

SOURCE

May 14, 2025

- Projeto Principal - Main
-

1 Primeira parte

Essa parte é foca em carregar o pdf e organizar os dados do pdf em uma pasta source para o modelo.

```
[1]: import os
import fitz # PyMuPDF
import faiss
import numpy as np
import pickle
from sentence_transformers import SentenceTransformer
import textwrap

# Mostrar os arquivos gerados
import pandas as pd
```

```
/home/darkcover/.cache/pypoetry/virtualenvs/leitor-pdf-S8aY7ORV-
py3.12/lib/python3.12/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not
found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
from .autonotebook import tqdm as notebook_tqdm
```

```
[2]: # Parâmetros
CHUNK_SIZE = 300 # caracteres por chunk
PDF_PATH = "/home/darkcover/Documents/Leitor_PDF/documentos/
↳SAGE_ManCfg_Anxi17_61850 1.pdf"
OUTPUT_DIR = "/home/darkcover/Documents/Leitor_PDF/pyproject/phi_2_Q4/
↳partes_pdf"
os.makedirs(OUTPUT_DIR, exist_ok=True)

# Modelo de embeddings leve
embed_model = SentenceTransformer('paraphrase-albert-small-v2')
```

```
[3]: # Função para dividir PDF em partes menores (ex: 10 páginas cada)
def dividir_pdf_em_partes(pdf_path, paginas_por_parte=10):
    doc = fitz.open(pdf_path)
```

```

num_paginas = doc.page_count
partes = []

for i in range(0, num_paginas, paginas_por_parte):
    subdoc = fitz.open()
    for j in range(i, min(i + paginas_por_parte, num_paginas)):
        subdoc.insert_pdf(doc, from_page=j, to_page=j)
        parte_path = os.path.join(OUTPUT_DIR, f"parte_{i // paginas_por_parte + 1}.pdf")
        subdoc.save(parte_path)
        partes.append(parte_path)

    return partes

# Extrair texto do PDF
def extrair_texto(pdf_path):
    doc = fitz.open(pdf_path)
    texto = ""
    for pagina in doc:
        texto += pagina.get_text()
    return texto

# Quebrar texto em chunks menores
def chunk_text(text, max_chars=CHUNK_SIZE):
    return textwrap.wrap(text, width=max_chars)

# Salvar FAISS e chunks
def salvar_index_e_chunks(chunks, nome_base):
    embeddings = embed_model.encode(chunks)
    index = faiss.IndexFlatL2(embeddings.shape[1])
    index.add(np.array(embeddings))

    faiss.write_index(index, f"{nome_base}.faiss")
    with open(f"{nome_base}_chunks.pkl", "wb") as f:
        pickle.dump(chunks, f)

# Pipeline principal de processamento
def processar_partes(pdf_path):
    partes = dividir_pdf_em_partes(pdf_path)
    for parte in partes:
        texto = extrair_texto(parte)
        chunks = chunk_text(texto)
        base = os.path.splitext(parte)[0]
        salvar_index_e_chunks(chunks, base)

```

```

[4]: # Rodar o pipeline
processar_partes(PDF_PATH)

```

```
arquivos_gerados = os.listdir(OUTPUT_DIR)
df = pd.DataFrame(arquivos_gerados, columns=["Arquivo"])
print(df) # ou use display(df) se estiver no Jupyter Notebook
```

```

      Arquivo
0  parte_6_chunks.pkl
1      parte_3.faiss
2      parte_5.pdf
3  parte_2_chunks.pkl
4      parte_1.faiss
5      parte_4.faiss
6      parte_6.faiss
7  parte_4_chunks.pkl
8      parte_2.pdf
9      parte_6.pdf
10 parte_5_chunks.pkl
11      parte_2.faiss
12      parte_4.pdf
13      parte_1.pdf
14      parte_5.faiss
15 parte_3_chunks.pkl
16      parte_3.pdf
17 parte_1_chunks.pkl
```

2 Segunda Parte

```
[5]: import os
import faiss
import pickle
import numpy as np
from sentence_transformers import SentenceTransformer
from llama_cpp import Llama
```

```
[6]: # Parâmetros
PARTES_DIR = "/home/darkcover/Documentos/Leitor_PDF/pyproject/phi_2_Q4/
↳ partes_pdf"
TOP_K = 2 # quantidade de chunks por parte
MAX_CONTEXT = 5 # máximo de chunks para resposta
MODEL_PATH = "/home/darkcover/Documentos/Leitor_PDF/models/phi-2.Q4_K_M.gguf"

# Carrega modelo de embeddings
embed_model = SentenceTransformer("paraphrase-albert-small-v2")

# Carrega modelo LLM local
```

```

llm = Llama(model_path=MODEL_PATH, n_ctx=2048, n_threads=4)

# Função para buscar chunks relevantes em um par de arquivos .faiss + .pkl
def buscar_chunks_relevantes(faiss_path, pkl_path, pergunta):
    with open(pkl_path, "rb") as f:
        chunks = pickle.load(f)

    index = faiss.read_index(faiss_path)
    pergunta_emb = embed_model.encode([pergunta])
    D, I = index.search(np.array(pergunta_emb), TOP_K)
    return [chunks[i] for i in I[0]]

# Junta todos os contextos das partes
def montar_contexto_global(pergunta):
    contextos = []
    arquivos = os.listdir(PARTES_DIR)
    partes = sorted(set(f.split(".")[0] for f in arquivos if f.endswith(".faiss")))

    for parte in partes:
        faiss_file = os.path.join(PARTES_DIR, f"{parte}.faiss")
        chunk_file = os.path.join(PARTES_DIR, f"{parte}_chunks.pkl")
        if os.path.exists(faiss_file) and os.path.exists(chunk_file):
            contextos.extend(buscar_chunks_relevantes(faiss_file, chunk_file,
            pergunta))

    return "\n".join(contextos[:MAX_CONTEXT])

# Gera resposta com modelo local
def gerar_resposta(pergunta):
    contexto = montar_contexto_global(pergunta)
    prompt = f"""\
Você é um assistente inteligente. Use o contexto abaixo para
responder a pergunta.

Contexto:
{contexto}

Pergunta:
{pergunta}

Resposta: """
    resposta = llm(prompt, max_tokens=200)
    return resposta["choices"][0]["text"].strip()

```

llama_model_loader: loaded meta data with 20 key-value pairs and 325 tensors from /home/darkcover/Documentos/Leitor_PDF/models/phi-2.Q4_K_M.gguf (version GGUF V3 (latest))

```

llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not
apply in this output.
llama_model_loader: - kv  0:                      general.architecture str
= phi2
llama_model_loader: - kv  1:                      general.name str
= Phi2
llama_model_loader: - kv  2:                      phi2.context_length u32
= 2048
llama_model_loader: - kv  3:                      phi2.embedding_length u32
= 2560
llama_model_loader: - kv  4:                      phi2.feed_forward_length u32
= 10240
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not
apply in this output.
llama_model_loader: - kv  0:                      general.architecture str
= phi2
llama_model_loader: - kv  1:                      general.name str
= Phi2
llama_model_loader: - kv  2:                      phi2.context_length u32
= 2048
llama_model_loader: - kv  3:                      phi2.embedding_length u32
= 2560
llama_model_loader: - kv  4:                      phi2.feed_forward_length u32
= 10240
llama_model_loader: - kv  5:                      phi2.block_count u32
= 32
llama_model_loader: - kv  6:                      phi2.attention.head_count u32
= 32
llama_model_loader: - kv  7:                      phi2.attention.head_count_kv u32
= 32
llama_model_loader: - kv  8:                      phi2.attention.layer_norm_epsilon f32
= 0.000010
llama_model_loader: - kv  9:                      phi2.rope.dimension_count u32
= 32
llama_model_loader: - kv 10:                      general.file_type u32
= 15
llama_model_loader: - kv 11:                      tokenizer.ggml.add_bos_token bool
= false
llama_model_loader: - kv 12:                      tokenizer.ggml.model str
= gpt2
llama_model_loader: - kv 13:                      tokenizer.ggml.tokens
arr[str,51200] = ["!", "\"", "#", "$", "%", "&", "'", ...
llama_model_loader: - kv 14:                      tokenizer.ggml.token_type
arr[i32,51200] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
llama_model_loader: - kv 15:                      tokenizer.ggml.merges
arr[str,50000] = ["Ġ t", "Ġ a", "h e", "i n", "r e",...
llama_model_loader: - kv 16:                      tokenizer.ggml.bos_token_id u32
= 50256

```

```

llama_model_loader: - kv 17:                tokenizer.ggml.eos_token_id u32
= 50256
llama_model_loader: - kv 18:                tokenizer.ggml.unknown_token_id u32
= 50256
llama_model_loader: - kv 19:                general.quantization_version u32
= 2
llama_model_loader: - type f32: 195 tensors
llama_model_loader: - type q4_K: 81 tensors
llama_model_loader: - type q5_K: 32 tensors
llama_model_loader: - type q6_K: 17 tensors
print_info: file format = GGUF V3 (latest)
print_info: file type   = Q4_K - Medium
print_info: file size   = 1.66 GiB (5.14 BPW)
load: missing pre-tokenizer type, using: 'default'
load:
load: *****
load: GENERATION QUALITY WILL BE DEGRADED!
load: CONSIDER REGENERATING THE MODEL
load: *****
load:
init_tokenizer: initializing tokenizer for type 2
load: special tokens cache size = 944
load: token to piece cache size = 0.3151 MB
print_info: arch           = phi2
print_info: vocab_only      = 0
print_info: n_ctx_train    = 2048
print_info: n_embd         = 2560
print_info: n_layer        = 32
print_info: n_head         = 32
print_info: n_head_kv      = 32
print_info: n_rot          = 32
print_info: n_swa          = 0
print_info: n_swa_pattern  = 1
print_info: n_embd_head_k   = 80
print_info: n_embd_head_v   = 80
print_info: n_gqa          = 1
print_info: n_embd_k_gqa    = 2560
print_info: n_embd_v_gqa    = 2560
print_info: f_norm_eps     = 1.0e-05
print_info: f_norm_rms_eps = 0.0e+00
print_info: f_clamp_kqv    = 0.0e+00
print_info: f_max_alibi_bias = 0.0e+00
print_info: f_logit_scale  = 0.0e+00
print_info: f_attn_scale   = 0.0e+00
print_info: n_ff           = 10240
print_info: n_expert        = 0
print_info: n_expert_used   = 0
print_info: causal attn    = 1

```

```

print_info: pooling type      = 0
print_info: rope type        = 2
print_info: rope scaling     = linear
print_info: freq_base_train  = 10000.0
print_info: freq_scale_train = 1
print_info: n_ctx_orig_yarn  = 2048
print_info: rope_finetuned   = unknown
print_info: ssm_d_conv       = 0
print_info: ssm_d_inner      = 0
print_info: ssm_d_state      = 0
print_info: ssm_dt_rank      = 0
print_info: ssm_dt_b_c_rms   = 0
print_info: model type       = 3B
print_info: model params     = 2.78 B
print_info: general.name     = Phi2
print_info: vocab type       = BPE
print_info: n_vocab          = 51200
print_info: n_merges         = 50000
print_info: BOS token        = 50256 '<|endoftext|>'
print_info: EOS token        = 50256 '<|endoftext|>'
print_info: EOT token        = 50256 '<|endoftext|>'
print_info: UNK token        = 50256 '<|endoftext|>'
print_info: LF token         = 198 'Ĉ'
print_info: EOG token        = 50256 '<|endoftext|>'
print_info: max token length = 256
load_tensors: loading model tensors, this can take a while... (mmap = true)
load_tensors: layer 0 assigned to device CPU, is_swa = 0
load_tensors: layer 1 assigned to device CPU, is_swa = 0
load_tensors: layer 2 assigned to device CPU, is_swa = 0
load_tensors: layer 3 assigned to device CPU, is_swa = 0
load_tensors: layer 4 assigned to device CPU, is_swa = 0
load_tensors: layer 5 assigned to device CPU, is_swa = 0
load_tensors: layer 6 assigned to device CPU, is_swa = 0
load_tensors: layer 7 assigned to device CPU, is_swa = 0
load_tensors: layer 8 assigned to device CPU, is_swa = 0
load_tensors: layer 9 assigned to device CPU, is_swa = 0
load_tensors: layer 10 assigned to device CPU, is_swa = 0
load_tensors: layer 11 assigned to device CPU, is_swa = 0
load_tensors: layer 12 assigned to device CPU, is_swa = 0
load_tensors: layer 13 assigned to device CPU, is_swa = 0
load_tensors: layer 14 assigned to device CPU, is_swa = 0
load_tensors: layer 15 assigned to device CPU, is_swa = 0
load_tensors: layer 16 assigned to device CPU, is_swa = 0
load_tensors: layer 17 assigned to device CPU, is_swa = 0
load_tensors: layer 18 assigned to device CPU, is_swa = 0
load_tensors: layer 19 assigned to device CPU, is_swa = 0
load_tensors: layer 20 assigned to device CPU, is_swa = 0
load_tensors: layer 21 assigned to device CPU, is_swa = 0

```

```

load_tensors: layer 22 assigned to device CPU, is_swa = 0
load_tensors: layer 23 assigned to device CPU, is_swa = 0
load_tensors: layer 24 assigned to device CPU, is_swa = 0
load_tensors: layer 25 assigned to device CPU, is_swa = 0
load_tensors: layer 26 assigned to device CPU, is_swa = 0
load_tensors: layer 27 assigned to device CPU, is_swa = 0
load_tensors: layer 28 assigned to device CPU, is_swa = 0
load_tensors: layer 29 assigned to device CPU, is_swa = 0
load_tensors: layer 30 assigned to device CPU, is_swa = 0
load_tensors: layer 31 assigned to device CPU, is_swa = 0
load_tensors: layer 32 assigned to device CPU, is_swa = 0
load_tensors: tensor 'token_embd.weight' (q4_K) (and 244 others) cannot be used
with preferred buffer type CPU_AARCH64, using CPU instead
load_tensors: CPU_AARCH64 model buffer size = 787.50 MiB
load_tensors: CPU_Mapped model buffer size = 1704.63 MiB
repack: repack tensor blk.0.attn_output.weight with q4_K_8x8
repack: repack tensor blk.0.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.1.attn_output.weight with q4_K_8x8
repack: repack tensor blk.1.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.2.attn_output.weight with q4_K_8x8
repack: repack tensor blk.2.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.3.attn_output.weight with q4_K_8x8
repack: repack tensor blk.3.ffn_down.weight with q4_K_8x8
.repack: repack tensor blk.3.ffn_up.weight with q4_K_8x8
repack: repack tensor blk.4.attn_output.weight with q4_K_8x8
.repack: repack tensor blk.4.ffn_up.weight with q4_K_8x8
repack: repack tensor blk.5.attn_output.weight with q4_K_8x8
.repack: repack tensor blk.5.ffn_down.weight with q4_K_8x8
.repack: repack tensor blk.5.ffn_up.weight with q4_K_8x8
repack: repack tensor blk.6.attn_output.weight with q4_K_8x8
.repack: repack tensor blk.6.ffn_down.weight with q4_K_8x8
repack: repack tensor blk.6.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.7.attn_output.weight with q4_K_8x8
repack: repack tensor blk.7.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.8.attn_output.weight with q4_K_8x8
repack: repack tensor blk.8.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.9.attn_output.weight with q4_K_8x8
repack: repack tensor blk.9.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.10.attn_output.weight with q4_K_8x8
repack: repack tensor blk.10.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.11.attn_output.weight with q4_K_8x8
.repack: repack tensor blk.11.ffn_up.weight with q4_K_8x8
repack: repack tensor blk.12.attn_output.weight with q4_K_8x8
.repack: repack tensor blk.12.ffn_down.weight with q4_K_8x8
repack: repack tensor blk.12.ffn_up.weight with q4_K_8x8
.repack: repack tensor blk.13.attn_output.weight with q4_K_8x8
repack: repack tensor blk.13.ffn_down.weight with q4_K_8x8
.repack: repack tensor blk.13.ffn_up.weight with q4_K_8x8

```



```

llama_context: constructing llama_context
llama_context: n_seq_max      = 1
llama_context: n_ctx         = 2048
llama_context: n_ctx_per_seq = 2048
llama_context: n_batch      = 512
llama_context: n_ubatch     = 512
llama_context: causal_attn   = 1
llama_context: flash_attn    = 0
llama_context: freq_base     = 10000.0
llama_context: freq_scale    = 1
set_abort_callback: call
llama_context:          CPU  output buffer size =      0.20 MiB
create_memory: n_ctx = 2048 (padded)
llama_kv_cache_unified: kv_size = 2048, type_k = 'f16', type_v = 'f16', n_layer
= 32, can_shift = 1, padding = 32
llama_kv_cache_unified: layer   0: dev = CPU
llama_kv_cache_unified: layer   1: dev = CPU
llama_kv_cache_unified: layer   2: dev = CPU
llama_kv_cache_unified: layer   3: dev = CPU
llama_kv_cache_unified: layer   4: dev = CPU
llama_kv_cache_unified: layer   5: dev = CPU
llama_kv_cache_unified: layer   6: dev = CPU
llama_kv_cache_unified: layer   7: dev = CPU
llama_kv_cache_unified: layer   8: dev = CPU
llama_kv_cache_unified: layer   9: dev = CPU
llama_kv_cache_unified: layer  10: dev = CPU
llama_kv_cache_unified: layer  11: dev = CPU
llama_kv_cache_unified: layer  12: dev = CPU
llama_kv_cache_unified: layer  13: dev = CPU
llama_kv_cache_unified: layer  14: dev = CPU
llama_kv_cache_unified: layer  15: dev = CPU
llama_kv_cache_unified: layer  16: dev = CPU
llama_kv_cache_unified: layer  17: dev = CPU
llama_kv_cache_unified: layer  18: dev = CPU
llama_kv_cache_unified: layer  19: dev = CPU
llama_kv_cache_unified: layer  20: dev = CPU
llama_kv_cache_unified: layer  21: dev = CPU
llama_kv_cache_unified: layer  22: dev = CPU
llama_kv_cache_unified: layer  23: dev = CPU
llama_kv_cache_unified: layer  24: dev = CPU
llama_kv_cache_unified: layer  25: dev = CPU
llama_kv_cache_unified: layer  26: dev = CPU
llama_kv_cache_unified: layer  27: dev = CPU
llama_kv_cache_unified: layer  28: dev = CPU
llama_kv_cache_unified: layer  29: dev = CPU
llama_kv_cache_unified: layer  30: dev = CPU
llama_kv_cache_unified: layer  31: dev = CPU
llama_kv_cache_unified:          CPU KV buffer size =    640.00 MiB

```

```

llama_kv_cache_unified: KV self size = 640.00 MiB, K (f16): 320.00 MiB, V
(f16): 320.00 MiB
llama_context: enumerating backends
llama_context: backend_ptrs.size() = 1
llama_context: max_nodes = 65536
llama_context: worst-case: n_tokens = 512, n_seqs = 1, n_outputs = 0
llama_context: reserving graph for n_tokens = 512, n_seqs = 1
llama_context: reserving graph for n_tokens = 1, n_seqs = 1
llama_context: reserving graph for n_tokens = 512, n_seqs = 1
llama_context: CPU compute buffer size = 167.01 MiB
llama_context: graph nodes = 1289
llama_context: graph splits = 1
CPU : SSE3 = 1 | SSSE3 = 1 | AVX = 1 | AVX2 = 1 | F16C = 1 | FMA = 1 | BMI2 = 1
| LLAMAFILE = 1 | OPENMP = 1 | AARCH64_REPACK = 1 |
Model metadata: {'tokenizer.ggml.unknown_token_id': '50256',
'tokenizer.ggml.eos_token_id': '50256', 'tokenizer.ggml.bos_token_id': '50256',
'general.architecture': 'phi2', 'general.name': 'Phi2', 'phi2.context_length':
'2048', 'general.quantization_version': '2', 'tokenizer.ggml.model': 'gpt2',
'tokenizer.ggml.add_bos_token': 'false', 'phi2.embedding_length': '2560',
'phi2.attention.head_count': '32', 'phi2.attention.head_count_kv': '32',
'phi2.feed_forward_length': '10240', 'phi2.attention.layer_norm_epsilon':
'0.000010', 'phi2.block_count': '32', 'phi2.rope.dimension_count': '32',
'general.file_type': '15'}
Using fallback chat format: llama-2

```

```

[ ]: # Exemplo de uso
if __name__ == "__main__":
    pergunta = input("Digite sua pergunta sobre o PDF: ")
    print(pergunta)
    resposta = gerar_resposta(pergunta)
    print("\n Resposta:")
    print(resposta)

```

```

llama_perf_context_print: load time = 88872.72 ms
llama_perf_context_print: prompt eval time = 88864.78 ms / 621 tokens (
143.10 ms per token, 6.99 tokens per second)
llama_perf_context_print: eval time = 503773.71 ms / 199 runs (
2531.53 ms per token, 0.40 tokens per second)
llama_perf_context_print: total time = 593851.78 ms / 820 tokens

```

Resposta:

S.M.S.I. 0 D.M.S.I. 1 S.M.S.I. 2 D.M.S.I. 3 S.M.S.I. 4 D.M.S.I. 5
S.M.S.I. 6 D.M.S.I. 7 S.M.S.I. 8 D.M.S.I. 9 S.M.S.I. 10 D.M.S.I. 11
S.M.S.I. 12 D.M.S.I. 13 S.M.S.I. 14 D.M.S.I. 15 S.M.S.I.

A:

I know this is an old question, but