel mode (not fully supported for all models)

-l LENGTH, --length LENGTH
    Maximum sequence length

-rs ROPE_SCALE, --rope_scale ROPE_SCALE
    RoPE scaling factor

-ra ROPE_ALPHA, --rope_alpha ROPE_ALPHA
    RoPE alpha value (NTK)

-ry ROPE_YARN, --rope_yarn ROPE_YARN
    Set RoPE YaRN factor (use default max_seq_len as
    original_max_position_embeddings if not configured)

-nfa, --no_flash_attn
    Disable Flash Attention

-nxf, --no_xformers
    Disable xformers, an alternative plan of flash attn for older devices

-nsdpa, --no_sdpa
    Disable Torch SDPA

-ng, --no_graphs
    Disable Graphs

-lm, --low_mem
    Enable VRAM optimizations, potentially trading off speed

-ept EXPERTS_PER_TOKEN, --experts_per_token EXPERTS_PER_TOKEN
    Override MoE model's default number of experts per token

-lq4, --load_q4
    Load weights in Q4 mode

-fst, --fast_safetensors
    Deprecated (does nothing)

-ic, --ignore_compatibility
    Do not override model config options in case of compatibility issues

-chunk CHUNK_SIZE, --chunk_size CHUNK_SIZE

```

1

2 !python /content/exllamav2/test_inference.py -http

3

4

5 usage: test_inference.py [-h] [-ed EVAL_DATASET] [-er EVAL_ROWS] [-
 6 [-eq4] [-eq6] [-eq8] [-ecl] [-p PROMPT] [-]
 7 [-mix MIX_LAYERS] [-nwu] [-s1] [-sp {wiki2
 8 [-mol MAX_OUTPUT_LEN] [-m MODEL_DIR] [-gs '
 9 [-rs ROPE_SCALE] [-ra ROPE_ALPHA] [-ry ROP
 10 [-ng] [-lm] [-ept EXPERTS_PER_TOKEN] [-lq4
 11 [-chunk CHUNK_SIZE]

12

13 Test inference on ExLlamaV2 model

14

15 options:

16 -h, --help show this help message and exit
 17 -ed EVAL_DATASET, --eval_dataset EVAL_DATASET
 18 Perplexity evaluation dataset (.parquet file)
 19 -er EVAL_ROWS, --eval_rows EVAL_ROWS
 20 Number of rows to apply from dataset (default 1000)
 21 -el EVAL_LENGTH, --eval_length EVAL_LENGTH
 22 Max no. tokens per sample