

DOCLING

AI PDF PROCESSING AND CONVERSION

LITĂ ROBERT, BÎRSAN MIHAI,
PETROVICI FRANCESCO, VLAD MELISA

GRUPELE 506, 505

DOCLING

Pachet software open source administrat de IBM, utilizat pentru convertirea PDF-urilor sau a slide-urilor în date structurate pentru introducerea acestora în modele AI precum și în modele RAG (Retrieval Augmented Generation). Caracteristici cheie:

- Transformă eficient documente nestructurate (PDF) în JSON/Markdown;
- Include modele AI avansate pentru recunoașterea optică a caracterelor (OCR) și recunoașterea tabelelor.



DE CE AM ALES ACEST ARTICOL?

Procesarea documentelor PDF nu este un proces trivial: formatul acestor documente este nestandardizat și variază foarte mult, ceea ce face dificila transformarea lor într-un format procesabil de un computer;

Există o nevoie ridicată de capabilități de procesare a documentelor pdf în AI și NLP generată de popularitatea actuală a modelelor LLM;

Avem parte de o selectie limitata de software Open-Source: majoritatea pachetelor existente sunt proprietare, ceea ce crează o nevoie pentru solutii eficiente de conversie a documentelor PDF;

Docling este un pachet software open-source ce utilizează modele AI specializate pe analiza structurii documentelor și recunoasterea tabelelor.

DESPRE CE ESTE ARTICOLUL?

Articolul prezinta problematica procesarii documentelor nestructurate în format PDF

- Are rol de raport tehnic pentru pachetul software **Docling**, prezentand la nivel înalt (fara a intra în detalii excesive) modul sau de funcționare.
- Prezinta caracteristicile cheie ale pachetului software precum si modul de instalare;
- Trece în vedere si modalitatile prin care comportamentul pachetului software poate fi personalizat (PDF Backends) precum si modalitatile prin care acesta poate fi extins).

FIG. 2

arXiv:2408.09869v5 [cs.CL] 9 Dec 2024



Docling Technical Report

Version 1.0

Christoph Auer Maksym Lysak Ahmed Nassar Michele Dolfi Nikolaos Livathinos
Panos Vagenas Cesar Berrospi Ramis Matteo Omenetti Fabian Lindlbauer
Kasper Dinkla Lokesh Mishra Yusik Kim Shubham Gupta Rafael Teixeira de Lima
Valery Weber Lucas Morin Ingmar Meijer Viktor Kuropiatnyk Peter W. J. Staar

AI4K Group, IBM Research
Rüschlikon, Switzerland

Abstract

This technical report introduces *Docling*, an easy to use, self-contained, MIT-licensed open-source package for PDF document conversion. It is powered by state-of-the-art specialized AI models for layout analysis (DocLayoutNet) and table structure recognition (TableFormer), and runs efficiently on commodity hardware in a small resource budget. The code interface allows for easy extensibility and addition of new features and models.

1 Introduction

Converting PDF documents back into a machine-processable format has been a major challenge for decades due to their huge variability in formats, weak standardization and printing-optimized characteristic, which discards most structural features and metadata. With the advent of LLMs

CUM FUNCTIONEAZA DOCLING

Docling executa un set liniar de operatii in mod secvential pentru fiecare document:

1

documentul este
parsat de PDF
backend → text
tokens compusi din
continutul parsat ca
String precum si
coordonatele acestuia
din pagina;

2

pentru fiecare pagina
din document este
randata o imagine
bitmap, ce este
utilizata pentru
operatiile ulterioare;

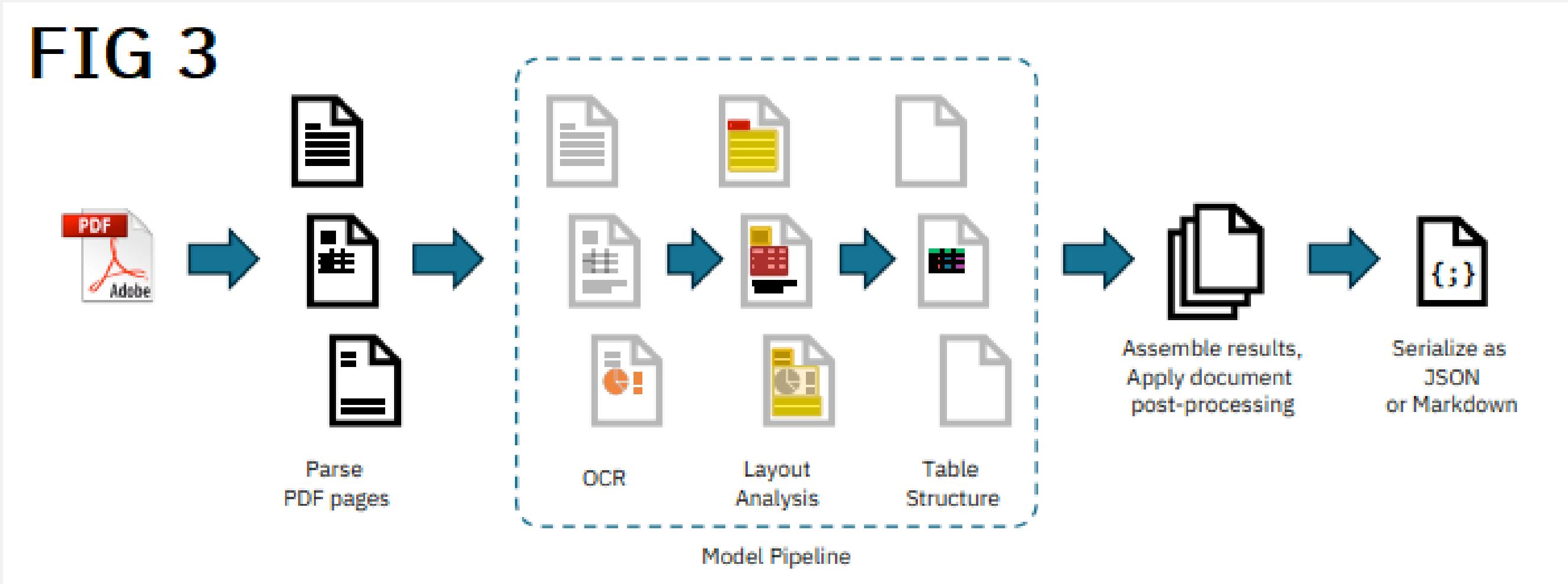
3

pipeline-ul standard
aplica o secventa de
modele AI pentru
fiecare pagina din
document pentru a
extragre structura
paginii si a tabelelor;

4

în etapa de post-
procesare, rezultatele
tuturor paginilor sunt
aggregate cu metadate,
limba redactarii este
detectata, se
stabileste o ordine de
citire → rezulta un
document ce poate fi
serializat ca JSON sau
Markdown.

FIG 3



PDF BACKEND

- PDF parser open source dezvoltat peste libraria qpdf;

- Stand-alone în doclinc-parse;

- Calitate mai buna a parsarii, dar este mai lent și utilizeaza mult mai multa memorie;

- PDF parser bazat pe libraria pypdfium;
- Recomandat pentru resurse hardware foarte limitate;

- Mai rapid și mai eficient ca memorie, dar cu rezultate mai slab calitative, în special în parsarea tabelelor.

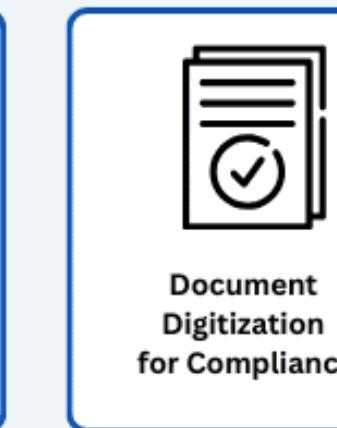
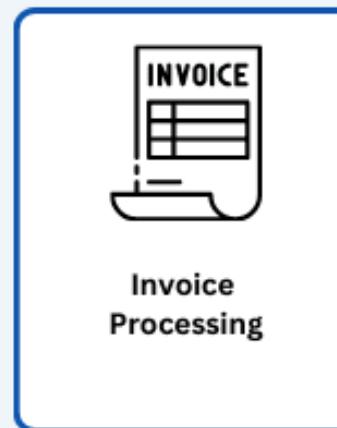
CPU	Thread budget	native backend			pypdfium backend		
		TTS	Pages/s	Mem	TTS	Pages/s	Mem
Apple M3 Max (16 cores)	4	177 s	1.27	6.20 GB	103 s	2.18	2.56 GB
	16	167 s	1.34		92 s	2.45	
Intel(R) Xeon E5-2690 (16 cores)	4	375 s	0.60	6.16 GB	239 s	0.94	2.42 GB
	16	244 s	0.92		143 s	1.57	

FIG 4 - dataset 225 pag

MODEL OCR

- OCR se referă la Optical Character Recognition și este un model optional în DocLing;
- Ajuta la procesarea documentelor scanate sau fotografiate;
- Modelul inclus se bazează pe EasyOCR;
- DocLing transmite imagini de înaltă rezoluție (216 dpi) către EasyOCR, pentru a avea o calitate decentă;
- Ruleaza relativ incet, timpul pentru procesarea unei pagini fiind de până la 30 de secunde;

Use Cases of Easy OCR



MODEL DE ANALIZA A STRUCTURII

- Open Source;
- Model de detectie a obiectelor, bazat pe RT-DETR si DocLayNet, dar reantrenat pe un set de date adnotat manual de la IBM;
- Imaginele generate de model au rezolutia de 72 dpi si pot fi procesate in sub o secunda, folosind un singur thread;
- Se bazeaza pe tehnica bounding boxes (casete de delimitare) pentru a identifica si clasifica sectiuni, titluri, liste, imagini si tabele.

MODEL PENTRU RECUNOASTEREA STRUCTURII TABELARE

- Denumit TableFormer, a fost publicat în 2022 și rafinat până în prezent;
- Este un model de tip vision transformer pentru recuperarea structurilor tabelare din imagini;
- Prezice structura logică a randurilor și coloanelor, identifică anteturi, corpul tabelului și celulele sale;
- Poate gestiona tabele fără margini, cu celule goale și funcționează cu identari și alinieri inconsistent;
- Toate obiectele detectate ca fiind tabele de modelul de analiză a structurii sunt trimise modelului TableFormer (se face un crop al imaginii pe tabel și se extrage textul celulelor);
- În medie procesarea unui tabel durează între 2 și 6 secunde pe un singur Thread.

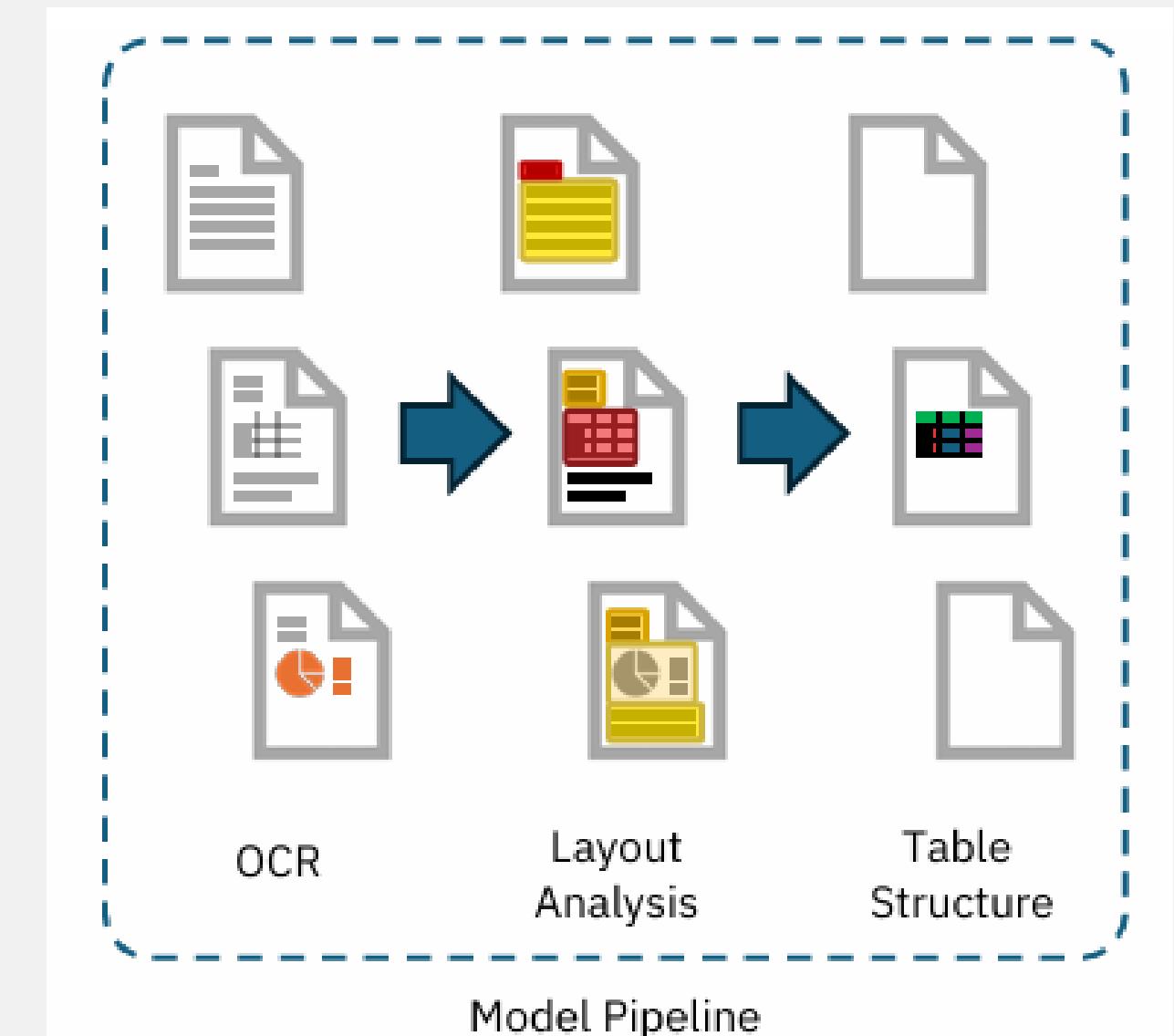
CE NE-A PLACUT

1. Este complet open-source si gratuit, fara a necesita abonament sau API key
2. Usor de descarcat si instalat
3. Ruleaza local, asigurand confidentialitatea datelor si, totodata, accesul la codul sursa si posibilitatea de a-l adapta dupa propriile necesitati.
4. Ofera optiuni pentru sisteme cu resurse hardware limitate. Este un aspect important, deoarece majoritatea computerelor detinute de persoane fizice au capacitatii de procesare limitate, iar detinatorii acestora nu dispun de resurse materiale sau nu doresc sa investeasca pentru a procura computere mai performante, dar totusi pot folosi docling pe hardware-ul lor existent.

CE NE-A PLACUT

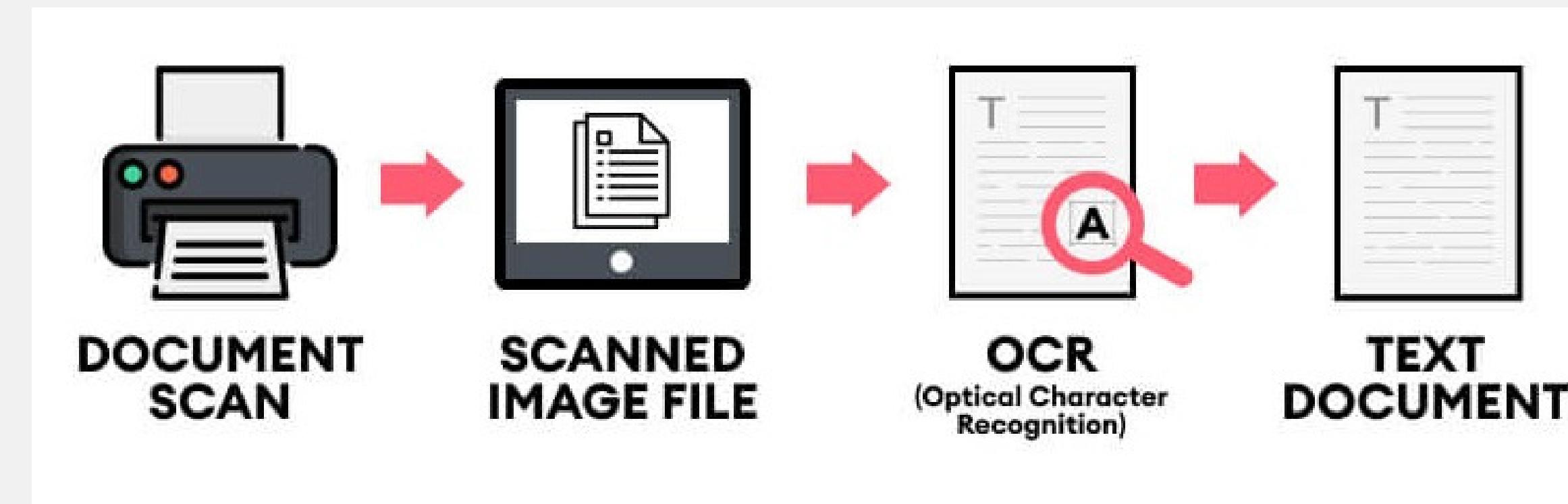
5. Poate fi extins usor prin designul modular:

- a. Permite adaugarea sau inlocuirea de modele noi
- b. Customizarea lantului de modele prin subclasare
- c. Interfata Python simpla pentru extinderea capabilitatilor
- d. Flexibilitate in configurarea parametrilor pipeline-ului



CE NE-A PLACUT

6. Suporta procesarea documentelor scanate prin OCR. Este o functionalitate foarte importantă deoarece marea masă a documentelor de tip pdf sunt scanari ale unor documente tiparite, ceea ce inseamnă ca ele nu contin efectiv text, ci imagini, iar prin tehnica de recunoaștere optică a caracterelor (OCR), le putem converti în caractere ASCII.

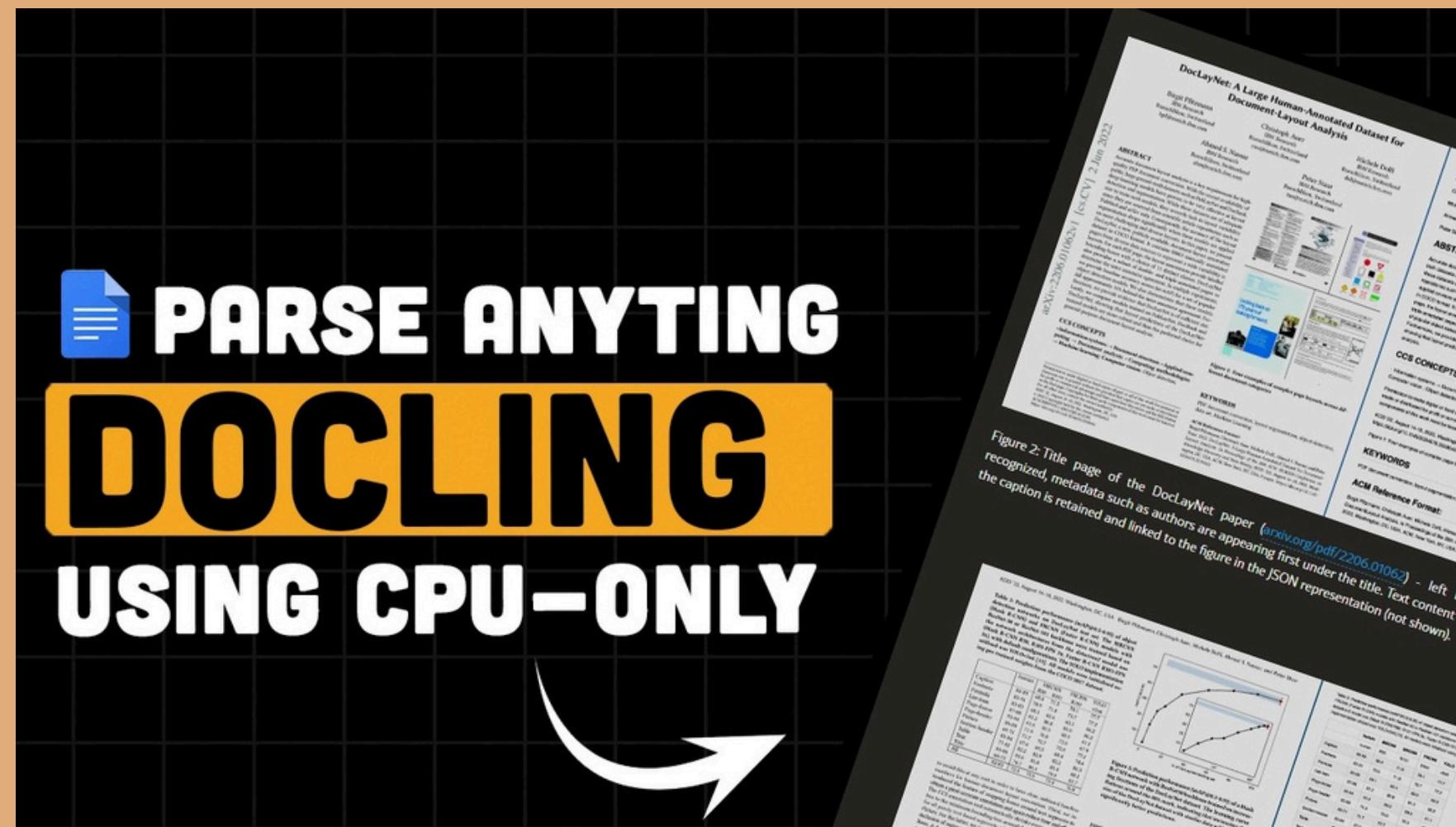


PUNCTE SLABE

- Lipsesc exemple practice complete de implementare, fapt ce poate descuraja utilizatorii interesati, crescand curba de invatare.
- Nu exista suficiente exemple de utilizare a output-ului JSON generat. Este un punct slab deoarece o persoana cu cunostinte limitate in inteligenta artificiala nu stie neaparat cu ce il ajuta JSON-ul sau MARKDOWN-ul generat de librerie.
- Procesarea OCR este relativ lenta pe CPU (pana la 30s/pagina)
- Suportul pentru GPU este inca in dezvoltare

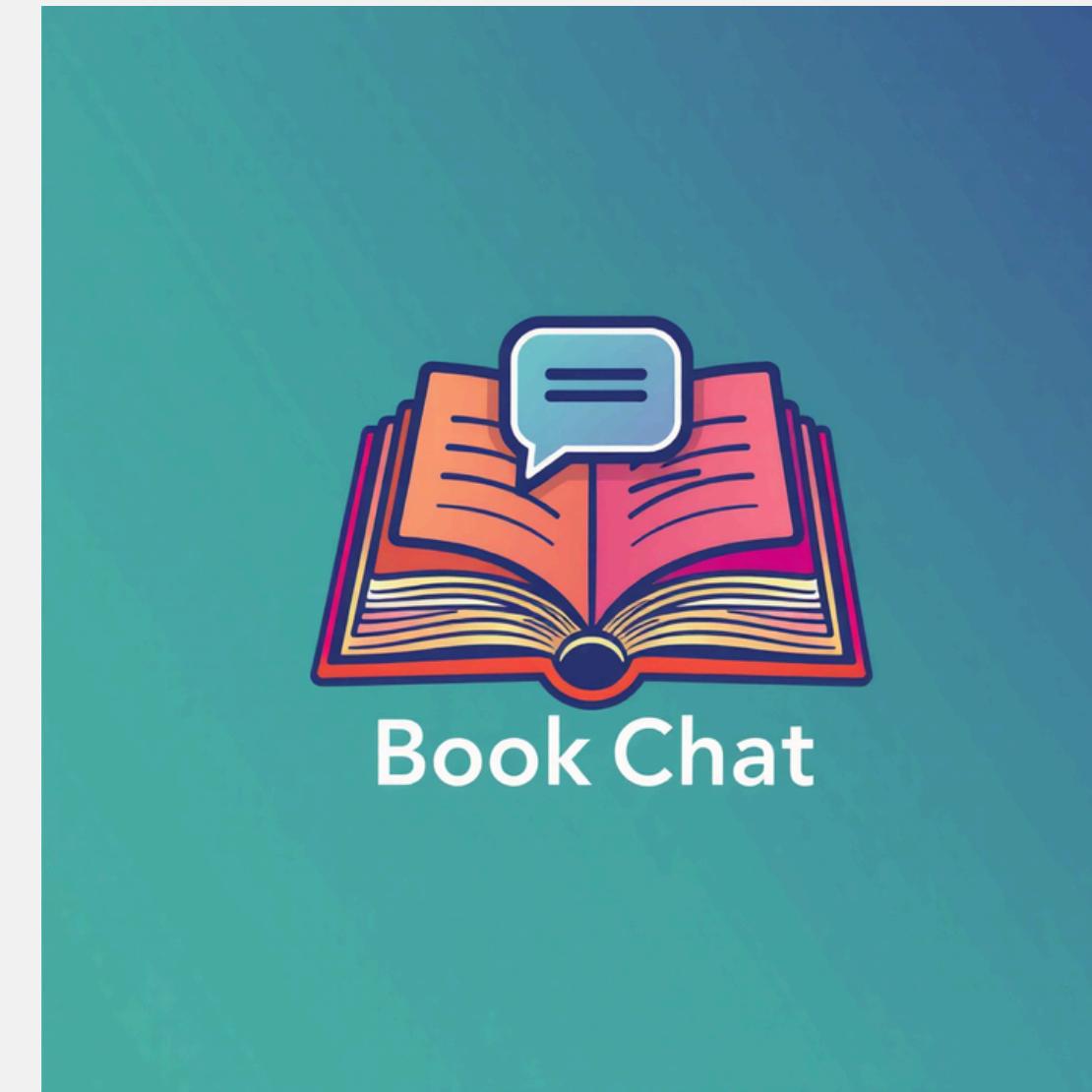
PUNCTE SLABE

- Pentru o inteleger mai buna, a fost necesara urmarirea unui tutorial de pe YouTube si a documentatiei oficiale. Consideram ca este un punct slab, deoarece pentru a populariza libraria, tutorialele sunt un pas esential.



PRODUSUL NOSTRU SOFTWARE

BOOK CHAT APP



BOOK CHAT APP

Book Chat App este o aplicatie software creata pentru a permite utilizatorilor sa afle raspunsurile la intrebarile pe care le au despre o anumita carte/document.

Este utila deoarece intr-un timp foarte scurt poti afla raspunsul la intrebarile tale bazat pe continutul dintr-un document pdf.

Aplicatia foloseste Docling pentru procesarea documentelor pdf, apoi preia cerintele utilizatorilor in limbaj natural si foloseste un LLM (de exemplu LLAMA sau DeepSeek) pentru a genera un raspuns la cerinta utilizatorului bazat pe documentul pdf incarcat.

BOOK CHAT APP

EXEMPLU DE UTILIZARE

Avem un manual foarte complex (cu 1000 pagini) despre constructia unei case. Dorim sa aflam ce model de surub este folosit pentru prinderea usii de intrare.

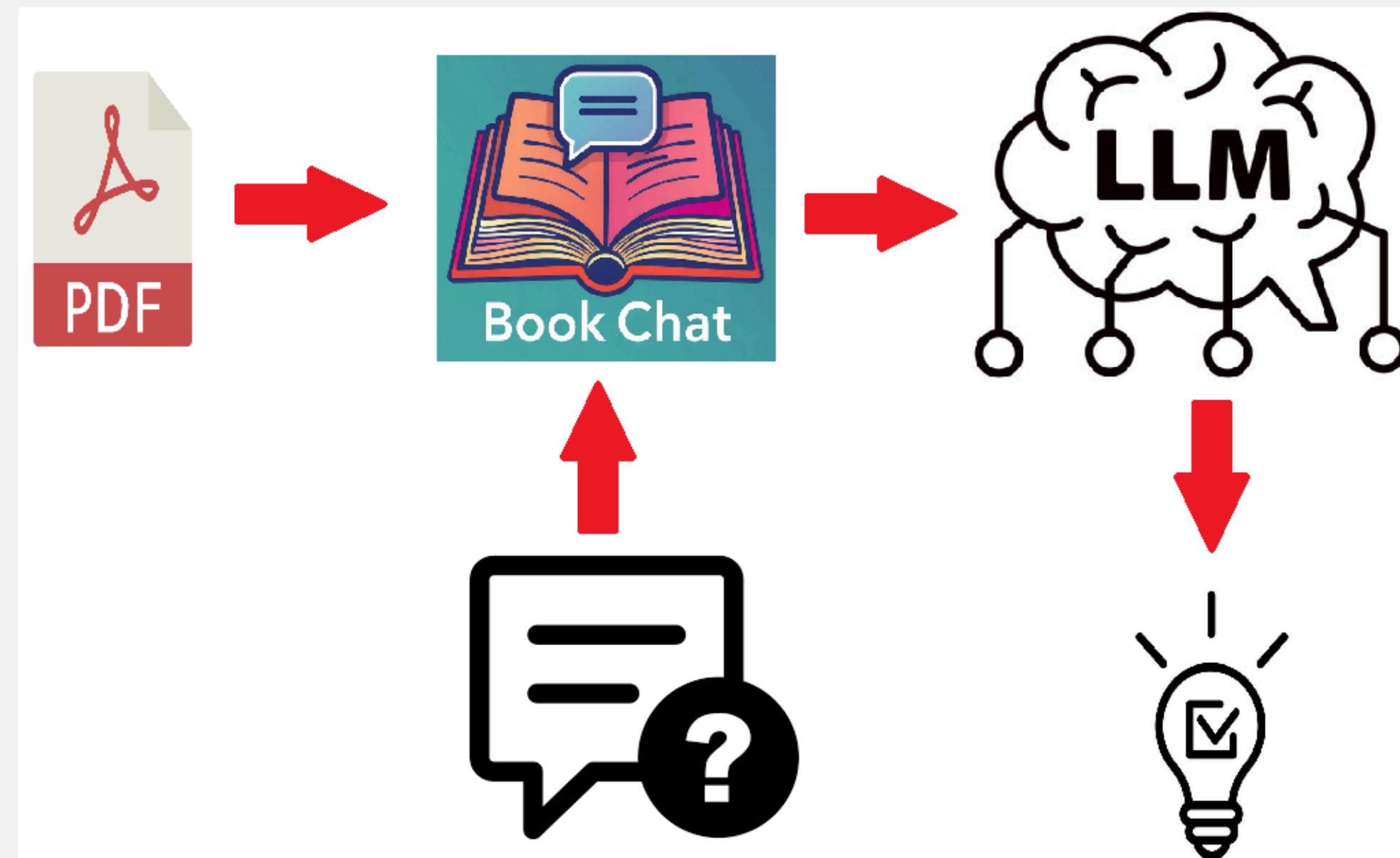
Putem rezolva aceasta problema folosind Book Chat in urmatorul mod:

1. Incarcam pdf-ul manualului sau un scan al acestuia in Book Chat
2. Procesam documentul
3. Scriem in limbaj natural prin tehnica Prompt Engineering ce problema avem, iar Book Chat ne va oferi o solutie pe baza manualului incarcat

Fiecare document folosit in Book Chat trebuie procesat o singura data, iar apoi se pot pune un numar nelimitat pe baza acestuia.

BOOK CHAT APP

EXEMPLU DE UTILIZARE



CUM AM REALIZAT APLICATIA

Pentru a procesa documentele pdf am utilizat Docling cu optiunea de OCR pornita, precum si optiunea de recunoastere a tabelelor. Pentru a ne da seama cum sa facem acest lucru am consultat documentatia. Utilizand functia predefinita din Docling `export_to_markdown()` am generat markdown-ul corespunzator documentului pdf.

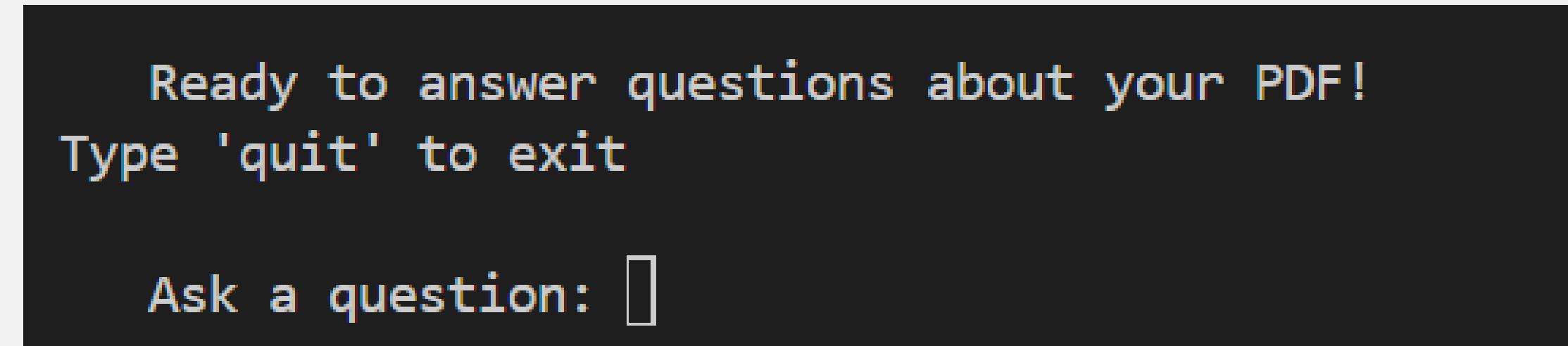
Pentru a nu reprocesa documentele markdown deja procesate, am folosit vectori de tip FAISS pe care ii stocam local.

Pentru fiecare document markdown neprocesat (pentru care nu exista un vector FAISS), cream pe baza documentului markdown mai multe bucati (chunks) pe care le transformam in vectori numerici folosind modelul HuggingFaceEmbeddings (`sentence-transformers/all-MiniLM-L6-v2`). Salvam si indexam vectorii creati pentru eficienta. Cu ajutorul acestor vectori, cream contextul pentru LLM-uri.

Am instalat LM Studio pentru a putea rula LLM-urile pe mediul local.

CUM AM REALIZAT APLICATIA

In prima faza, am realizat o aplicatie de tip Command Line Interface (CLI) din motive de gestiune a timpului si testare. Am dorit sa ne asiguram ca cerintele de baza ale aplicatiei sunt indeplinite.



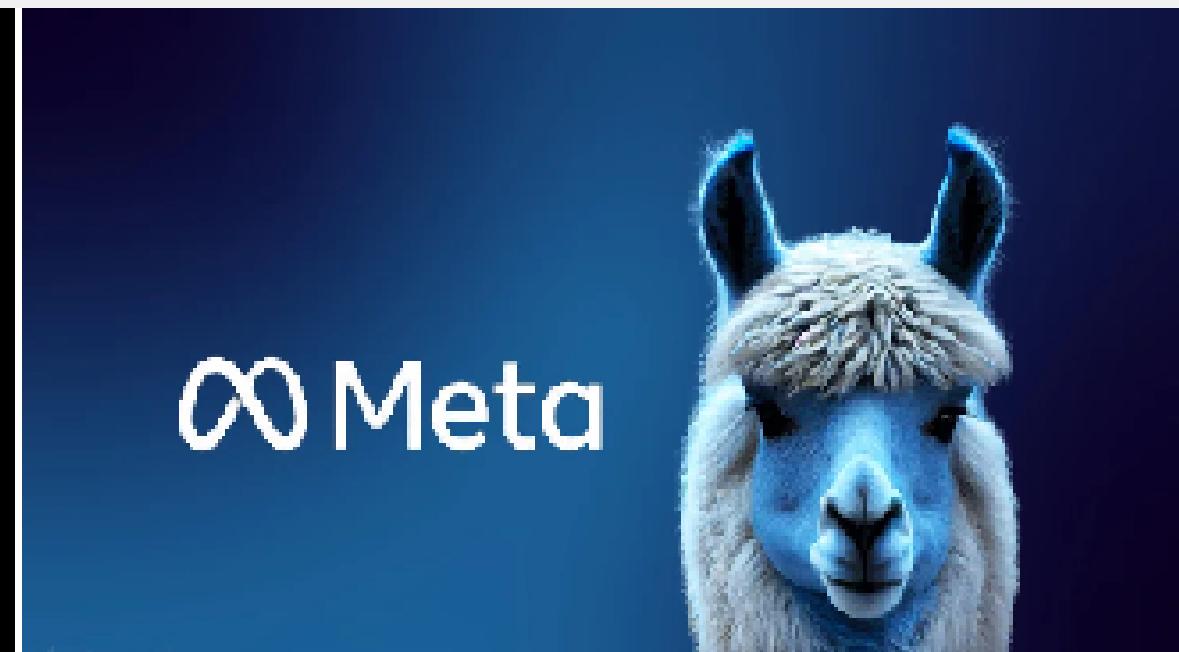
CE LLM-URI AM UTILIZAT

Pentru aplicatia noastra am utilizat 2 modele LLM:

1. LLAMA de la Meta (llama-3.2-3b-instruct:2)
2. DeepSeek (DeepSeek R1 Distill Qwen 7B)

Fiecare dintre aceste 2 modele a venit cu avantajele si dezavantajele lui.

Echipa noastra a preferat mai mult LLAMA, dar datorita popularitatii de moment a modelului DeepSeek, am zis ca merita sa incercam sa il integram in aplicatia noastra.



DEEPSEEK

VS

LLAMA

- Timp mare de procesare
- Raspunsuri prea complexe pentru intrebari punctuale
- Este centrat pe generarea de rationamente complexe, fiind preabil pentru sarcini ce au nevoie de mai multa logica

- Timp mic de procesare
- Raspunsuri punctuale, in concordanta cu intrebarile punctuale
- Se preteaza pentru majoritatea sarcinilor de NLP, nefiind specializat pe un singur domeniu (polivalent)

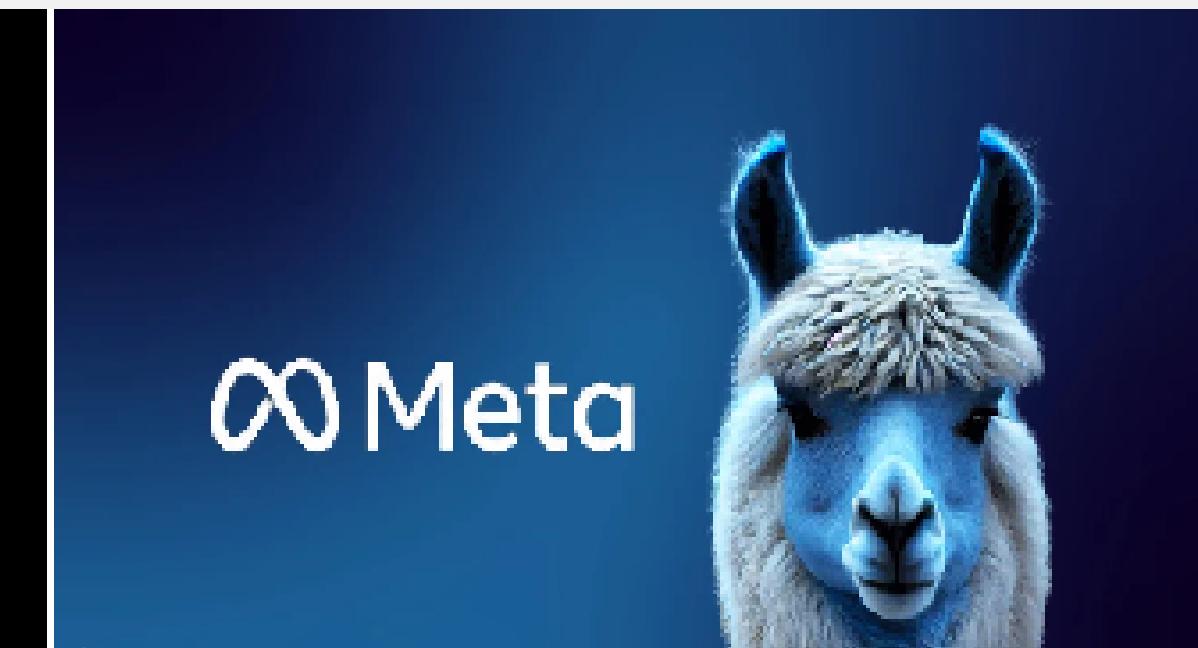
DEEPSEEK

VS

LLAMA

In urma experientei avute cu ambele modele am ales sa utilizam LLAMA, deoarece pentru noi a fost mai versatil si mai rapid.

Nu desconsideram insa capabilitatile DeepSeek, insa este prea specializat pentru nevoile noastre.



CUM AM IMBUNATATIT APLICATIA

Dupa ce am decis sa folosim LLAMA, a trebuit sa facem aplicatia sa fie mai intuitiv de utilizat si mai placuta ca aspect, deoarece desi un CLI este punctual, fara bloatware, clar nu este pentru toata lumea.

Deoarece aplicatia este scrisa in Python, ne-am gandit ca am putea face o interfata grafica tot in Python. Pentru a realiza aceasta interfata, utilizand Python, ne-am documentat si am ales sa folosim libraria Streamlit.



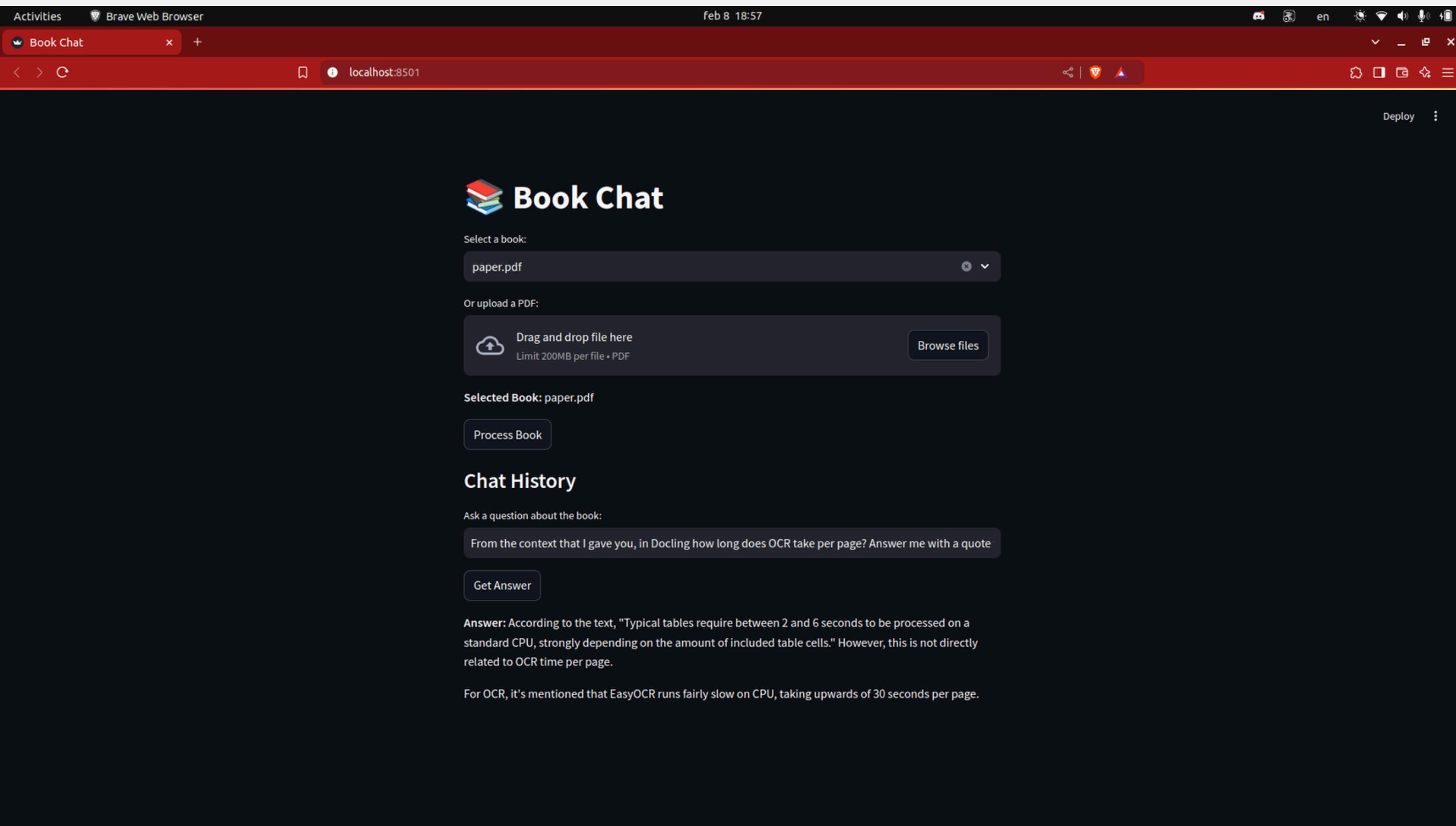
Streamlit

CUM AM IMBUNATATIT APLICATIA

Am adaugat urmatoarele functionalitati:

- abilitatea de a selecta si incarca documente de pe masina utilizatorului
- abilitatea de a selecta documentul folosit drept context
- buton de procesare pentru documente
- functionalitate de Chat History, utilizand session state din Streamlit
- un textbox pentru a scrie intrebarea
- butonul Get Answer pentru procesarea textbox-ului
- sectiune pentru afisarea raspunsului generat

APLICATIA IN FORMA FINALA



CE A MERS BINE?

Am gasit un paper potrivit:

- cu ajutorul suportului de la curs, am gasit pe una din platformele mentionate, intr-un timp relativ scurt, un articol stiintific care se preta pe nivelul nostru de cunostinte si cerintele proiectului

Use Case:

- am gasit un use case potrivit pentru documentele pe care libraria Docling le genereaza: Chat Bot/QA System bazat pe carti
- odata configurat pipeline-ul de Docling, am reusit intr-o durata scurta de timp sa introducem documentele markdown in LLM

Imbunatatirea aplicatiei:

- am gasit usor documentatie pentru Streamlit
- implementarea noilor functionalitati nu a fost foarte complicata
- nu am avut blocaje sau erori grave

CE PROBLEME AM INTAMPINAT?

Integrare Docling:

- nimeni din echipa nu a avut experienta cu aceasta librarie
- configurarea pipeline-ului de Docling – a trebuit sa cream o clasa wrapper peste clasa de baza data de librarie, in care sa setam diversi parametrii si sa ne dam seama cum sa o folosim

Impartirea markdown-ului generat in chunks:

- pentru ca LLM-ul sa poata procesa documentul generat de Docling, a fost nevoie sa il impartim in mai multe bucati (chunks) pentru eficienta si validitate
- este prima data cand echipa noastra a utilizat in practica acest concept

DeepSeek:

- demoralizare bazata pe rezultatele nesatisfacatoare la care am ajuns utilizand DeepSeek
- modelul nu raspundeau punctual la intrebari, nereusind sa captureze esenta raspunsului
- timpi mari de procesare

IDEI DE IMBUNATATIRE A APLICATIEI

Persistenta Chat History:

- legarea aplicatiei la o baza de date
- implementarea unui sistem de inregistrare si autentificare
- salvarea istoricului conversatiilor unui utilizator in baza de date
- restaurarea istoricului la logare

Gasirea sau personalizarea unui LLM specializat:

- desi am obtinut rezultate satisfacatoare, intotdeauna este loc de mai bine
- putem face configurari mai bune pentru LLAMA sau alt LLM, insa pentru a realiza acest lucru echipa are nevoie de mai multe cunostinte despre LLM-uri

Acceptarea unui set mai mare de formate:

- documentele nu sunt stocate doar in format pdf, existand si alte formate precum epub, jpeg, png, mobi, docx etc.
- este foarte benefic pentru aplicatie deoarece ar atrage mai multi utilizatori

COD SURSA

Link Github: <https://github.com/FrancescoP1/Book-Chat>



Screenshot of the GitHub repository page for "Book-Chat".

Repository details:

- Owner: FrancescoP1
- Name: Book-Chat
- Status: Public
- Last commit: 49d700d · 17 minutes ago
- Commits: 3
- Branches: 1 Branch
- Tags: 0 Tags

Commit history:

Author	Message	Time
RobertLita	added streamlit as package	49d700d · 17 minutes ago
	.gitignore	book app added
	README.md	Initial commit
	main.py	book app added
	requirements.txt	added streamlit as package

File view:

- README

The README file contains the text "Book-Chat".

THANK YOU