

# Conversational Chatbot (NLP)

## Advanced NLP System for Academic Document Retrieval

**Students:** Tayma Mokrani , Ilies Mraihi

**Email:** tayma.mokrani@fsb.ucar.tn ,  
ilies.mraihi@fsb.ucar.tn

**Course:** CI3 - MLOPS

**Instructor:** Besma Guesmi

**FSB - Computer Science Department**

**January 10, 2025**

GitHub Repository: [https://github.com/TaymaMokrani/Conversational\\_Chatbot\\_MLOPS](https://github.com/TaymaMokrani/Conversational_Chatbot_MLOPS)

# Contents

---

<b>1</b>	<b>Summary</b>	<b>1</b>
1.1	Key Achievements . . . . .	1
<b>2</b>	<b>System Architecture</b>	<b>1</b>
2.1	Directory Structure . . . . .	2
2.2	Technology Stack . . . . .	2
2.3	Data Flow . . . . .	3
<b>3</b>	<b>Implementation Details</b>	<b>3</b>
3.1	Document Ingestion Pipeline . . . . .	3
3.1.1	PDF Processing . . . . .	3
3.2	Intelligent Chunking Strategy . . . . .	4
3.3	Embedding Generation . . . . .	4
3.4	Hybrid Retrieval System . . . . .	5
3.5	Smart Caching Mechanism . . . . .	5
3.6	Architectural Decisions Summary . . . . .	6
3.7	Configuration Options . . . . .	6
<b>4</b>	<b>MLOps-Oriented Configuration Management</b>	<b>7</b>
4.1	Model Configuration Versioning . . . . .	7
4.2	Pipeline Parameter Versioning . . . . .	8
4.3	Implementation Constraints and Limitations . . . . .	8
<b>5</b>	<b>Common Failure Modes</b>	<b>9</b>

# 1 Summary

## Work overview

This project presents an AI-powered chatbot that helps users easily find information from institutional documents using natural language.

## 1.1 Key Achievements

- ✓ **Document Processing:** Ingestion and preprocessing of various PDF files
- ✓ **Intelligent Chunking:** Text segmentation that preserves context
- ✓ **Hybrid Retrieval:** Integration of BM25 and vector search with RRF
- ✓ **Chat Interface:** Web interface with conversation history
- ✓ **Performance Optimization:** Smart caching that reduces processing time
- ✓ **Deployment Ready:** Dockerized application

# 2 System Architecture

## Architecture Highlights

The system is designed like a microservices architecture, with separate components that can be developed and modified independently, while still working together .

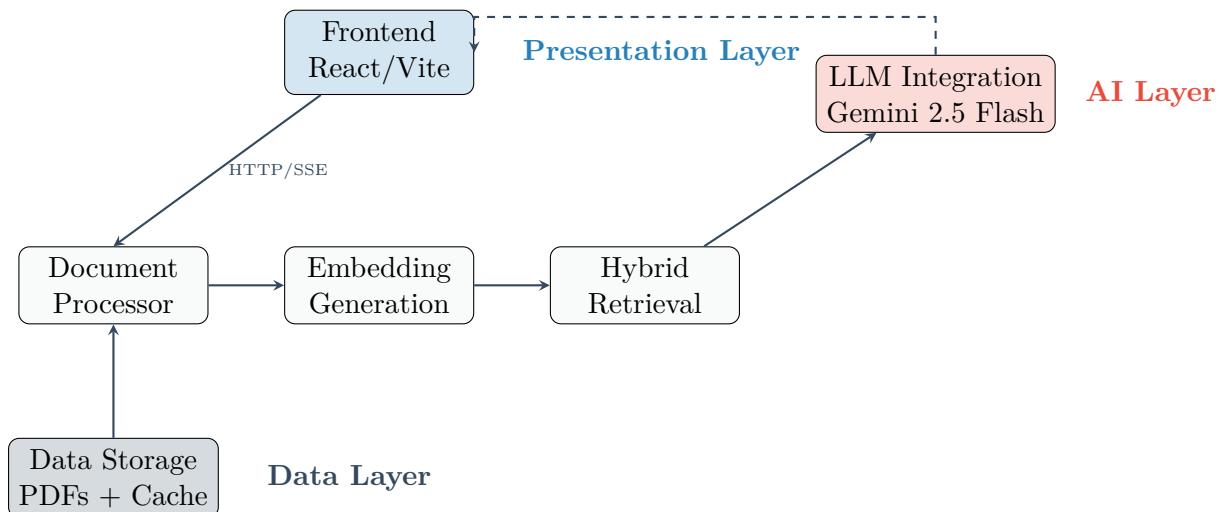


Figure 1: System architecture showing component interactions and data flow

## 2.1 Directory Structure

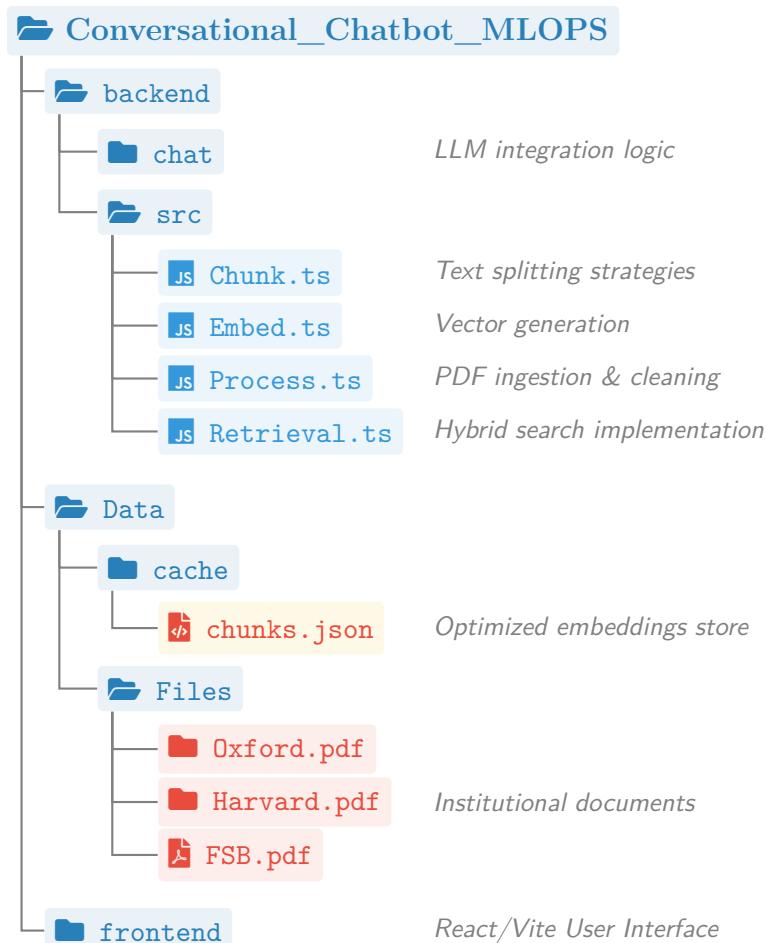


Figure 2: Visual representation of the project directory structure

## 2.2 Technology Stack

Component	Technology	Purpose
<b>Backend Runtime</b>	Bun 1.3.5	High-performance JS runtime
<b>Frontend</b>	React 19 + Vite	UI development
<b>LLM</b>	Google Gemini 2.5	Advanced reasoning
<b>Embeddings</b>	all-MiniLM-L6-v2	semantic vectors
<b>Vector Ops</b>	Custom implementation	Cosine similarity
<b>Text Processing</b>	pdf-parse	PDF extraction
<b>API Framework</b>	Elysia.js	Fast API server
<b>Deployment</b>	Docker	Containerization

Table 1: Technology stack used in the project

## 2.3 Data Flow

The system handles user queries through this pipeline:

1. **Query Reception:** User input submitted via the chat interface
2. **Embedding Generation:** Conversion to semantic vector representations
3. **Hybrid Retrieval:** Parallel BM25 and vector-based search
4. **Result Fusion:** Combining rankings using RRF
5. **Context Assembly:** Selecting top-k results
6. **LLM Synthesis:** Generating a natural language response
7. **Response Delivery:** Returning the formatted answer to the user

## 3 Implementation Details

### 3.1 Document Ingestion Pipeline

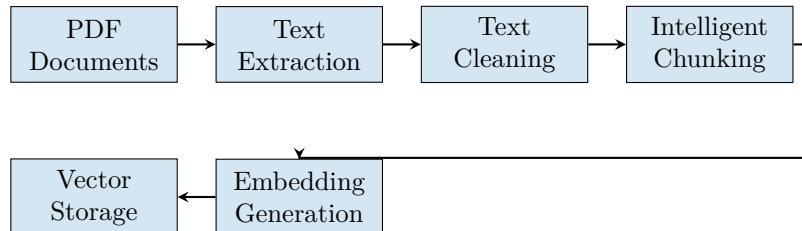


Figure 3: Document processing pipeline

#### 3.1.1 PDF Processing

The process includes:

- **File Ingestion:** PDFs are loaded and converted into a buffer for parsing.
- **Text Extraction:** Raw text is extracted using a PDF parser.
- **Text Cleaning:**
  - Removal of URLs.
  - Elimination of control and non-printable characters.
  - Normalization of whitespace to standardize spacing.
  - Preservation of paragraph breaks.

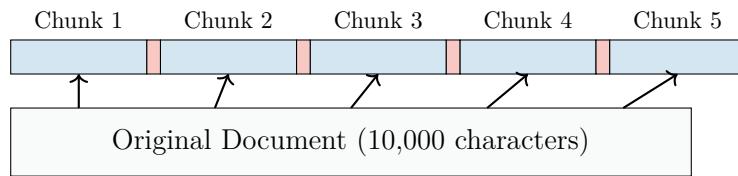
This pipeline ensures that text is clean, structured, and ready for tasks such as chunking, embedding generation, and retrieval.

### 3.2 Intelligent Chunking Strategy

The system implements an adaptive text chunking mechanism to efficiently segment documents :

- **Adaptive Segmentation:** Text is split into chunks based on sizes and overlaps.
- **Language:** Markdown-specific splitting that preserves structure.
- **Embedding:** Each chunk is converted into a vector and assigned a unique ID.
- **Metadata Tracking:** Optional source information is attached to each chunk.

The following figure demonstrates the chunking and embedding process:



*Chunking strategy with 1000-character chunks and 200-character overlap*

Figure 4: Visualization of the chunking strategy showing overlaps

### 3.3 Embedding Generation

- **Pre-trained Model:** Use the MiniLM model ('all-MiniLM-L6-v2') for efficient and accurate sentence embeddings.
- **Feature Extraction:** transformer-based pipeline to generate embeddings from input text.
- **Pooling and Normalization:** Applies mean pooling and normalization .
- **Chunking:** Each text chunk produced in the previous step is transformed into a fixed-size embedding .

The following TypeScript implementation demonstrates the embedding generation:

```

1 import { pipeline } from "@xenova/transformers";
2
3 const Embedder = await pipeline(
4   "feature-extraction",
5   "Xenova/all-MiniLM-L6-v2"
6 );
7
8 export const Embed = async (value: string) => {
9   const output = await Embedder(value, {
10     pooling: "mean",
11     normalize: true
12   });
13   return Array.from(output.data as Float32Array);
14 };

```

Listing 1: Embedding Generation with MiniLM (Embed.ts)

### 3.4 Hybrid Retrieval System

To maximize retrieval accuracy, the system combines traditional and vector-based search using a hybrid approach :

- **Parallel Retrieval:** Queries are executed simultaneously on a BM25 index and a vector store.
- **Reciprocal Rank Fusion (RRF):** Results from both methods are combined using RRF to produce a single ranked list.
- **Top-K Selection:** The system returns the top-k documents along with their relevance scores.

### 3.5 Smart Caching Mechanism

To improve performance the system implements an intelligent caching mechanism :

- **File State Tracking:** The system generates a hash based on file size and modification time for all PDF documents in the directory.
- **Change Detection:** By comparing the generated hash with previous runs, the system can detect updated or new files.
- **Cache Reuse:** Only new or modified files are reprocessed, while unchanged data is retrieved from the cache, significantly reducing processing time.

```

1 function generateFileHash(filesDir: string, pdfFiles: string[]): string {
2   const fileInfos = pdfFiles.map(file => {
3     const filePath = path.join(filesDir, file);
4     const stats = fs.statSync(filePath);
5     return `${file}:${stats.size}:${stats.mtimeMs}`;
6   }).sort().join(" | ");
7
8   let hash = 0;
9   for (let i = 0; i < fileInfos.length; i++) {
10     const char = fileInfos.charCodeAt(i);
11     hash = ((hash << 5) - hash) + char;
12     hash = hash & hash;
13   }
14   return hash.toString(16);
15 }

```

Listing 2: Intelligent Cache Implementation

### 3.6 Architectural Decisions Summary

Bun Runtime	Hybrid Retrieval	Google Gemini 2.5 Flash
Fast, built-in TypeScript , native tools and hot reload	Combines keyword + semantic search , robust RRF fusion	Multilingual, fast, cost-effective , supports tool calling

Table 2: Compact summary of key architectural decisions.

### 3.7 Configuration Options

Parameter	Type	Default	Description
chunkSize	number	1000	Characters per text chunk
chunkOverlap	number	200	Overlap between chunks
topK	number	5	Number of documents to retrieve
rrfK	number	60	RRF smoothing constant
maxTokens	number	4096	Maximum LLM response tokens
temperature	number	0.7	LLM creativity/temperature

## 4 MLOps-Oriented Configuration Management

To integrate MLOps principles into our system, we adopted a **configuration versioning strategy** that separates model-related settings from data processing parameters.

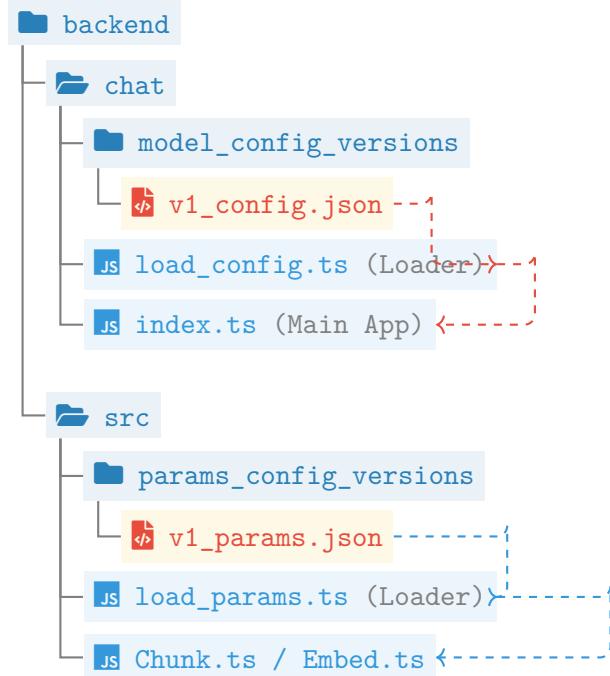


Figure 5: Project directory structure highlighting the separation of configuration versions (JSON) from application logic (TypeScript).

### 4.1 Model Configuration Versioning

Located in `backend/chat/model_config_versions`, this module controls the LLM behavior. A loader script (`load_config.ts`) dynamically injects these settings, allowing us to swap models or prompt strategies without changing the application code.

```

1  {
2      "model": {
3          "provider": "openrouter",
4          "name": "google/gemini-2.5-flash",
5          "maxSteps": 5
6      },
7      "rag": {
8          "topK": 5
9      },
10     "systemPrompt": "You are a helpful assistant... ALWAYS use the RAG
11         tool when the user asks about specific details... If information
12             is not found, let the user know."
13 }

```

Listing 3: Example Model Configuration (`v1_config.json`)

## 4.2 Pipeline Parameter Versioning

Located in `backend/src/params_config_versions`, this module controls the Retrieval pipeline. This allows for testing of chunking strategy.

```

1  {
2      "chunking": {
3          "chunkSize": 1000,
4          "overlapSize": 200
5      },
6      "embedding": {
7          "modelName": "Xenova/all-MiniLM-L6-v2",
8          "pooling": "mean",
9          "normalize": true
10     },
11     "processing": {
12         "removeUrls": true,
13         "normalizeWhitespace": true,
14         "preserveNewlines": true
15     },
16     "retrieval": {
17         "topK": 5,
18         "bm25": { "k1": 1.5, "b": 0.75 },
19         "hybrid": { "rrfK": 60 }
20     }
21 }
```

Listing 4: Example Pipeline Parameters (v1\_params.json)

## 4.3 Implementation Constraints and Limitations

While applying MLOps principles improved modularity, certain industrial MLOps components were adapted or omitted due to specific project constraints:

- ⚠ **No Data Versioning (DVC):** Standard data versioning was not doable because the dataset consists of documents from different universities (FSB, MIT, Harvard, etc.) with no changes applicable. The "data" is treated as a static knowledge base.
- ⚠ **No Training/Test Registry:** As the system relies on pre-trained Large Language Models (Gemini) and pre-trained Embeddings (MiniLM), there is no model training phase. Consequently, we cannot store training metrics (Loss, Accuracy) or maintain a labeled test set for regression testing.

## 5 Common Failure Modes

- Ambiguous Queries:** General questions are too vague. The system often focuses on a single document and ignores others.

The Faculté des Sciences de Bizerte (FSB) offers engineering programs in computer science. These programs are structured as follows:  
 Un Cycle Préparatoire Intégré (Integrated Preparatory Cycle):  
 Duration: 4 semesters.  
 Focus: Specialties related to computer science and technology.  
 This cycle prepares students for the engineering degree program.  
 Un Cycle de Formation pour l'obtention d'un Diplôme National d'Ingénieurs (DNI) (Engineering Degree Program):  
 Duration: 3 years.  
 Admission: Through a competitive exam (concours) open to students who have completed the integrated preparatory cycle at the FSB.  
 Specializations:  
 Génie logiciel et système d'information (Software Engineering and Information Systems)  
 Génie logiciel et ingénierie des données (Software Engineering and Data Engineering)  
 The FSB emphasizes the development of transversal skills, certifications in various domains, industry integration, and entrepreneurship through its 4C center. The engineering program also highlights the importance of the English language and entrepreneurship. The computer science department has a strong faculty with many engineering doctors in various IT-related fields.

- Formatting Issues:** Tables or non-standard PDF elem may not be parsed correctly.
- Multilingual Institution Names:** The system struggles to recognize institution names written in different languages or if there are small typos.

I'm sorry, but based on the documents I have access to, there is no information about the "Faculty of Science of Bizerte." The results primarily focus on computer science programs at Oxford University.

faculté des sciences de bizerte

The Faculté des Sciences de Bizerte (FSB) focuses on engineering training in computer science. They offer:  
 Un Cycle Préparatoire Intégré (Integrated Preparatory Cycle): This cycle lasts 4 semesters and focuses on specialties related to computer science and technology.  
 Un Cycle de Formation pour l'obtention d'un Diplôme National d'Ingénieurs (DNI): This is a 3-year engineering degree program with specializations in:  
 Génie logiciel et système d'information (Software Engineering and Information Systems)  
 Génie logiciel et ingénierie des données (Software Engineering and Data Engineering) Admission to this cycle is through a competitive exam for students who have completed the integrated preparatory cycle at the faculty.  
 The Faculty of Science of Bizerte has a department of computer science with over thirty engineering doctors specializing in computer science, software engineering, information technology, and telecommunications. They also have a 4C center focused on developing transversal skills, certification in various fields, industry integration, and entrepreneurship. The engineering program emphasizes English language and entrepreneurship.

- Incomplete Context for Broad Questions:** Broad questions often result in the system focusing on a single relevant document, missing information from others unless explicitly referenced.

give me diffrent programs that allows u to be a web developer

Based on the information retrieved, while a "Computer Science" degree can lead to roles like "software developer," there's a specific course mentioned: "Framework Web & Mobile" (UC4241: Framework Web). This course aims to equip students with the skills for web development, specifically mastering the front-end of web applications. It covers:  
 Web development fundamentals: Definition, functioning, history, and architectures of the web.  
 HTML5 and CSS3: Essential languages for web structure and styling.  
 JavaScript/ECMAScript: Introduction to JavaScript, its core, and client-side web application.  
 TypeScript: A superset of JavaScript.  
 Angular Framework: Developing ergonomic, interactive, and dynamic interfaces and adapting web pages for different devices (responsive design).  
 Therefore, the "Framework Web & Mobile" program directly addresses skills needed for a web developer.