Academic Citation Extractor using LLMs and Streamlit – Final Project Report

Student Introduction:

My name is **Harman Gill**, and I am a B.Tech graduate in **Computer Science and Engineering**. I am passionate about **Artificial Intelligence**, especially in **Natural Language Processing (NLP)** and **Generative AI**. This project is a demonstration of applying core AI concepts like **RAG (Retrieval-Augmented Generation)**, **LLMs**, and **PDF processing** to build a complete working solution.

Project Overview:

This project, *Academic Citation Extractor*, is a Streamlit-based web application that automates the extraction of academic citations from research PDFs. It leverages a Retrieval-Augmented Generation (RAG) pipeline, combining document chunking, semantic search (using FAISS and sentence-transformers), and language models (LLMs) to identify citation details like author, title, journal, year, and DOI. The extracted citations are then verified using the CrossRef API to check their authenticity. The app displays both extracted and verified citations and generates a downloadable report. This tool simplifies manual citation management and ensures accurate referencing in academic writing. It is modular, lightweight, and optimized for local environments.

Technologies Used:

Component	Tool/Library
UI Framework	Streamlit
PDF Text Extraction	PyMuPDF (fitz)
Embedding Model	sentence-transformers/all-MiniLM-L6-v2
LLM	sshleifer/tiny-gpt2 (lightweight test model)
RAG Pipeline	LangChain + FAISS
Citation Verification	CrossRef API
JSON Parsing	Python ast & json

Project Structure:

```
academic_citation_extractor/

|

├── app.py              # Streamlit app main entry

├── requirements.txt       # All dependencies

├── src/

|   ├── model.py          # Model loading and RAG pipeline

|   ├── prompt.py         # PromptTemplate definition

|   ├── utils.py          # PDF parsing & citation verification


|   ├── report.py         # Final report generator
```

Project Workflow:

1. **PDF Upload**: User uploads a research paper PDF.
2. **Text Chunking**: Extract text using PyMuPDF, chunk it.
3. **Embedding**: Use Sentence-BERT to vectorize chunks.
4. **FAISS Retrieval**: Store and retrieve most relevant text.
5. **LLM Processing**: Use PromptTemplate + tiny GPT2 model to extract citations.
6. **Verification**: Send each extracted title to the CrossRef API for validation.
7. **Report Generation**: Display results and allow report download in .txt format.

Challenges Faced:

Problem	**Solution**
gpt2 model caused index errors during decoding	Switched to sshleifer/tiny-gpt2 which supports correct padding
LangChain breaking changes (Document import, vectorstores)	Migrated all imports to langchain_community and langchain_core
Sentence-transformer error due to HuggingFace version	Downgraded huggingface_hub to compatible version
Prompt variable mismatch (input vs context)	Fixed by explicitly using context as input variable
Unexpected characters causing JSON parsing errors	Filtered PDF content and handled malformed Unicode gracefully

Project Outcome:

- Successfully extracts citation data into **JSON**
- Verifies citations via **CrossRef**
- Fully functional **Streamlit web interface**
- Modular and clean code, supports future LLM swapping
- Can run **locally without GPU** using a small language model

Future Enhancements:

1. Upgrade to LLaMA 2 or Mistral 7B **for better generation accuracy.**
2. Enable **.csv** or **.bib** report downloads **for researchers.**
3. Deploy on Hugging Face Spaces / Render **for public access.**
4. Add Plagiarism Detection **using semantic similarity.**
5. Fine-tune on academic citation datasets **for domain adaptation.**
6. Highlight citations visually inside PDFs **with bounding boxes**.

Sample Output:

Extracted JSON:

```
json
CopyEdit
[
 {
   "Author": "Vaswani et al.",
   "Title": "Attention is all you need",
   "Year": "2017",
   "Journal": "NeurIPS",
   "DOI": null
 },
 ...]
```

Conclusion:

The *Academic Citation Extractor* successfully demonstrates the practical integration of Generative AI with information retrieval for academic document processing. By combining a lightweight language model, vector-based retrieval (FAISS), and citation verification (CrossRef), the system automates a typically manual and error-prone task. This project has enhanced my understanding of RAG pipelines, prompt engineering, and real-world deployment of LLMs. It serves as a strong foundation for building more advanced research assistance tools in the future.

References:

1. Vaswani, A., et al. (2017). *Attention is all you need*. NeurIPS.
2. Devlin, J., et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers*. NAACL.
3. Reimers, N., & Gurevych, I. (2019). *Sentence-BERT*. EMNLP.
4. Touvron, H., et al. (2023). *LLaMA: Open and Efficient Foundation Models*. arXiv.
5. LangChain Docs: https://docs.langchain.com
6. CrossRef API: https://www.crossref.org
7. Hugging Face Transformers: https://huggingface.co/transformers