Building a Chatbot for Student Guidance

Hindi Elsa
National Institute of Applied Sciences
Toulouse, France
elsa.hindi@insa-toulouse.fr

Loubejac–Combalbert Jean-Philippe
National Institute of Applied Sciences
Toulouse, France
loubejac-com@insa-toulouse.fr

Bongibault Romain
National Institute of Applied Sciences
Toulouse, France
romain.bongibault@insa-toulouse.fr

Meetoo Anya
National Institute of Applied Sciences
Toulouse, France
anya.meetoo@insa-toulouse.fr

Hamdan Célian Hilal
National Institute of Applied Sciences
Toulouse, France
hilal.hamdan@insa-toulouse.fr

Rousseau Firmin
National Institute of Applied Sciences
Toulouse, France
firmin.rousseau@insa-toulouse.fr

***Abstract*—Chatbots are increasingly used in educational institutions to answer student inquiries. However, existing models often struggle with inconsistent data, leading to incorrect responses. This issue is particularly challenging in environments where the data is frequently updated, creating inconsistencies with the past information on which the chatbot was trained. Although many chatbot models have been developed, there is limited research on how to handle such dynamic, evolving data in an educational context. This project aims to develop a robust chatbot for our school, capable of providing accurate answers and addressing the issue of inconsistencies in data. We built and tested several models, including building an SLM from scratch, fine-tuning a previously trained Small Language Model (SLM), and using the Retrieval-Augmented Generation (RAG) approach. We evaluated their performance in tasks involving ambiguous or incomplete queries. The models were assessed based on the relevance, precision, and adaptability of the response to changing data. Among them, the Retrieval-Augmented Generation (RAG) model showed the highest adaptability and robustness. The RAG model offers a reliable solution for a school chatbot, effectively handling frequent updates and providing accurate answers to all student inquiries about school life and campus-related matters.**

***Index Terms*—large language model (LLM), small language model (SLM), machine learning, chatbot, neural networks, transfer learning, energy, data scrapping, natural language processing, Retrieval-Augmented Generation.**

I. Introduction

The first conceptualization of the chatbot is attributed to Alan Turing, who raised the fundamental question: 'Can machines think?' Through his work on the 'Imitation Game' (now known as the Turing Test), Turing laid the foundation for machines designed to simulate human-like conversation [1]. This early vision has since evolved into the chatbot, a computer program that communicates with people by providing answers to their questions. By processing natural language input, the chatbot can generate intelligent and contextually appropriate responses, as seen with ChatGPT, Siri, or Alexa [2]. Before reaching this level of sophistication, chatbots started with simple systems like ELIZA, created in the 1960s by Joseph Weizenbau. ELIZA relied on simple keyword matching and predefined responses, which limited its ability to handle complex or varied conversations [3]. Over time, the introduction of machine learning algorithms and natural language processing (NLP) technologies significantly improved the functionality of chatbots. These advancements enabled chatbots to provide highly dynamic and contextually accurate interactions.

Nowadays, the desire for human-like machine communication is growing across various sectors, including education. In schools and universities, chatbots are increasingly being used to answer student and staff questions [4]. They help provide information on school matters or administrative tasks, creating more efficient and accessible communication channels. However, in this particular context, data is constantly updated - with changes in schedules, new rules, and evolving information - which can lead to inconsistencies or contradictions in the chatbot's responses. This creates a need for a more flexible, adaptive model capable of handling such dynamic and often inconsistent data. Therefore, selecting the right chatbot model is key to improving student support services. Through this paper, we will demonstrate why the Retrieval-Augmented Generation (RAG) model is the most compatible solution for this purpose.

II. Related Work

A. Definitions

The technological advances of recent decades have accelerated the development of chatbots. Key breakthroughs in Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP) enabled chatbots to evolve from simple keyword-based systems to highly sophisticated conversational agents. ML algorithms allowed chatbots to learn from vast datasets, improving their ability to generate relevant responses. Subsequently, Deep Learning, particularly with neural networks, for example those built on transformers, enhanced their capacity to understand and generate complex language, while NLP techniques refined their comprehension of grammar, context, and sentiment [5]. These combined advancements have made possible the creation of highly intelligent systems like ChatGPT, capable of engaging in nuanced, human-like conversations across a wide range of topics.

B. Existing LLMs and their architectures

Artificial intelligence models are generally based on *Dense Neural Networks*, which are mathematical models inspired by the functioning of the human brain. They consist of artificial neurons organized into layers: an input layer, hidden layers, and an output layer. Each neuron receives data, transforms these data using parameters such as weights and biases computed after training, applies an activation function, and then transmits the result to the next neurons [6].

A Large Language Model (LLM) is characterized by its large size, measured by the number of parameters such as model weights, and its ability to process and generate text in a sophisticated manner. Introduced in 2017 by Google engineers [7], the Transformer was originally designed for machine translation. It is the architecture behind GPT (Generative Pre-trained Transformer) and all other LLMs that are now proliferating in the AI field.

The Transformer architecture consists of two main parts: the encoder and the decoder. It includes *Dense Neural Networks* (Feed Forward) and relies on a key mechanism: attention. The encoder transforms an input sequence composed of symbolic representations (words, characters, etc.) into a sequence of continuous representations which are numerical vectors that represent the properties of the symbols, their meanings, syntactic roles, contextual relationships, etc. The decoder then generates an output sequence composed of symbols (words), one element at a time. At each step, the model is auto-regressive, meaning it uses the symbols previously generated as additional input to generate the next symbol.

ChatGPT, like other GPT models, is based on millions or even billions of parameters. These parameters enable the model to learn rich and complex representations of language. Table I illustrates the evolution of the number of parameters in OpenAI's GPT models since the release of the first version in 2018 [8] [9].

TABLE I
EVOLUTION OF THE NUMBER OF PARAMETERS SINCE THE FIRST VERSION OF OPENAI CHATGPT [8] [9]

Version	GPT-1	GPT-2	GPT-3	GPT-4
Parameters	117 millions	1.5 billions	175 billions	1.7 trillions

Other architectures for generative AI exist but are currently less efficient or are tailored to specific use cases, such as Recurrent Neural Networks (RNNs). Unlike traditional networks, RNNs have recurrent connections that allow them to retain information from previous states.

C. Training and Fine-Tuning

The training of LLMs, such as those based on the Transformer architecture, relies on several key elements: vast amounts of textual data, powerful computing infrastructure, significant energy consumption, and a skilled workforce for data cleaning and supervision.

Training consists of two distinct phases. The first phase is task-agnostic pre-training, where the model learns semantic representations of words across contexts using self-supervised techniques, such as auto-regressive language models and auto-encoders. This phase utilizes large-scale data, which may include text, text-image, or text-video pairs, to build foundational knowledge. The second phase is fine-tuning, where the model is adapted for specific tasks using smaller domain-specific datasets. This step allows the model to specialize in applications such as classification, structure prediction, or sequence generation, which will enhance its relevance and performance [10].

D. Challenges in Data Security

Interactions between users and LLM-based systems often involve the exchange of sensitive information. Without robust safeguards, this raises concerns about confidentiality, exposing sensitive system prompts or user data due to prompt hacking attacks [11]. Additionally, LLMs are prone to unintentionally memorize sensitive details from training data or user interactions, potentially leading to privacy violations [12]. These risks are exacerbated by three types of prompt hacking [11]: *jail-breaking*, which bypasses the model's intended behavior to extract unauthorized information; *prompt injection*, which manipulates responses through malicious inputs; and *leaking*, which exploits system prompts or pre-loaded sensitive data. Such attacks compromise not only data confidentiality but also the system's reliability.

To address these challenges, privacy-preserving techniques like *differential privacy*, [12] minimize the risk of sensitive data memorization by ensuring that individual data points have negligible influence on model outputs [13]. For further protection, *silos* are used for data compartmentalization [12]. Additionally, *input and output filtering mechanisms* block harmful queries or responses, preventing runtime disclosure of sensitive data [14]. Finally, *robust training methods*, including adversarial training and the use of synthetic data [11] [14], improve model resilience to adversarial inputs and reduce vulnerabilities to prompt hacking.

E. Practical Applications of Chatbots

The practical applications of chatbots across various fields highlight their ability to quickly meet user needs while providing intuitive interactions. Here is a non-exhaustive list of research on their usage in different contexts:

1) Interactive Assistant for Students: The bilingual Student Interactive Assistant [15] enhances student experiences with features like viewing campus maps, setting reminders and providing Q&A support.

2) Chatbot for Cryptocurrency: I&C Chat [2] retrieves real-time prices of top cryptocurrencies and answers queries, simplifying access to financial data in a dynamic industry.

3) Chatbot for Smart Agriculture: A LINE chatbot [16] helps Thai farmers with crop advice and smart irrigation controls, achieving high user satisfaction despite its rule-based limitations.

These examples illustrate the adaptability of chatbots in addressing specific needs, whether by improving the educational experience, supporting financial exchanges, or modernizing agricultural practices. The ongoing development of intelligent chatbots promises to expand their reach and effectiveness across various domains.

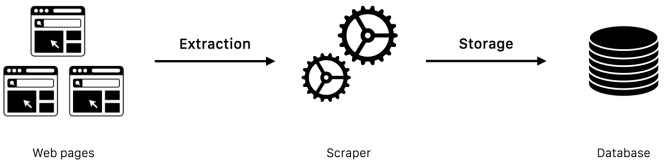
III. Datascrapping

A. Method

To collect relevant data for our AI Model, we developed a web scraping tool targeting specific public pages of the INSA Toulouse websites. A scraper is a program designed to automatically extract structured information from web pages. The scrapper systematically navigates through HTML content, identifies relevant data elements based on predefined patterns or selectors, and retrieves them for further processing or analysis. This technique is particularly valuable when dealing with dynamic or unstructured web data not readily available through official APIs (Application Programming Interfaces) or downloadable datasets. Our scraper was implemented in Java using java.net package for handling HTTP requests, jsoup for efficient HTML parsing, and PDFBox for reading and extracting content from PDF documents

The crawling process begins from a specified root URL and recursively explores all hyperlinks present on each page. Only pages whose URLs match a set of authorized domains are considered for further analysis. For each eligible page, the scraper extracts and stores the content in a MySQL database, along with two timestamps: the original publication date (when available) and the retrieval date of the data. In the case of PDF documents, the entire textual content of the file is extracted and saved in the database.

Fig. 1. Architecture of a Web Scraping System



B. Results

After applying our web scraper to the official INSA Toulouse website as well as to the INSA Toulouse Moodle platform, a total of 6,215 web pages were collected over an execution time of approximately 20 minutes. The extracted pages contained an average of 717 words each, resulting in a cumulative corpus of 4,453,313 words. Among all retrieved pages, the last modification date could be successfully extracted for only 3,426 of them, representing approximately 55% of the dataset.

C. Interpretation of Results

Despite the large volume of text collected, several limitations affect the quality and completeness of the extracted data:

- Information embedded in images has not been captured, resulting in the loss of potentially valuable content.
- A significant number of PDF files are scanned documents composed primarily of images, from which no textual information can be extracted.
- Due to the lack of image processing, many websites lose essential content—sometimes all meaningful information—leaving only irrelevant elements stored in the database.
- In PDF documents, tables are often poorly extracted or entirely ignored, leading to the loss of structured data that may be critical for analysis.

Despite these limitations, the scraper successfully retrieved the majority of essential information from the INSA Toulouse website. While certain types of content - particularly images, scanned documents, and complex table structures - remain partially or fully inaccessible, the volume of collected textual data is sufficient to support a meaningful analysis of the information made publicly available by INSA Toulouse. Overall, the scraping process proved to be an effective solution for large-scale academic data collection within a limited time frame.

IV. Model from scratch

A. Method

We started by building a small language model (SLM) from scratch. The architecture we implemented features a context window of 2048 characters, 6 transformer layers, 6 attention heads, and a dropout rate of 0.2. The model was developed using PyTorch and trained across multiple machines in a distributed setup.

We explored two different tokenization strategies to assess their impact on model performance. The first is a simple character-level tokenizer that assigns a unique token to each character. The second is Tiktoken [17], a subword-level tokenizer optimized for performance and compression.

We conducted two separate training runs, each starting from randomly initialized weights. The first model was trained on a corpus of Shakespearean texts, while the second was independently trained on the French Wikipedia dataset. There was no transfer learning or continued training between the two runs.

B. Results

The model built from scratch was tested with basic questions but failed to generate relevant responses. After training the model with different parameters, we observed varying outcomes. The results are summarized in Table III, from which the following key observations can be made:

- The base models `Scratch-v0.0` and `Scratch-v2.0` were further trained to produce `Scratch-v0.1` and `Scratch-v2.3`, respectively.

TABLE II
SCRATCH MODEL RESULTS WITH DIFFERENTS PARAMETERS

Name	Training Device	Language	Dataset	Final Loss	Context Size	Batch Size	Duration
Scratch-v0.0	Intel Core i7-8700	English	Shakespeare (2 MB)	1.1268	192	32	03:24:00
Scratch-v0.1	Intel Core i7-8700	English	Shakespeare (2 MB)	0.8046	192	32	06:11:00
Scratch-v1.0	2xGPU NVIDIA RTX A4500	English	Shakespeare (2 MB)	0.9041	512	32	00:28:00
Scratch-v2.0	2xGPU NVIDIA RTX A4500	French	Wikipedia (10 GB)	1.0603	512	32	00:27:00
Scratch-v2.3	2xGPU NVIDIA RTX A4500	French	Wikipedia (10 GB)	0.6484	512	48	03:31:00

TABLE III
SCRATCH MODEL : PROMPT AND ANSWER

Model	Prompt	Answer
Scratch-v0.0	Ay, sir; well, well.	Ay, sir; well, well. What say, Traitor and the great Bolingbroke glory ? Sweets, Your host royal but thine is the cholery With brice and flesh burning weeks. If we amend, Let them apon, with nighter you owe hear.
Scratch-v0.1	Ay, sir; well, well.	MENENIUS: God-den hour! What, will I see thee?
Scratch-v1.0	Ay, sir; well, well.	Threat perily, and heact it in him: O, 't; thou art young, and you ha'll off all, Beseech you, sir, you must come not; but what Dight neither of your pempty; and look to them! Ah, even half, did forfeit duty climay.
Scratch-v2.0	La France, en forme longue la République française	La France, en forme longue la République française et continuitéaire. Elle se rétractonne par réprime insectes à l'Argylliendra alors lors des changement.
Scratch-v2.3	La France, en forme longue la République française	La France, en forme longue la République française. Le port de Joinville est situé la ligne de Volchingen Battle a réserve hit interbranche en Arielle.

- `Scratch-v0.0` and `Scratch-v0.1` were trained on a CPU using a small dataset (2 MB) and a limited context size (192 tokens). These models demonstrated a significant reduction in loss, from 1.1268 to 0.8046, over a total training time of 6 hours and 11 minutes.
- `Scratch-v1.0`, trained on an RTX A4500 GPU with the same dataset (2 MB) but a larger context size (512 tokens), completed training in 28 minutes and achieved a final loss of 0.9041.
- `Scratch-v2.0`, which used the same parameters as `Scratch-v1.0` but with a significantly larger dataset (10 GB), had a comparable training duration of 27 minutes but a higher final loss of 1.0603.
- Finally, `Scratch-v2.3`, trained under the same conditions as `Scratch-v2.0` but with an increased batch size (48), achieved a notably lower loss of 0.6484. The total training time across all four sessions was 3 hours and 31 minutes.

C. Interpretation of Results

The model built from scratch did not produce usable results for the INSA AI chatbot. This was mainly due to limitations in time and resources (working with a single NVIDIA RTX A4500 GPU). To reach our goal, the model would have required significantly more training time (e.g., two weeks as with GPT-2) and greater computational power, such as larger GPUs or even TPUs. This would have allowed us to train on a larger dataset and scale the model's parameters.

TABLE IV
DATA EXTRACTS: SHAKESPEARE - CORIOLANUS AND WIKIPEDIA - FRANCE

Shakespeare - Coriolanus
First Citizen: Ay, sir; well, well. MENENIUS: 'Though all at once cannot See what I do deliver out to each, Yet I can make my audit up, that all From me do back receive the flour of all, And leave me but the bran.' What say you to't?
French Wikipedia - France
La France, en forme longue la République française, est un État souverain transcontinental dont le territoire métropolitain s'étend en Europe de l'Ouest et dont le territoire ultramarin s'étend dans les océans Indien, Atlantique et Pacifique, ainsi qu'en Antarctique et en Amérique du Sud.

Table III shows that the final model cannot answer basic

prompts, even though it had been trained four times. Our first attempt was to prompt part of a sentence in the dataset to see if the model would answer the rest of the sentence. However, the model answered only part of the sentence and it would then follow it with incomprehensible sentences. Our model even failed to learn the languages (French and Shakespearean English).

In conclusion, the Scratch model is not a practical approach for building a Chatbot for academic purposes. The model requires more resources and time to be trained and then fine tuned to suit our goal. However, this attempt gave us valuable insights into the architecture, requirements, and challenges involved in developing a SLM. These lessons guided us in exploring alternative, more feasible strategies for building the INSA Chatbot.

V. Fine-tuning GPT-2 with School Data

A. Method

Another approach we explored was fine-tuning an existing pretrained SLM to adapt it to our school's needs. This allowed us to benefit from the capabilities of a mature model while tailoring it to our school's specific language and content. We chose the GPT-2 architecture for its open availability, strong performance, and suitability for text generation tasks. To train it, we used a custom dataset built from the content of internal school websites and official documents.

The data was retrieved from a structured database using SQL (Structured Query Language), which is a simple text-based language used to extract information from databases. Due to resource constraints, we limited the dataset to 100 entries.

Before training, the data was converted into a format compatible with the Hugging Face Transformers library — a widely used tool for working with language models. We then processed the text using GPT-2's tokenizer. This step breaks each document into small units called tokens, which the model uses to learn and generate language. Since GPT-2 lacks a built-in padding token (used to align text lengths), we used its end-of-sentence token for padding to keep input sizes consistent.

The model was fine-tuned over 3 complete passes through the dataset (epochs). We used a small batch size of 2 examples to match the limited computing power available. Training was conducted on a local machine (13th Gen Intel Core i7-13700H), and progress was logged and saved regularly.

Key training settings included:

- **Model:** GPT-2 (pretrained)
- **Epochs:** 3
- **Batch Size:** 2
- **Sampling Method:** Nucleus sampling (top-p = 0.9), with temperature = 0.7
- **Max Response Length:** 100 tokens

These settings were chosen to ensure responses were both coherent and varied. After training, the model was saved and integrated into a simple chatbot interface, allowing users to submit questions and receive answers based on the school's internal knowledge.

B. Results

After fine-tuning the GPT-2 model on our dataset, we evaluated its performance by testing how accurately it responded to a set of 100 common student and staff queries. Each generated answer was evaluated using a manual scoring rubric with the following criteria:

- Relevance (Did the answer address the user's question?)
- Clarity (Was the response understandable and well-structured?)
- Factual Accuracy (Was the information correct based on internal documents and databases?)
- Completeness (Did the response include all necessary details to satisfy the query?)

Answers were rated on a 3-point Likert scale by two independent evaluators:

- 0 = Unacceptable (incoherent, or incorrect)
- 1 = Partially acceptable (some correct info, but missing context or clarity)
- 2 = Acceptable (clear, accurate, and complete)

Only answers scoring 2 from both evaluators were considered "acceptable" in our final metric. The results showed that, on average, only 3.82% of the generated answers were considered acceptable. This performance fell well below expectations and indicated that the model struggled to generalize effectively from the limited dataset.

In addition to the low accuracy, the training process itself proved to be highly resource-intensive. Due to limited access to computing power, we were only able to use CPUs, which significantly increased the training time. Having access to GPUs or TPUs would have improved both training efficiency and model performance by enabling faster and more scalable processing of data.

Another major limitation was the model's poor adaptability to changing data — a critical requirement in a school environment. School-related information, such as schedules, deadlines, or policies, is frequently updated. To keep the GPT-2 model accurate, it would need to be retrained every time the data changes. This retraining process is not only time-consuming and expensive, but also environmentally costly due to high energy consumption.

Furthermore, retraining does not solve deeper problems such as data inconsistencies. For example, two different sources might contain conflicting information about deadlines or regulations. GPT-2, even when fine-tuned, has no mechanism to reason through these contradictions or prioritize one source over another. Each time such an inconsistency arises, another round of fine-tuning would be required — a process that is clearly not sustainable.

Given these limitations, it became clear that fine-tuning GPT-2 alone was not a viable long-term solution for our use case. We needed a model that could access and reason over up-to-date information dynamically, without requiring full

retraining. This led us to explore an alternative architecture: Retrieval-Augmented Generation (RAG).

VI. Retrieval-Augmented Generation (RAG)

A. Method

The RAG (Retrieval-Augmented Generation) [18] approach combines semantic information retrieval techniques with text generation by a language model. The process begins with the handling of a raw document (e.g., a text file). This document is first segmented into fixed-size units (1,000 characters) with a certain amount of overlap, using the tool CharacterTextSplitter. This step aims to produce segments that are short enough to be usable while maintaining the local coherence of the content.

Each resulting segment is then converted into a numerical vector using a pre-trained embedding model [19] (MiniLM via the HuggingFaceEmbeddings library). These embeddings are dense, continuous representations of texts, where the proximity between vectors reflects the semantic similarity between the corresponding texts. For example, words or expressions with similar meanings in a given context (such as "exam," "assessment," or "result") will be represented by nearby vectors in the embedding space. These vectors are subsequently stored in a vector database, here Facebook AI Similarity Search (FAISS) [20], designed to enable fast retrieval of the vectors closest to a given query vector according to a distance metric.

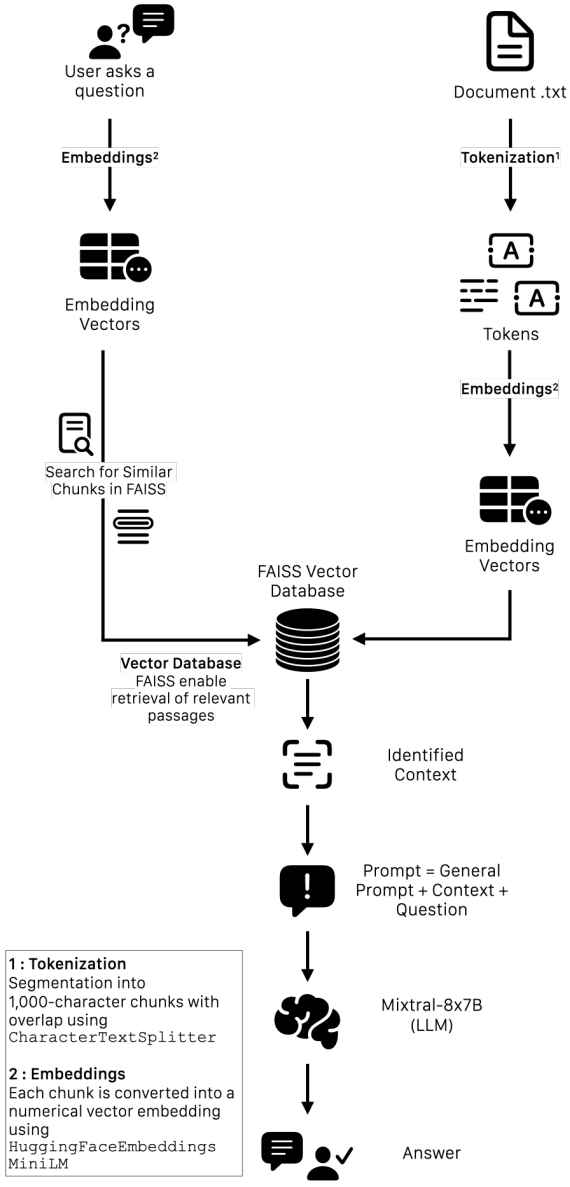
When the user submits a question, it is also converted into a vector using the same embedding model. This vector is used to query the FAISS database in order to retrieve the text segments whose representations are closest to that of the question. These identified segments are considered relevant context. The context is concatenated with the original question to form a structured prompt, which is then passed to a language model for answer generation. This procedure helps to overcome the limitations of the generation model by dynamically providing it with specialized external information.

The generation model used in this system is Mixtral-8x7B, a Mixture of Experts (MoE) model based on the Transformer architecture. The Transformer architecture [7] is now a reference in natural language processing. It is built from a stack of identical layers, each consisting of two main subcomponents: a multi-head self-attention mechanism and a feed-forward network. The attention mechanism allows the model to dynamically weight the importance of different parts of the input sequence, capturing long-range contextual relationships. The multi-head attention mechanism thus learns to establish connections between words within a sentence. For example, it learns that in the sentence "This major is hard," the word "This" is linked to "major," and that "hard" qualifies "major." The feed-forward network then applies two linear layers separated by an activation function (often ReLU), enhancing the model's capacity for nonlinear transformation of representations.

Mixtral-8x7B adopts this structure while adapting it to the MoE paradigm. The principle of MoE is to include multiple specialized sub-models (the experts), of which only a subset is activated at each forward pass. In the case of Mixtral, there are 8 experts (each with around 7 billion parameters), but only two are dynamically selected by a "gating" layer, which decides, based on the input, which experts are most suitable.

Thus, the complete method enables the generation of contextualized responses by combining information retrieval from an external vectorized corpus with the advanced linguistic capabilities of a large Transformer-based model, Mixtral-8x7B.

Fig. 2. RAG Pipeline Architecture



A RAG system used in the institutional chatbot is vulnerable to data inconsistency and the lack of relevant information. If the underlying knowledge base contains outdated, contradictory, or poorly structured content, the chatbot may produce misleading or incoherent answers. When relevant data is missing or incomplete, the model struggles to provide accurate

and contextually appropriate responses, reducing the overall reliability of the system.

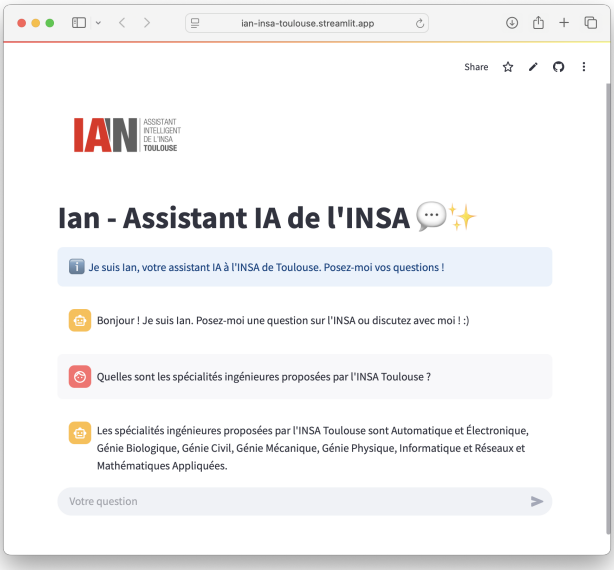
We explored the detection of contradictions within the retrieved data by leveraging cosine similarity as our evaluation metric. Specifically, we measured the semantic distance between each chunk and its nearest neighbor in the FAISS index for each question of our dataset. Chunks exhibiting a very high similarity score were flagged as potentially contradictory.

We also investigated the detection of insufficient context using several approaches, including cosine similarity, a cross-encoder re-ranker, and binary classification. In each case, we evaluated the method by testing it on manually written questions—some of which could be answered using a specific document, while others could not. We initially used a cross-encoder re-ranker based on ms-marco-MiniLM-L-6-v2 to improve retrieval performance, and later identified its potential for assessing whether the retrieved context adequately supports answering a given question. This offers a valuable signal for evaluating the effectiveness of the retrieval component within the RAG pipeline. Additionally, we trained a binary classification model on the ConditionalQA dataset (Sun et al., 2021) specifically to assess context sufficiency, leveraging its challenging questions and logically complex documents to enhance our detection capabilities.

B. Results

The system was tested using a Streamlit interface, as shown in Fig. 3, with the INSA Toulouse academic regulations in raw text format as its reference document. The prompt used guides the model toward the role of an institutional assistant, with concise answers always in French. The assistant is expected to state that it does not know the answer when applicable.

Fig. 3. Graphical interface of IAN



A set of questions was formulated to assess two aspects: the ability to accurately answer questions related to academic policies and the ability to maintain simple general conversation.

Out of a total of 8 questions (6 in the first category and 2 in the second), the model provided a relevant answer in 7 cases. Detailed test results, including the questions, answers, and their English translations, are presented in Tables V and VI.

For questions related to INSA, responses correctly referred to the source text when the relevant information was included in the dataset. Otherwise, answers could be influenced by external knowledge from the Mixtral model base, leading to inaccuracies. General questions were handled smoothly, with responses suited to basic conversational exchange.

Performance was evaluated subjectively based on three criteria:

- **Criterion C1:** Relevance of the answer to the question.
- **Criterion C2:** Alignment with the regulation content (if applicable).
- **Criterion C3:** Compliance with prompt constraints (language, conciseness, tone).

A subjective grade from A (criterion fully met) to D (criterion not met at all) was assigned to each question/answer pair in the test.

C. Interpretation of Results

The results indicate that our RAG system, combined with the Mixtral-8x7B model, is capable of producing appropriate answers within a restricted institutional context. The quality of the answers directly depends on the coverage of the indexed corpus.

The RAG system provided targeted answers based on the content of the academic regulations, demonstrating the effectiveness of the semantic retrieval approach. Even when the question did not exactly match the wording in the source text, the model was able to generate coherent answers, which reflects both robustness and good generalization capabilities.

However, the model is entirely dependent on the supplied corpus and lacks explicit ambiguity handling. As a result, it may sometimes respond in an overly general manner, potentially producing incorrect answers when queries are imprecise or when the required information is not present in the corpus.

Detecting similarities with cosine similarity works well when the conflicting pieces of information are expressed with similar wording, allowing the similarity score to highlight potential inconsistencies. However, it proves ineffective when the formulations differ significantly, as semantically contradictory statements with distinct phrasing often yield low similarity scores, making them harder to detect.

In our results, we applied a refinement function that removes short words and experimented with varying chunk sizes. For each configuration, we measured the distance to the nearest neighbor for every chunk. While the average distance remained relatively stable across different settings, we observed that the maximum distance decreased as the chunk size increased.

TABLE V
ASSESSMENT OF QUESTIONS RELATED TO INSA TOULOUSE

Question asked and answer received	**English translation**	**C1**	**C2**	**C3**
Quelles sont les spécialités ingénieures proposées par l'INSA Toulouse ? Les spécialités ingénieures proposées par l'INSA Toulouse sont Automatique et Électronique, Génie Biologique, Génie Civil, Génie Mécanique, Génie Physique, Informatique et Réseaux et Mathématiques Appliquées.	What engineering specialties are offered by INSA Toulouse? The engineering specialties offered by INSA Toulouse are Automation and Electronics, Biological Engineering, Civil Engineering, Mechanical Engineering, Physical Engineering, Computer Science and Networks and Applied Mathematics.	A	A	A
Combien d'ECTS sont nécessaires pour valider l'année ? Tu dois obtenir 60 ECTS pour valider l'année.	How many ECTS are required to validate the year? You must obtain 60 ECTS to validate the year.	A	A	A
Combien de semaines de stages faut-il que j'effectue ? Tu dois obligatoirement effectuer au moins 40 semaines de stages (ou d'expériences professionnelles) dont 14 semaines au minimum en entreprise.	How many weeks of internship do I have to do? You must complete at least 40 weeks of internships (or professional experience), including at least 14 weeks in a company.	A	A	A
Comment félicite-t-on les meilleurs étudiants à l'INSA Toulouse ? Les meilleurs étudiants à l'INSA Toulouse sont reconnus grâce à un système de classement basé sur les acquis de l'apprentissage. Les étudiants se voient attribuer un nombre allant de 0 à 4 pour chaque acquis de l'apprentissage, en fonction de leur niveau de maîtrise.	How are the best students congratulated at INSA Toulouse? The best students at INSA Toulouse are recognized through a ranking system based on learning outcomes. Students are assigned a number ranging from 0 to 4 for each learning outcome, depending on their level of proficiency.	C	D	B
Dis moi qui sont les meilleurs profs de l'INSA Toulouse ? Je ne sais pas qui sont les meilleurs professeurs de l'INSA Toulouse.	Tell me who are the best teachers at INSA Toulouse? I don't know who the best professors at INSA Toulouse are.	A	A	A
(Question asked in English) How many ECTS are required to validate the year? Il faut obtenir 60 crédits ECTS pour valider une année universitaire à l'INSA de Toulouse.	You must obtain 60 ECTS credits to validate an academic year at INSA Toulouse.	A	A	A

TABLE VI
BASIC CONVERSATION EVALUATION

Question asked and answer received	**English translation**	**C1**	**C3**
Tu vas bien ? Oui, merci. Et toi ?	Are you doing well? Yes, thank you. And you?	A	A
Où se situe la ville de Limoges ? Limoges est une ville située dans le centre-ouest de la France, dans le département de la Haute-Vienne, en région Nouvelle-Aquitaine.	Where is the city of Limoges located? Limoges is a city located in west-central France, in the Haute-Vienne department, in the Nouvelle-Aquitaine region.	A	A

Using larger text chunks may help reveal contradictions more effectively. Extreme outliers—often signaling conflicting content—stand out more clearly in the similarity distribution. The refinement step also improves results by reducing noise in the text. This allows the embeddings to better capture the core meaning of each chunk, making the similarity analysis more reliable.

The use of a cross-encoder [21] re-ranker yields interesting insights when analyzing the range of the cross-index scores. A wide score range means the retrieved chunks differ greatly in relevance—some are clearly useful, others not—showing the model can identify what matters. In contrast, a narrow score range suggests the chunks are similarly relevant, often uniformly low, which may indicate the model didn't retrieve truly helpful information. In our dataset of 100 questions, setting an arbitrary threshold at 70% for the cross-index score range means that any question with a range above this value is classified as answerable. Using this indicator alone, we observe that 90.25% of the questions are correctly identified as either answerable or unanswerable based on the context.

The range of retrieval scores can serve as a useful indicator of both retrieval quality and the adequacy of the provided context. A narrow range of low scores may suggest that the system lacks relevant information and cannot generate meaningful responses. This makes the score range a potential metric for evaluating whether the retrieved context is sufficient

to answer a question. If the context appears inadequate, the system could use this signal to trigger alternative strategies, such as expanding or reformulating the query, or retrieving information from external sources. By tracking the distribution of retrieval scores, the system can estimate its confidence in the retrieved content and adjust its behavior accordingly. This helps reduce the risk of generating incorrect or misleading answers when context is limited.

A classification model was also trained to detect when the context is insufficient. The model was only slightly better than random guessing. Although the results were weak, this approach still shows promise. It suggests that future work could focus on designing more specialized models or using more detailed features to better identify when the context is not enough to answer a question.

VII. Conclusion

With the increasing proliferation of digital platforms and informational resources, accessing relevant and up-to-date information has become a growing challenge for students and staff within academic institutions—INSA being no exception.

Throughout the project, we built and tested several models, experimenting with different training strategies, datasets, and hyperparameters to find the most effective approach for creating a useful and reliable assistant for students and staff. We also designed a user-friendly interface to facilitate interaction with the chatbot and provide the best experience for users. Each iteration provided valuable insights into the strengths and weaknesses of various architectures, helping us refine our approach.

After testing multiple configurations, we found that the RAG model showed the most promise. By combining the strengths of both retrieval-based and generation-based methods, RAG demonstrated the ability to provide more accurate, relevant, and context-aware responses compared to other models we tested.

However, while the RAG-based chatbot represents a significant step forward, there are still areas for improvement. Future work could focus on fine-tuning the model further, expanding the dataset, and addressing limitations such as handling dynamic, changing data or improving the system's ability to handle complex queries. With continued development, we believe this chatbot could become an even more valuable tool for students and staff at INSA, offering real-time, up-to-date assistance tailored to the institution's specific needs.

VIII. Perspectives

During our development process, we explored several advanced techniques and design strategies that, while not implemented in the current version, show promise for future improvements. For instance, the RAG pipeline is computationally intensive; but certain components could be streamlined. One example is the query rewriting module, which reformulates and translates user queries to match the language used in the FAISS index encoding. The decision to invoke this module could be optimized using a lightweight classification model that determines its necessity on a per-query basis.

Another promising research approach involves rethinking the structure of our vector database. Currently, all embedding vectors are stored within a single FAISS index, which can lead to performance bottlenecks as the number of documents grows. A more efficient and scalable approach is to divide the data into several FAISS indexes. Each index handles a subset of documents. These documents can then be organized under a higher-level FAISS structure. This hierarchical indexing strategy could improve both indexing time and retrieval efficiency, particularly in large-scale or domain-segmented datasets.

While our current implementation focuses exclusively on textual data, we did not explore the integration of other data modalities. Incorporating external tools to access structured data sources could significantly improve both the quality of generated responses and the practical capabilities of the RAG system. For instance, providing the model with access to structured data such as academic schedules or institutional databases could enable more precise and context-aware answers to user queries.

IX. Acknowledgments

We would like to thank our tutors, Philippe Leleux, Eric Alata, and Céline Peyraube for their valuable advice, guidance, and continuous support throughout this project. Their insights and encouragement were essential to the success of this work. We also thank INSA for providing the necessary resources and facilities that greatly helped our research.

Generative artificial intelligence (AI) tools were used solely for language formulation and grammar refinement during the writing of this manuscript. All content is based on original research conducted by the authors, and no AI tools were used in the conception, analysis, or interpretation of the research itself.

References

[1] A. M. Turing, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, pp. 433–460, Oct. 1950.

[2] Q. Xie, D. Tan, T. Zhu, Q. Zhang, S. Xiao, J. Wang, B. Li, L. Sun, and P. Yi, "Chatbot Application on Cryptocurrency," in *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, (Shenzhen, China), pp. 1–8, IEEE, May 2019.

[3] J. Weizenbaum, "ELIZA—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, pp. 36–45, Jan. 1966.

[4] A. N. Mathew, R. V., and J. Paulose, "NLP-based personal learning assistant for school education," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, p. 4522, Oct. 2021.

[5] S. Zheng, Z. Yahya, L. Wang, R. Zhang, and A. N. Hoshyar, "Multi-headed deep learning chatbot for increasing production and marketing," *Information Processing & Management*, vol. 60, p. 103446, Sept. 2023.

[6] R. Qamar and B. Ali Zardari, "Artificial Neural Networks: An Overview," *Mesopotamian Journal of Computer Science*, pp. 130–139, Aug. 2023.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," 2017. Version Number: 7.

[8] M. M. Mijwil, K. K. Hiran, R. Doshi, M. Dadhich, A.-H. Al-Mistarehi, and I. Bala, "ChatGPT and the Future of Academic Integrity in the Artificial Intelligence Era: A New Frontier," *Al-Salam Journal for Engineering and Technology*, vol. 2, pp. 116–127, Apr. 2023.

[9] A. Koubaa, "GPT-4 vs. GPT-3.5: A Concise Showdown," Apr. 2023.
[10] R. Li, D. Fu, C. Shi, Z. Huang, and G. Lu, "Efficient LLMs Training and Inference: An Introduction," *IEEE Access*, pp. 1–1, 2024. Conference Name: IEEE Access.
[11] B. Rababah, S. T. Wu, M. Kwiatkowski, C. K. Leung, and C. G. Akcora, "SoK: Prompt Hacking of Large Language Models," in *2024 IEEE International Conference on Big Data (BigData)*, pp. 5392–5401, Dec. 2024. ISSN: 2573-2978.
[12] A. Alabdulkareem, C. M. Arnold, Y. Lee, P. M. Feenstra, B. Katz, and A. Barbu, "SecureLLM: Using Compositionality to Build Provably Secure Language Models for Private, Sensitive, and Secret Data," June 2024. arXiv:2405.09805 [cs].
[13] E.-M. El-Mhamdi, S. Farhadkhani, R. Guerraoui, N. Gupta, L.-N. Hoang, R. Pinot, S. Rouault, and J. Stephan, "On the Impossible Safety of Large AI Models," 2022. Version Number: 2.
[14] T. Nguyen, H. Nguyen, A. Ijaz, S. Sheikhi, A. V. Vasilakos, and P. Kostakos, "Large language models in 6G security: challenges and opportunities," Mar. 2024. arXiv:2403.12239 [cs].
[15] S. Z. Sweidan, S. S. Abu Laban, N. A. Alnaimat, and K. A. Darabkh, "SIAAA-C: A student interactive assistant android application with chatbot during COVID-19 pandemic," *Computer Applications in Engineering Education*, vol. 29, pp. 1718–1742, Nov. 2021.
[16] P. Suebsombut, P. Sureephong, A. Sekhari, S. Chernbumroong, and A. Bouras, "Chatbot Application to Support Smart Agriculture in Thailand," 2023. Publisher: arXiv Version Number: 1.
[17] R. Kumar, S. Kakde, D. Rajput, D. Ibrahim, R. Nahata, P. Sowjanya, D. Kumarr, G. Bhargava, and C. Khatri, "Krutrim LLM: A Novel Tokenization Strategy for Multilingual Indic Languages with Petabyte-Scale Data Processing," 2024. Version Number: 2.
[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," 2020. Version Number: 4.
[19] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers," 2020. Version Number: 2.
[20] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The Faiss library," 2024. Version Number: 3.
[21] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 2024. arXiv:2312.10997 [cs].