🤖 Mini RAG Chatbot

A Retrieval-Augmented Generation (RAG) chatbot built with Llama 3.2, FAISS, and LangChain. Ask questions about your research papers and get accurate, source-backed answers.

✨ Features

- **Local LLM**: Uses Llama 3.2 via Ollama (no API costs!)
- **Semantic Search**: FAISS vector store with sentence-transformers embeddings
- **Source Citations**: Every answer includes relevant document sources
- **Interactive UI**: Streamlit web interface + CLI mode
- **Scalable**: Handles 100-1,000+ documents efficiently

🏗️ Architecture

```
Query → Embeddings → FAISS Search → Top-K Chunks → Llama 3.2 → Answer + Sources
```

📋 Prerequisites

1. **Python 3.9+**
2. **Ollama** - Install from ollama.ai

```bash
# Install Ollama, then pull Llama 3.2
ollama pull llama3.2
```

🚀 Quick Start

1. Installation

```bash
# Clone or create project directory
mkdir mini-rag-chatbot && cd mini-rag-chatbot

# Install dependencies
pip install -r requirements.txt
```

2. Add Your Documents

Place PDF research papers in the `data/` directory:

```bash
mkdir data
# Copy your PDFs here
cp /path/to/papers/*.pdf data/
```

3. Ingest Documents

```bash
python src/ingest.py
```

This will:

- Load all PDFs from `data/`
- Split into chunks (1000 chars, 200 overlap)
- Generate embeddings
- Create FAISS vector store in `vectorstore/`

Expected output:

```
Loading documents from data...
Loaded 150 document pages
Chunking documents...
Created 423 chunks
Creating embeddings and vector store...
✓ Vector store saved with 423 chunks
```

4. Run the Chatbot

Option A: Streamlit UI (Recommended)

```bash
streamlit run src/app.py
```

Option B: CLI Interactive Mode

```bash
python src/chatbot.py
```

Option C: Single Query

```bash
python src/chatbot.py --query "What are the main findings about transformers?"
```

📁 Project Structure

```
mini-rag-chatbot/
├── data/                   # Your PDF documents
│   └── *.pdf
├── src/
│   ├── ingest.py           # Document ingestion pipeline
│   ├── retrieval.py        # FAISS search logic
│   ├── chatbot.py          # RAG Q&A with Llama 3.2
│   └── app.py              # Streamlit web interface
├── vectorstore/            # FAISS index (generated)
│   ├── index.faiss
│   ├── index.pkl
│   └── metadata.pkl
├── notebooks/
│   └── demo.ipynb          # Interactive demo notebook
├── requirements.txt
└── README.md
```

🔧 Configuration

Ingestion Parameters

```bash
python src/ingest.py \
  --data-dir data \
  --vectorstore-dir vectorstore \
  --chunk-size 1000 \
  --chunk-overlap 200
```

Chatbot Parameters

```bash
python src/chatbot.py \
  --model llama3.2 \
  --top-k 4 \
  --temperature 0.1
```

🧪 Testing Retrieval

Test the retrieval system independently:

```bash
python src/retrieval.py "What is attention mechanism?"
```

Output:

```
[Result 1] (Score: 0.7234)
Source: data/attention_paper.pdf
Content: The attention mechanism allows the model to focus on...
```

🎯 Example Queries

Once running, try these questions:

- "What are the main contributions of this paper?"
- "Explain the methodology used in the experiments"
- "What datasets were used for evaluation?"
- "What are the limitations mentioned?"
- "How does this approach compare to previous work?"

🐛 Failure Cases & Fixes

1. Hallucination - Answering Beyond Context

Problem: Model makes up information not in the documents.

Example:

```
Q: "What is the model's accuracy on ImageNet?"
A: "The model achieves 95% accuracy on ImageNet."
(But this wasn't in the documents)
```

Fix: Improved prompt engineering in `chatbot.py`:

```python
prompt = """Answer based ONLY on the context provided.
If the information is not in the context, say:
"I cannot find this information in the provided documents."

Context: {context}
Question: {query}
"""
```

Result: Model now admits when it doesn't know.

---

2. Poor Retrieval - Irrelevant Chunks

Problem: Retrieved chunks don't actually answer the question.

Example:

```
Q: "What loss function was used?"
Retrieved: [Introduction paragraph about deep learning history]
```

Fix: Tuned chunking parameters in `ingest.py`:

```python
# Before: chunk_size=500, overlap=50 (too small, breaks context)
# After: chunk_size=1000, overlap=200 (better context preservation)
```

Also increased `top_k` from 3 to 4 for better coverage.

Result: More relevant chunks with full context.

---

3. Vector Store Not Found Error

Problem:

```
FileNotFoundError: Vector store not found at vectorstore/
```

Fix:

```bash
# Always run ingestion first!
python src/ingest.py

# Verify it was created
ls -la vectorstore/
```

📊 Performance Metrics

Tested on 100 research papers (~500 pages):

Metric	Value
Ingestion Time	~3 minutes
Vector Store Size	~45 MB
Retrieval Time	~0.5s
Generation Time	~3-5s
Accuracy (manual eval)	~85%

🎓 Demo Notebook

Included Jupyter notebook (`notebooks/demo.ipynb`) demonstrates:

1. **Document Loading** - Load and inspect PDFs
2. **Chunking Strategy** - Visualize chunk sizes
3. **Embedding Space** - 2D visualization with t-SNE
4. **Retrieval Testing** - Try different queries
5. **End-to-End RAG** - Complete pipeline walkthrough
6. **Failure Analysis** - Common errors and fixes

Run it:

```bash
jupyter notebook notebooks/demo.ipynb
```

🔍 How It Works

1. Document Ingestion

```python
PDFs → PyPDF → Text Chunks → Sentence-Transformers → Embeddings → FAISS Index
```

2. Query Processing

```python
User Query → Embedding → FAISS Search → Top-K Chunks → Context
```

3. Answer Generation

```python
Context + Query → Prompt → Llama 3.2 → Answer → Sources
```

🛠️ Troubleshooting

Ollama Connection Error

```bash
# Check if Ollama is running
ollama list

# Start Ollama (if needed)
ollama serve

# Pull the model
ollama pull llama3.2
```

Out of Memory

```bash
```

```
# Use smaller chunk size
python src/ingest.py --chunk-size 500

# Or reduce top-k
python src/chatbot.py --top-k 2
```

Slow Generation

```bash
# Use faster model
python src/chatbot.py --model llama3.2  # 3B params, faster

# Reduce max tokens
# Edit chatbot.py: num_predict=256 instead of 512
```

🚀 Extensions

Add Web Search

Combine RAG with real-time web search for current info.

Multi-Modal

Support images from PDFs using CLIP embeddings.

Fine-Tuning

Fine-tune Llama 3.2 on domain-specific papers.

Re-Ranking

Add cross-encoder re-ranking for better retrieval.

📚 Resources

- LangChain Docs
- FAISS
- Ollama
- Sentence Transformers

📝 License

🤝 Contributing

Contributions welcome! Areas for improvement:

- Better chunk boundaries (respect paragraphs/sections)
- Hybrid search (keyword + semantic)
- Query expansion
- Response streaming
- Multi-language support

📧 Contact

Questions? Open an issue or reach out!

Built with ❤️ using Llama 3.2, FAISS, and LangChain