Journal of Systemics, Cybernetics and Informatics (2024) 22(5), 77-87
https://doi.org/10.54808/JSCI.22.05.77

Aurel_AI: Automating an Institutional Help Desk Using an LLM Chatbot

Diego ORDÓÑEZ-CAMACHO
Facultad Internacional de Innovación PUCE-Icam – Pontificia Universidad Católica del Ecuador, daordonezc@puce.edu.ec
ICAM - Institut catholique d'arts et métiers, diego.ordonez@icam.fr
Quito, Ecuador

Rafael MELGAREJO-HEREDIA
Facultad Internacional de Innovación PUCE-Icam – Pontificia Universidad Católica del Ecuador, rmelgarejo@puce.edu.ec
ICAM - Institut catholique d'arts et métiers, rafael.melgarejo@icam.fr
Quito, Ecuador

Mohsen ABBASI
Facultad Internacional de Innovación PUCE-Icam – Pontificia Universidad Católica del Ecuador, mabassi@puce.edu.ec
ICAM - Institut catholique d'arts et métiers, mohsen.abbasi@icam.fr
Quito, Ecuador

Lucía GONZÁLEZ-SOLIS
Facultad Internacional de Innovación PUCE-Icam – Pontificia Universidad Católica del Ecuador, algonzalezs@puce.edu.ec
ICAM - Institut catholique d'arts et métiers, lucia.gonzalez@icam.fr
Quito, Ecuador

ABSTRACT

The Aurel_AI research project was born from the need to implement a virtual help desk for a university, providing accurate organizational information to both internal and external clients. The information includes details about academic programs, regulations, processes, and personnel. Aurel_AI is part of a broader research program on the use of AI in academia. Traditional solutions for a help desk, such as telephone call centers, present quality and efficiency issues that are difficult to solve. Call center staff generally lack comprehensive knowledge about the institution, rely on specific information that is sometimes outdated, require additional systems for information retrieval, and experience high turnover rates. This leads to associated costs and issues related with outdated information, resulting in inaccurate responses and long waiting times. Generative artificial intelligence models, known as Large Language Models (LLMs), offer an interesting alternative for an automated virtual help desk. These models can understand even vague and poorly structured questions and generate reasonably appropriate answers. However, they are not without flaws, as they tend to present issues like "hallucinations" when the required information is not present in their training data. To minimize this problem, it is crucial to ensure that the model has precise and comprehensive information, which needs a specific methodology for information collection, validation, and updating. Base models require an adaptation process to be used for specific cases, for which techniques like Fine-Tuning and Retrieval Augmented Generation (RAG) exist. Fine-tuning retrains a model's weights with new specific information, while RAG uses both proprietary information—in this case, from the university—and publicly available internet data. Both techniques have pros and cons that need to be evaluated to select the most suitable option. They also demand appropriate and specialized infrastructure, which is often expensive. Thus, another challenge is to find a balance between suitable equipment and reasonable costs. The final system, from the user's perspective, must be accurate, flexible, and adaptable to deliver a satisfactory experience. As the results show, Aurel_AI represents an advance in the digitalization of educational services, standing out for its ability to generate accurate and personalized responses. However, its current limitations, such as handling concurrent queries and hallucinations, underscore the need for adjustments to both infrastructure and data processing methodology. With strategic improvements, the system has the potential to consolidate itself as a replicable model for multiple university digital services.

Keywords: LLM, chatbot, automated help desk, fine-tuning, RAG.

1. INTRODUCTION

The Aurel_AI research project represents an innovative and strategic response to contemporary demands for precise and timely information within an educational institution. This project aims to implement a virtual help desk that transforms how the university interacts with its internal and external clients, optimizing the delivery of data related to academic offerings, regulations, administrative processes, and key contacts for students and collaborators. The primary objective of this project is the accuracy of the

information provided, as any error or outdated information could undermine users' trust and the service's effectiveness.

Aurel_AI is part of a larger project on the use of artificial intelligence in the academic field [1], aligning with a global trend where emerging technologies are transforming higher education, from teaching and learning processes to administrative services. In this context, Aurel_AI not only serves as a technological tool but also as a strategic component that demonstrates how AI can address complex problems efficiently, surpassing the limitations of traditional solutions such as call centers.

Call centers have been a widely used solution for providing support to users in various organizations, including academic institutions, corporations, and public services. However, this model, while functional in certain contexts, faces multiple challenges that limit its effectiveness, especially when used as a help desk to address questions about institutional information or complex processes. These challenges range from inaccuracies in the information provided to high operational costs, highlighting the need to explore more efficient alternatives adapted to current demands.

Although information access to public data is not a limitation [2] within an enterprise like a university, one of the main issues with call centers is the inaccuracy of the information offered. This occurs because the personnel operating these platforms often lack comprehensive and up-to-date knowledge about all aspects of the organization. The information available to the operators depends largely on databases or manuals that are often outdated. This lack of accuracy frustrates users, who expect quick and correct responses, and can also compromise trust in the institution, especially if errors persist or recur.

Additionally, the call center model heavily relies on human factors, which come with both advantages and disadvantages. On the one hand, human contact can be beneficial for certain complex interactions that require empathy or contextual judgment. However, in terms of efficiency and consistency, this dependency introduces additional problems. Training personnel is costly and must be recurrent to ensure that operators are up to date with changes in processes, regulations, or organizational services. Even with training, operators can make mistakes due to time pressure or the repetitive nature of tasks, ultimately negatively impacting the user experience.

Another significant challenge is the operational cost of call centers. These costs include salaries, benefits, training, technological infrastructure, and maintenance. Furthermore, the need to maintain an adequate number of operators to cover service hours increases these expenses, especially if extended support hours are required. For organizations with limited resources, such as some universities, these costs can be prohibitive.

Time is also a weakness of call centers. Although they are designed to provide quick responses, users often face long waits due to saturated lines or the need to transfer calls between operators to find someone with the appropriate knowledge. These delays not only affect user satisfaction but also create a negative perception of the organization's responsiveness.

In terms of customer experience, call centers also face issues related to personalization. Users often receive generic responses that do not fully address their specific concerns. This happens because operators are typically trained to handle a high volume of interactions rather than to dedicate time to each individual user. This lack of personalization can be particularly problematic in contexts where questions may be highly specialized, such as information about academic programs, enrollment processes, or specific scholarship requirements.

In contrast, Aurel_AI proposes a solution based on natural language processing algorithms and machine learning. This tool would be capable of providing accurate real-time responses, minimizing errors and reducing operational costs. Its potential to integrate with other institutional information systems would ensure that the data provided to users is always up-to-date and aligned with the university's internal regulations and dynamics. Furthermore, Aurel_AI has the advantage of being able to operate 24/7, offering uninterrupted support and eliminating the time barriers that have traditionally limited customer service models like call centers.

The implementation of a system like Aurel_AI would not only benefit users but also strengthen the university's institutional image by positioning it as an innovative entity committed to service excellence. The trust generated by a reliable information system would positively impact user experience, encouraging a closer and more efficient relationship between the institution and the community.

Overall, the objective of this research project is to develop an automated system functioning as a help desk or chatbot, providing specific information about the university as accurately as possible. Specific objectives include building a methodology for collecting, cleaning, and updating information for the system; finding the appropriate combination of model and adaptation technique to deliver the most accurate responses possible; and developing a flexible and adaptable system that provides an acceptable user experience for both the administrative team and end users.

2. CONCEPTUAL BACKGROUND

From the earliest steps taken by Alan Turing in conceptualizing machines that mimic human thought [3], cognitive sciences have inspired significant advances in the creation of systems capable of analyzing, reasoning, and making decisions [4]. The central goal has been to replicate and enhance the human brain's ability to handle large volumes of information efficiently, answering to complex problems with precise solutions. In this quest, two initial approaches emerged which, although useful in their contexts, presented important limitations.

The first approach, based on expert systems, consisted of building inference machines that operated on a previously filled knowledge base. These systems functioned like large libraries of rules and facts, capable of making reasoned decisions based on the information available. While

revolutionary at the time, these systems faced inherent design problems: their effectiveness depended on the quality and quantity of manually encoded knowledge, making them rigid and difficult to scale. Moreover, as the volume of data grew exponentially, they became unable to efficiently process information in real-time.

The second approach sought to simplify the available information, highlighting only the most relevant elements for decision-making. This method, inspired by human cognitive strategies for filtering data [5], offered a more agile solution but at the cost of precision. By prioritizing certain data over others, there was a risk of overlooking critical information, which could lead to significant errors in the results. Additionally, this approach was not suitable for highly dynamic problems where seemingly irrelevant details could become crucial over time.

With the advent of modern artificial intelligence models, especially Large Language Models (LLM) [6], these two approaches appear to have found a point of convergence. LLMs, such as Chat GPT, represent a technological evolution that combines the strengths of both previous paradigms. These models are not only capable of processing vast amounts of information in real-time but also feature advanced contextual understanding mechanisms, allowing them to identify meaningful patterns and relationships without losing sight of important details.

LLMs utilize deep learning techniques based on neural networks. Unlike expert systems, they do not rely on a predefined set of rules but learn and generalize from large datasets during training. This allows them to adapt to a wide variety of contexts and problems without requiring manual reprogramming. Additionally, by integrating the ability to identify relevant information within massive datasets, they overcome the limitations of simplification methods, reducing the risk of inaccuracy.

Another fundamental advantage of LLMs is their ability to handle the exponential growth of data, a challenge traditional approaches could not effectively address. These models are designed to scale efficiently, taking profit of advances in computational hardware and optimization techniques. Their ability to process information in parallel and contextually enables them to quickly answer complex questions, generating precise and comprehensive answers even in scenarios of high uncertainty.

Furthermore, LLMs have proven to be valuable tools not only for decision-making but also for advanced creative and cognitive tasks. Their ability to generate natural language, summarize information, translate languages, and perform predictive analysis positions them as versatile solutions across a wide range of fields, from education to medicine and scientific research.

LLMs represent one of the most significant advances in modern artificial intelligence, specializing in understanding and interpreting human language. These models have revolutionized how we interact with machines, particularly through applications such as chatbots, virtual assistants, and content generators. A crucial aspect of LLMs is their ability to work with generative artificial intelligence models [7], which can produce coherent and plausible text based on an initial user-provided input. In this regard, LLMs demonstrate a remarkable capacity to interpret questions or text inputs even if they are poorly structured. This is particularly useful in applications where users may express their questions informally, such as in customer service chatbots or interactive educational systems. For example, an LLM can understand a poorly phrased question and provide a coherent answer, lowering the entry barrier for users.

Despite their capabilities, LLMs are not without limitations. One of the most notable issues is the phenomenon known as "hallucinations", where the model generates incorrect or fabricated answers. This typically occurs when the model lacks sufficient relevant information in its training data to address a specific question. For instance, if a user asks about a highly specialized topic that was not well-represented in the initial dataset, the model might provide an apparently plausible response, but one whose content is incorrect.

This problem is exacerbated by the confidence with which LLMs present their responses. For the average user, distinguishing between a precise answer and a hallucination can be challenging, potentially leading to misunderstandings or decisions based on erroneous information. This underscores the importance of implementing validation strategies and human oversight in critical applications of LLMs, particularly in sectors such as healthcare, law, and education, where precision is essential.

The functioning of LLMs is based on a fundamental principle: predicting the next word or token in a given text sequence. This process is made possible by massive training with enormous volumes of text gathered from the internet and other data sources. The primary goal is for the models to develop a statistical understanding of how words and phrases relate to each other in different contexts. Thus, when a user poses a question or provides an initial text, the model can generate a continuation that is coherent and relevant.

The precision and versatility of LLMs largely depend on their ability to interpret the context of the text they process. This is where Transformer Models [8] come into play, a subtype of models that have radically changed the field of natural language processing.

Transformer Models are a neural network architecture designed to efficiently and effectively handle sequential data, such as natural language. Introduced in the paper "Attention is All You Need" [9], transformers have revolutionized natural language processing and other areas of artificial intelligence by overcoming the limitations of previous architectures, such as recurrent neural networks (RNNs) [10] and convolutional neural networks (CNNs) [11], especially in tasks that require understanding complex and long-term relationships between elements of a sequence.

The functioning of transformers is based on two key principles: the attention mechanism and processing parallelization. Both combine to provide a more efficient way to model relationships between different parts of a

sequence. The core of transformers is the Self-Attention mechanism [12], which allows models to focus on different parts of the input sequence while processing a particular element. This mechanism calculates the relative importance of each word or token concerning the others in the same sequence, assigning weights that indicate how they influence one another. To calculate attention, three main vectors are used: Query, representing the word currently being processed; Key, representing the reference words in the sequence; and Value, containing the contextual information associated with each word. Attention is measured by calculating the similarity between the Query and Key vectors, obtaining a score that is used to weigh the corresponding Values. This process allows each word in a sequence to dynamically attend to others, capturing complex contextual relationships. Multi-Head Attention [13] further enhances this mechanism, enabling the model to capture different types of relationships simultaneously, increasing its ability to understand varied contexts.

Unlike recurrent networks, which process data sequentially and face scalability challenges, transformers process all elements of a sequence simultaneously. This approach is achieved through positional representations that indicate the relative order of tokens, as transformers lack inherent sequential structure. The resulting parallelization not only significantly accelerates training but also enables the handling of larger datasets, facilitating the creation of large-scale models such as GPT [14] and BERT [15].

The architecture of transformers includes two main components: the encoder and the decoder. The encoder takes an input sequence and generates a contextualized representation of it, useful for tasks like text classification or machine translation. The decoder generates an output based on the representation created by the encoder, being particularly valuable in generative tasks such as text creation. Transformers excel in their ability to capture long-term relationships and process complete sequences in parallel. These characteristics make them more scalable and effective than previous architectures.

LLMs are highly versatile and can be adapted to perform a variety of specific functions through an adaptation process. In this process, the base model—trained on general data—is customized using additional datasets designed for specific use cases. For instance, "chat" or "instruct" models are specifically designed to maintain fluid conversations and generate clear answers to user questions. Similarly, LLMs can be customized for creative tasks, such as writing in a specific literary style, or for business applications, like providing specialized answers based on a company's internal information. For example, a base model can be tailored to answer specific questions about products, internal policies, or operational procedures unique to an institution. This adjustment allows the model to provide responses that are not only accurate but also aligned with the organization's context and specific needs. This approach has been used to develop specialized virtual assistants, content generation tools for marketing, and even automated technical support systems.

The development of adapters for artificial intelligence models, particularly LLMs, has advanced significantly, offering increasingly efficient and practical solutions to customize these models for specific needs. Currently, there are two fundamental techniques for this purpose: fine-tuning [16] and Retrieval Augmented Generation (RAG) [17]. These strategies allow a base model, previously trained on large volumes of general data, to be adjusted for more precise responses to specific tasks or domains.

Fine-tuning, in its purest form, involves retraining all the weights of the original model by feeding it new information, enabling it to learn additional relationships between tokens or words. Technically, this means the model adjusts the values that define connections between its layers, modifying its understanding of language and its ability to generate responses. However, this approach is highly demanding in terms of computational resources, requiring significant processing power and memory, as it involves working with billions of parameters. For this reason, full fine-tuning is challenging to implement in environments where access to specialized hardware, such as high-end graphics processing units (GPUs), is limited.

In response to these limitations, an alternative technique known as Low Rank Adaptation (LoRA) [18] has emerged, which significantly simplifies the model adjustment process. LoRA introduces an additional matrix of weights superimposed on the original model, and it is this matrix, rather than the complete model, that is trained. This approach is notably more efficient because the additional matrix is relatively small compared to the entire structure of the base model. Despite its simplicity, LoRA has proven to be nearly as effective as full fine-tuning in a variety of applications, making it a viable and attractive option for adapting models in resource-constrained scenarios.

One of LoRA's main advantages is its ability to reduce memory demands during the adaptation process. This feature has been further enhanced with variations such as Quantized LoRA (QLoRA) [19], which uses quantization techniques to further decrease hardware requirements without significantly compromising model performance. QLoRA is particularly useful for adapting large language models on relatively common hardware devices, democratizing access to the customization of advanced models. This innovation paves the way for its implementation in academic projects, small businesses, and other ventures where budgets or infrastructure are limited.

RAG represents an increasingly popular approach. RAG combines the generative capabilities of LLMs with external databases that enrich the context of responses. Instead of retraining the model, this technique allows it to query additional information in real-time to generate more precise and relevant results. This avoids the costs and time associated with fine-tuning or LoRA, as it does not require directly modifying the model's weights. However, RAG presents its own challenges, such as the need to design highly organized databases and efficient information retrieval systems.

RAG is a technique that enhances the capabilities of LLMs by integrating them with external sources of information

without modifying the base model. Essentially, RAG improves the quality of generated responses by constructing enriched prompts that include additional context relevant to the user's question. This context is obtained through a prior process of searching and retrieving information, which uses external sources such as documents, websites, or specialized databases. The LLM receives this expanded prompt and processes it to summarize, structure, and present a coherent and well-informed response.

The key to RAG lies in its ability to complement the inherent limitations of LLMs, such as the lack of updated or specialized information not included in the training data. Instead of retraining the base model or adapting it through techniques like fine-tuning, RAG delegates the task of contextualization to an external infrastructure that gathers and organizes relevant information. In this way, the base model acts as a synthesizer of information, transforming previously retrieved data into a precise and structured response.

For RAG to function effectively, an infrastructure beyond the LLM is required. First, it is necessary to have additional sources of information that can be quickly queried. These sources may include organizational databases, document libraries, or even public websites. However, the challenge lies not only in accessing these sources but also in identifying and extracting only the information relevant to the user's query. To achieve this, vector databases [20] such as Faiss [21] are used, which efficiently organize and search information through vector indices.

A vector database is a storage and retrieval system specifically designed to handle and organize data in the form of vectors in a multidimensional space. In this context, vectors are mathematical representations of data, generated through machine learning techniques or data processing, encapsulating relevant information about the represented elements, such as text, images, audio, or video. Unlike traditional databases, which typically work with structured data like rows and columns, vector databases are optimized for search and retrieval operations based on vector similarities.

The primary purpose of a vector database is to facilitate the search for similar elements within a dataset using proximity metrics such as Euclidean distance, cosine similarity, or Manhattan distance. These metrics measure how close or related the vector representations of different elements are. For example, in a system that stores vector representations of words, words with similar meanings can be searched based on the proximity of their vectors in multidimensional space.

The vector representation of data is generated by machine learning models known as embedding models, which convert complex data (such as a sentence) into fixed-dimensional vectors. These vectors preserve semantic and contextual relationships between data, allowing related elements to be close to each other in the vector space. This principle is particularly useful in tasks such as information retrieval, image analysis, and product recommendation, where it is essential to compare elements based on their content or context.

A vector database is organized around indexes that facilitate the quick search of similar vectors. The vector index is an optimized data structure that significantly reduces search time in large datasets. Without these indexes, finding similar vectors would require calculating the distances between all stored vectors and the query vector, which would become computationally expensive as the amount of data increases.

Vector indexes employ advanced optimization techniques, such as Approximate Nearest Neighbor (ANN) search algorithms [22], which quickly find the vectors most like a query without requiring an exhaustive search. Among the most well-known implementations of these algorithms are Faiss (Facebook AI Similarity Search), ScaNN (Scalable Nearest Neighbor) [23], and Milvus [24], which offer specialized tools for handling large volumes of vector data in practical applications.

The typical workflow of using a vector database begins with creating embeddings of the data to be stored. For instance, in a text search application, each document or text fragment is processed by an embedding model that generates a vector representing its content. These vectors are stored in the vector database along with an identifier that links them to the original data. When a user submits a query, it is also converted into a vector using the same embedding model. The vector database compares this query vector with the stored vectors to identify the most similar ones, returning the corresponding elements.

Vector databases have proven to be particularly useful in artificial intelligence and machine learning applications. For example, in recommendation systems, they enable the comparison of user profiles or product features to suggest relevant items. In image recognition, they can store vector representations of images and quickly retrieve those visually like a query. In natural language processing, they are used to build systems such as Retrieval Augmented Generation (RAG), where vector representations of documents allow the retrieval of relevant information based on user-generated queries.

However, working with vector databases also presents technical challenges. One of the main issues is the dimensionality of the data. High-dimensional vectors, while expressive, can be costly to store and process, leading to problems known as the "curse of dimensionality." This phenomenon implies that, as dimensionality increases, the distances between vectors become less meaningful, potentially complicating the accurate identification of nearest neighbors. To mitigate this problem, techniques such as dimensionality reduction and optimization of vector indexes are employed.

Another challenge is scalability, especially in applications that manage billions of vectors. Implementations like Faiss address this issue by offering support for distributed searches and optimization for specialized hardware, such as GPUs, which allow large volumes of data to be handled efficiently. Additionally, modern vector databases often integrate mechanisms for managing real-time data updates,

ensuring that the system can dynamically adapt to changes in the stored data.

Vector databases inherit the limitations and biases of the embedding models that generate the vector representations. If an embedding model is trained on biased data, searches conducted in the vector database may perpetuate these biases, affecting the fairness and objectivity of applications. Therefore, it is essential to consider both the quality of the training data and the interpretation of the results.

An embedding model is a type of machine learning model designed to represent data—such as words, phrases, documents, or any type of textual or non-textual information—in a fixed-dimensional mathematical vector space. In this space, each data element is converted into a numerical vector, and the semantic relationships between elements are reflected in the distances or similarities between their respective vectors. This technique has revolutionized multiple fields, including natural language processing, information retrieval, and other domains, by providing an efficient and effective way to capture complex and contextual relationships between data.

The underlying concept of an embedding model lies in its ability to transform input data, which is often categorical or sequential, into a dense and continuous representation that captures the essential features of the content. For example, in the case of words in text, an embedding model like Word2Vec or GloVe assigns each word a numerical vector of reduced dimensions, typically ranging from 50 to 300 dimensions [25] [26]. This vector not only represents the word in isolation but also encapsulates information about its context, such as its meaning and its relationship to other words.

The way embedding models are trained varies, but their common goal is to learn representations that preserve the semantics of the data. In natural language processing, this is achieved using methods such as supervised learning, unsupervised learning, or self-supervised learning. For instance, in the Word2Vec model, two main techniques are employed: the CBOW (Continuous Bag of Words) model [27], which predicts a word based on its surrounding context, and the Skip-gram model [33], which predicts the context given a word. Both methods aim to optimize a loss function that measures the model's ability to capture contextual relationships in the training data. As the model's parameters are fine-tuned, the vectors representing words begin to organize themselves in the vector space, so that semantically similar words are positioned closer to each other.

One of the most powerful properties of embeddings is their ability to capture complex semantic and syntactic relationships. For example, in a word vector space trained with Word2Vec, the relationship between "King" and "Queen" can be mathematically represented as a vector similar to that linking "Man" and "Woman." This emergent property, known as semantic compositionality, enables embedding models to generalize the knowledge learned to new contexts, making them extremely useful in tasks such as machine translation, text classification, and information retrieval.

Embedding models are not limited to natural language processing. In information retrieval applications, such as RAG, embeddings are used to represent documents, text fragments, or even user queries as vectors. A key example of this is the use of vector databases, where indexed data is stored in the form of embeddings. When a user poses a query, it is also vectorized using the same embedding model, and the similarities between the query vector and the stored vectors are calculated to retrieve the most relevant data.

Embeddings have also evolved to handle more complex and contextual data. Advanced models like BERT and GPT generate contextual embeddings, where the vector representations of a word or phrase vary depending on its position and meaning in a specific context. This represents a significant improvement over traditional models, which assigned a single fixed vector to each word regardless of its use in different contexts.

In addition to their practical utility, embedding models present technical and ethical challenges. The dimensionality of vectors can be a critical factor: higher-dimensional representations tend to be more expressive but also more costly in terms of storage and computation. Furthermore, since embeddings are trained on large datasets, they may reflect and perpetuate biases present in that data. For example, if an embedding model is trained on texts containing gender or racial biases, these biases may manifest in the vector representations, raising significant ethical concerns.

Once relevant information is retrieved, it is incorporated into the original prompt as additional context. The LLM processes this enriched prompt to generate a response that combines the model's generative capabilities with the specific information provided by the retrieval system. This approach ensures that responses are not only contextually appropriate but also based on up-to-date or specialized data, which is especially useful in environments where precision is critical, such as medicine, law, or scientific research.

However, using RAG is not without challenges. The quality of the system largely depends on the precision and relevance of the retrieved information. If the embedding model or the vector database fails to correctly identify the relevant fragments, the LLM might generate incorrect or unhelpful responses. Additionally, implementing RAG requires a more complex technical infrastructure, which can be a barrier for organizations with limited resources. The process of vectorizing external information and the user's query creates a workflow as described in *Figure 1*, culminating in the generation of the final response. As an example of some of the most relevant recent works in this field, we can highlight JayBot [28], which uses ChatGPT to provide a support system for a university's admissions process. In [29], the potential of such systems in public enterprises is explored to improve interactions between governments and citizens. In [30], Lakkaraju et al. delve into the trust that can be placed in these systems within a financial context. The issue of information reliability and its consequences is further analyzed in [31]. Finally, Sahaay [32] is worth mentioning, a framework proposing

an ecosystem to improve customer service using LLM-based chatbots.

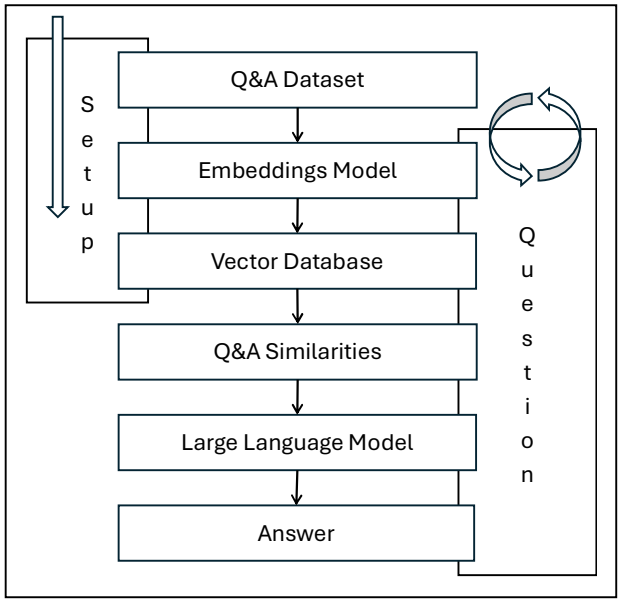

Figure 1. Information processing flow in a RAG framework. First, there is a one time setup process where external information inside a Q&A Dataset is vectorized using an embedding model and stored in a vector database. Then, there is an iterative process, driven by each question: the user's question is vectorized using the same embedding model and sent to the vector database, where the information most similar to the question is retrieved; a contextualized prompt is built with this information and sent to the main model as context; finally, the LLM generates the answer for the user.

3. METHODOLOGY

The project began with the need to evaluate and select the appropriate equipment to implement a system based on LLMs capable of managing sensitive university information. This requirement involved initial tests with different LLMs to determine the minimum hardware capacities needed. One of the first conclusions was that hardware requirements, especially in terms of GPU memory (VRAM), were significantly high. This led the team to seek a balance between cost and capacity, dismissing the use of cloud services due to the sensitivity of the data being handled.

A system equipped with an Intel Core i9 processor, 64GB of RAM, and a NVidia RTX 4090 GPU with 24GB of VRAM was chosen. This last component proved critical for efficiently processing models of moderate size. With this configuration, it was possible to work with models ranging from 7 billion (7B) to 13 billion (13B) parameters but not with larger ones, such as those between 60–80B, which would have required more advanced and costly hardware. The models selected for the initial tests included Llama2 in 7B and 13B versions, Falcon 7B, and Mistral 7B, both in their foundational variants and their chat or instruct adaptations.

The next step involved collecting and organizing the information needed to feed the system. This process was carried out in collaboration with PUCE's Academic Directorate, which required a structured communication strategy with the university's departments and faculties. Group meetings were organized to present the general concept of the project and request their collaboration in gathering data in a question-and-answer (Q&A) format. This approach allowed the capture of the most frequent inquiries that end-users typically make, based on the experience of the different departments.

However, the information initially collected was scattered and lacked a uniform structure. To address this problem, the development team prepared a set of generic questions that were distributed to all departments, asking them to adjust and answer these questions according to the specificities of their area. This structured approach resulted in a final database of over 2,000 Q&A entries, which served as a fundamental input for building the system's adapters.

In the adapter development stage, two main techniques were explored: fine-tuning and Retrieval Augmented Generation (RAG). Fine-tuning, which allows a base model to be adjusted for specific tasks, presented significant challenges due to the memory limitations of the available hardware. To overcome this obstacle, Parameter-Efficient Fine-Tuning (PEFT) techniques were implemented, such as LoRA (Low Rank Adaptation) and QLoRA (Quantized LoRA). These techniques enabled the use of additional weight matrices on the original model, reducing computational load and effectively adjusting the models within hardware constraints.

In the case of RAG, the main challenge was selecting a suitable embedding model to vectorize the data and an efficient vector database for storing and retrieving information. The embedding model hiiamsid/sentence_similarity_spanish_es was chosen for its solid performance in Spanish, along with the Faiss vector database, known for its efficiency and flexibility. This approach allowed the organization of information in a way that could be quickly retrieved based on user queries.

The system's development involved the use of various software libraries to manage technical complexities. Among them, torch and transformers were used for general operations related to LLMs; bitsandbytes, accelerate, and peft facilitated model reduction to fit the available hardware; and langchain, sentence-transformers, and faiss-gpu provided support for RAG implementation. Each combination of model and adaptation technique was systematically tested to build specific adapters.

Once the adapters were developed, a qualitative evaluation phase was conducted with the internal development team. This process involved testing the adapters with a subset of questions selected from the Q&A database and evaluating the accuracy of the responses on two levels. First, the correspondence of the responses with the data retrieved from the vector database was examined. Second, the quality of the final response generated by the model was analyzed, ensuring that it was coherent, relevant, and complete.

Based on the results of this initial evaluation, the most promising adapter was selected for a broader evaluation by users from different departments and faculties of the university. This step was crucial, as it involved those who

initially provided the base Q&A entries. Evaluators were asked to focus on identifying problematic responses, recording both the model-generated response and the response they considered appropriate. This detailed feedback was collected through a form designed to capture concrete examples of discrepancies, providing the development team with valuable information for future system iterations.

The described process not only allowed for the construction of a system tailored to the university's specific needs but also highlighted the importance of an iterative approach in developing artificial intelligence-based systems. Lessons learned during this phase, such as the importance of a robust infrastructure, the need for well-organized data, and the careful selection of adaptation techniques, laid the groundwork for future improvements and project expansions.

4. RESULTS AND DISCUSSION

The first element to highlight is that between fine-tuning and RAG, RAG is definitively the better choice. All tests conducted with fine-tuning, regardless of the model or internal technique, demonstrated noticeable inaccuracies. This is attributed, as observed, to the adapter's inability to incorporate new information into the model. As a result, the responses tend to be predominantly hallucinatory and contain information clearly derived from other sources used during the base training.

Fine-tuning experiments were conducted with 20, 40, 60, and 80 epochs on the data. While more repetitions seemed to adjust the responses better to the provided information, this ultimately resulted in mere stylistic refinements. Fundamentally, the responses remained hallucinatory in terms of informational quality. It is worth noting that while chat or instruct models tend to provide better-structured responses, this improvement is limited to their general format and does not extend to the content itself. Regarding the different models—Llama2, Falcon, or Mistral—no significant differences were observed. Additionally, fine-tuning required considerable time, taking approximately 14 hours with the largest models (13B) on the described setup.

When applying RAG, the first significant advantage is that it does not require training time. Reconfiguring the application with a new model is sufficient to test it almost immediately (in just a few minutes). At this point, the models were divided into two categories. For foundational models, RAG produced poorly structured responses that were ultimately not usable in a production system. However, with chat or instruct models, the situation changed radically.

In general, all these models delivered acceptable responses, clearly incorporating the new contextual information provided. However, some hallucinatory responses still occurred, often when the Q&A retrieved from the database was not closely related to the user's question. Another cause observed in some cases was that the information contained in the Q&A was inconclusive (e.g., not explicitly affirmative or negative), leading the model to generate an erroneous interpretation.

Another influential factor appears to be the embedding model used to vectorize the information, including the Q&A stored in the database and the user's queries. Certain tests conducted with a model different from the one used in this study, sentence-transformers/distiluse-base-multilingual-cased-v1, revealed differences that seemed to favor this latter model.

After preliminary tests by the research team, it was decided to use the Llama2 7B model for evaluation by the internal test user group. While it could be argued that the 13B model is superior, the differences were not substantial. Furthermore, the 13B model required more time to generate responses, which in some cases was noticeable to the user. To improve user experience concerning response time, the 7B model was preferred.

Regarding resource usage, once the selected model (Llama2 7B), reduced (quantized) to 4 bits, was loaded, it utilized only 6GB of GPU VRAM, which remained stable while the application was active. However, each user query consumed 100% of the GPU until the response was generated, creating a bottleneck when handling concurrent queries. The CPU and its RAM usage were insignificant.

The application's response times, excluding data request/response transmission, varied significantly and were primarily affected by simultaneous queries that became queued and congested the GPU. *Table 1* provides a breakdown of average times by deciles. Eighty percent of queries (deciles 1 to 8) took less than 20 seconds. Extreme cases in decile 10, with an average time of 4 minutes, were due to numerous simultaneous queries, often caused by the same user sending identical requests in bursts or rapid succession.

Table 1. Average response times by deciles.

Decil	***Avg.Req.Time***
1	0,95
2	2,78
3	3,74
4	5,18
5	6,92
6	8,87
7	13,17
8	19,97
9	59,99
10	237,77
	35,93

This information was gathered from a total of 147 sessions or connections to the system, during which 329 questions were asked, averaging 2.24 questions per session.

The most critical point, however, concerns the accuracy of the system's responses. One of the final steps in this initial phase of the project was to ask users to report any issues they encountered. Although participation in this task was not very high, 122 incidents were recorded, corresponding to approximately one-third of the queries made.

To better understand the causes of these problems and attempt to address them in the next phase of the project, a sample of these incidents was reviewed and compared against the Q&A information in the database. An initial categorization was attempted, and it was preliminarily decided to create four categories to classify the incidents: (1) lack of information in the Q&A database, (2) incorrect or imprecise information in the Q&A database, (3) questions unrelated to PUCE (Pontificia Universidad Católica del Ecuador), and (4) "pure" hallucinations by the model.

Of these four categories, it is anticipated that the majority of incidents will fall into the first two categories, representing the easiest cases to resolve, primarily requiring database cleaning. Cases falling into the third category may be more challenging to detect, but their resolution would be straightforward, as it would only require informing users that the system provides information exclusively related to PUCE. The final category will be the most complex, as it implies that, even though the required information is available in the database, the model chooses to follow a different path to generate the response.

Finally, the system is currently running as a web application developed in Flask, which has not yet been refined or optimized for production environments. There is confidence that there are several opportunities to improve response times, particularly in multi-user contexts.

5. CONCLUSIONS AND FUTURE WORK

The development and implementation of a system based on large language models (LLMs) to manage institutional information for PUCE yielded valuable results that provide fundamental conclusions and clear directions for future work. These conclusions highlight both the strengths and limitations of the techniques used, offering a framework for optimizing the system in subsequent phases.

Firstly, the results demonstrated that Retrieval Augmented Generation (RAG) is superior to fine-tuning for this type of application. Fine-tuning, while conceptually appealing for model customization, exhibited significant inaccuracies that compromised the quality of the generated responses. These shortcomings stem from fine-tuning's inherent difficulty in incorporating new information without significantly altering the model's pre-existing knowledge. The resulting responses, often hallucinatory, did not meet the quality standards required for a production system. Additionally, the lengthy training times—up to 14 hours for 13B models—and the lack of significant differences among the models tested (Llama2, Falcon, and Mistral) underscored the disadvantages of this technique.

In contrast, RAG proved to be a more efficient and accurate solution. Its main advantage lies in the ability to integrate new information immediately by enriching prompts with external data stored in vector databases. This approach eliminates the need for retraining, significantly reducing the time required to test or update the system with a new model. Although RAG also presents some issues, such as occasional hallucinations and errors when queries are insufficiently related to the stored Q&A, these incidents are more manageable compared to the problems observed with fine-tuning.

Regarding technical performance, chat or instruct models proved more effective in the RAG context, delivering better-structured and more aligned responses to user queries. However, the embedding model used to vectorize information directly affected response quality. Tests conducted with sentence-transformers/distiluse-base-multilingual-cased-v suggested that this model could be more effective than hiiamsid/sentence_similarity_spanish_es, opening a line of research to optimize embedding selection in future project iterations.

The system's performance analysis, based on tests with the Llama2 7B model reduced to 4 bits, identified both strengths and areas for improvement. On the one hand, the model's initial load onto the GPU was efficient, using only 6GB of VRAM and maintaining stable consumption during operation. However, concurrent queries created a significant bottleneck, with response times in the worst-case scenarios (10th decile) reaching up to 4 minutes due to GPU congestion. This issue directly impacts user experience and highlights the need to optimize resource management and infrastructure to support a multi-user environment.

Another critical aspect identified was the accuracy of the system's generated responses. The collection of incidents, although limited to 122 reports, enabled the classification of problems into four main categories: lack of information in the Q&A database, incorrect or imprecise information, questions outside the system's scope, and pure hallucinations by the model. This preliminary categorization is a crucial step in guiding system improvements. The first two categories, related to the quality and coverage of the Q&A database, represent the most immediate areas of intervention, as their resolution primarily involves reviewing and cleaning the data. Incidents related to questions outside the system's scope can be addressed by implementing mechanisms to clearly inform users about the system's limitations. However, pure hallucinations by the model present a more complex challenge, as they require deeper adjustments to the model's integration and contextual information handling.

In terms of infrastructure, the current system runs as a web application based on Flask, offering a functional but not yet optimized foundation for a production environment. Improving this application is a priority to ensure its scalability, stability, and usability in real-world contexts. It is essential to implement solutions to efficiently manage simultaneous queries, such as load distribution across multiple GPUs or the use of message queues to handle user requests.

Future work includes several key actions to improve the system. One of the first tasks will be to clean and enrich the Q&A database to ensure it adequately addresses users' most frequent and relevant queries. This involves correcting errors, filling in missing information, and structuring the data uniformly. The implementation of automated tools for periodic validation and database updates could be

considered. Given the significant impact of the embedding model on response quality, it will be crucial to conduct extensive tests with different models to identify the most suitable option. Additionally, fine-tuning an embedding model specifically for PUCE's data and context could be explored.

To address the bottleneck caused by simultaneous queries, strategies such as deploying additional GPU servers, distributing the load across multiple system instances, and employing techniques like asynchronous response generation will be evaluated. Although reduced in RAG, the phenomenon of model hallucinations remains a critical challenge. Efforts will be made to mitigate these incidents by adjusting the prompt generation pipeline, improving data curation in the vector database, and potentially modifying the LLM itself.

The Flask-based application must evolve into a more robust, intuitive interface capable of handling multiple users simultaneously. This includes improvements in user experience, such as faster loading times and clear error messages for out-of-scope queries. A continuous monitoring system that automatically logs reported incidents and system performance metrics is also necessary. This monitoring will provide valuable data for future iterations and enable more agile problem diagnostics as they arise.

Finally, once the initial improvements are implemented, the system should undergo testing in a limited production environment to evaluate its performance in real-world scenarios and gather feedback from end users. These actions will help consolidate the system's effectiveness and reliability in a practical and scalable setting.

6. REFERENCES

[1] M. Abbasi, M. Maks Davis, R. Melgarejo Heredia and D. Ordóñez Camacho, "Artificial Intelligence: A Look Back To The Future In University Education," in **Proceedings of International Structural Engineering and Construction**, Quito, Ecuador: ISEC Press, 2024. doi: 10.14455/ISEC.2024.11(1).EPE-11.

[2] Rafael Melgarejo-Heredia, Leslie Carr, and Susan Halford. 2016. The public web and the public good. In **Proceedings of the 8th ACM Conference on Web Science** (WebSci '16). Association for Computing Machinery, New York, NY, USA, 330–332. https://doi.org/10.1145/2908131.2908181

[3] A. Turing, "Computing Machinery and Intelligence," **Mind**, vol. 59, no. October, pp. 433–60, 1950, doi: 10.1093/mind/lix.236.433.

[4] R. I. O. Martínez and M. N. B. Tello, "La mente computacional. Orígenes y fundamentos de la ciencia cognitiva.," **Protrepsis**, no. 6, Art. no. 6, 2014, doi: 10.32870/prot.i6.92.

[5] C. B. Chadwick, "Estrategias cognoscitivas y afectivas de aprendizaje. Parte (A)," **Revista latinoamericana de psicología**, vol. 20, no. 2, pp. 163–184, 1988.

[6] S. Bubeck et al., "Sparks of Artificial General Intelligence: Early experiments with GPT-4." **arXiv**, Apr. 13, 2023. doi: 10.48550/arXiv.2303.12712.

[7] C. Stokel-Walker and R. Van Noorden, "What ChatGPT and generative AI mean for science," **Nature**, vol. 614, no. 7947, pp. 214–216, Feb. 2023, doi: 10.1038/d41586-023-00340-6.

[8] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing." **arXiv**, Aug. 28, 2021. doi: 10.48550/arXiv.2108.05542.

[9] A. Vaswani et al., "Attention Is All You Need," Aug. 01, 2023, **arXiv**: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[10] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in **International conference on machine learning**, PMLR, 2015, pp. 2342–2350.

[11] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," **IEEE Transactions on Neural Networks and Learning Systems**, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[12] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," Apr. 12, 2018, **arXiv**: arXiv:1803.02155. doi: 10.48550/arXiv.1803.02155.

[13] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "Multi-Head Attention: Collaborate Instead of Concatenate," May 20, 2021, **arXiv**: arXiv:2006.16362. doi: 10.48550/arXiv.2006.16362.

[14] L. Floridi and M. Chiriatti, "GPT-3: Its nature, scope, limits, and consequences," **Minds and Machines**, vol. 30, pp. 681–694, 2020.

[15] M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding," Mar. 22, 2021, **arXiv**: arXiv:2103.11943. doi: 10.48550/arXiv.2103.11943.

[16] Z. Hu et al., "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models." **arXiv**, Oct. 09, 2023. doi: 10.48550/arXiv.2304.01933.

[17] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A Survey on Retrieval-Augmented Text Generation." **arXiv**, Feb. 13, 2022. doi: 10.48550/arXiv.2202.01110.

[18] S. Devalal and A. Karthikeyan, "LoRa Technology - An Overview," in **2018 Second International Conference on Electronics, Communication and Aerospace Technology** (ICECA), Mar. 2018, pp. 284–290. doi: 10.1109/ICECA.2018.8474715.

[19] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs." **arXiv**, May 23, 2023. doi: 10.48550/arXiv.2305.14314.

[20] Z. Jing, Y. Su, and Y. Han, "When Large Language Models Meet Vector Databases: A Survey," Nov. 01,

2024, **arXiv**: arXiv:2402.01763. doi: 10.48550/arXiv.2402.01763.
[21] M. Douze et al., "The Faiss library," Sep. 06, 2024, **arXiv**: arXiv:2401.08281. doi: 10.48550/arXiv.2401.08281.
[22] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in **Proceedings of the thirtieth annual ACM symposium on Theory of computing**, in STOC '98. New York, NY, USA: Association for Computing Machinery, May 1998, pp. 604–613. doi: 10.1145/276698.276876.
[23] P. Sun, "Announcing ScaNN: Efficient Vector Similarity Search." **Google Research**. http://research.google/blog/announcing-scann-efficient-vector-similarity-search/
[24] J. Wang et al., "Milvus: A Purpose-Built Vector Data Management System," **in Proceedings of the 2021 International Conference on Management of Data**, in SIGMOD '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 2614–2627. doi: 10.1145/3448016.3457550.
[25] L. Ma and Y. Zhang, "Using Word2Vec to process big text data," in 2015 **IEEE International Conference on Big Data** (Big Data), Oct. 2015, pp. 2895–2897. doi: 10.1109/BigData.2015.7364114.
[26] T. Shi and Z. Liu, "Linking GloVe with word2vec," Nov. 26, 2014, **arXiv**: arXiv:1411.5595. doi: 10.48550/arXiv.1411.5595.
[27] H. Xia, "Continuous-bag-of-words and Skip-gram for word vector training and text classification" **J. Phys.: Conf. Ser**., vol. 2634, no. 1, p. 012052, Nov. 2023, doi: 10.1088/1742-6596/2634/1/012052.
[28] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A closer look at skip-gram modelling.," in **LREC**, 2006, pp. 1222–1225.
[29] J. Odede and I. Frommholz, "JayBot -- Aiding University Students and Admission with an LLM-based Chatbot," in Proceedings of the 2024 Conference on Human Information Interaction and Retrieval, in CHIIR '24. New York, NY, USA: **Association for Computing Machinery**, Mar. 2024, pp. 391–395. doi: 10.1145/3627508.3638293.
[30] A. Androutsopoulou, N. Karacapilidis, E. Loukis, and Y. Charalabidis, "Transforming the communication between citizens and government through AI-guided chatbots," **Government Information Quarterly**, vol. 36, no. 2, pp. 358–367, Apr. 2019, doi: 10.1016/j.giq.2018.10.001.
[31] K. Lakkaraju, S. E. Jones, S. K. R. Vuruma, V. Pallagani, B. C. Muppasani, and B. Srivastava, "LLMs for Financial Advisement: A Fairness and Efficacy Study in Personal Decision Making," in **Proceedings of the Fourth ACM International Conference on AI in Finance**, in ICAIF '23. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 100–107. doi: 10.1145/3604237.3626867.
[32] J. Crowder, "What Happens When a Chatbot Gives Detrimental Advice? Who's Responsible?," in **AI Chatbots: The Good, The Bad, and The Ugly**, J. Crowder, Ed., Cham: Springer Nature Switzerland, 2024, pp. 121–133. doi: 10.1007/978-3-031-45509-4_13.
[33] K. Pandya and M. Holia, "Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations." **arXiv**, Oct. 09, 2023. doi: 10.48550/arXiv.2310.0