Design and Implementation of a Voice-Based Multilingual AI Assistant for Restaurants

Authors: Corentin JUSTE, Vianney LE BOURHIS,
Noé LE YHUELIC, Maxime LANGELIER

Course: LLM and GenAI – ESILV DIA A5

1 Introduction

The restaurant industry faces increasing operational constraints due to high customer demand, limited staff availability, and growing expectations for fast and reliable service. Customer phone interactions such as table reservations, food orders, and general inquiries remain a major operational bottleneck, as they are repetitive, time-consuming, and difficult to manage efficiently during peak hours.

Manual call handling often diverts staff attention away from on-site customers and increases the risk of missed calls, errors, and inconsistent service quality. These issues are further amplified in multilingual or tourist environments, where language barriers can negatively impact both customer satisfaction and operational efficiency.

Recent advances in conversational Artificial Intelligence offer a compelling solution to these challenges. By combining large language models with speech recognition and synthesis technologies, voice-based AI assistants are now capable of understanding natural language, managing multi-turn dialogues, and executing concrete actions through structured back-end systems.

This project aims to design and implement a fully automated, voice-based multilingual AI assistant tailored to restaurant operations. The system enables natural phone-based interactions without requiring any dedicated application, supports multiple languages, and reliably manages reservations and orders through a database-driven architecture. The proposed solution demonstrates how voice-enabled conversational AI can enhance customer experience while reducing staff workload in real-world restaurant environments.

2 Global System Overview

The proposed system is a voice-based conversational AI assistant designed to automate key customer interactions in a restaurant setting. Its primary goal is to handle reservations, food orders, and general inquiries in a reliable and scalable manner, while providing a natural and intuitive user experience. To achieve this, the system combines speech processing, large language models, structured data management, and a modular multi-agent architecture.

At a high level, the system is organized around a central orchestration component that coordinates multiple specialized AI agents. User requests are captured through various interfaces, processed by the orchestrator to determine intent, and delegated to the appropriate agent, which may interact with databases or knowledge sources before generating a response. This modular

design ensures both flexibility and robustness, allowing each component to focus on a well-defined responsibility.

The assistant supports multiple interaction interfaces in order to accommodate different use cases. The primary interface is a phone-based system built on top of the Twilio telephony API, enabling real-time voice conversations with customers. This interface is particularly suited to real-world restaurant scenarios, where phone calls remain the dominant communication channel. In addition, a command-line interface is provided for development, testing, and debugging purposes, allowing developers to interact with the system using text input. Finally, a Streamlit-based web dashboard offers an administrative and analytical interface, enabling restaurant operators to visualize data, monitor activity, and manage reservations and orders.

The core idea behind the system lies in the separation of concerns between intent routing and task execution. A central orchestrator is responsible for analyzing user input, maintaining conversational context, and classifying the user's intent. Based on this classification, the request is routed to one of several specialized AI agents, each designed to handle a specific domain such as table reservations, order management, or general restaurant inquiries. These agents follow a reasoning-and-acting paradigm, allowing them to decide when to query databases, retrieve external knowledge, or ask for missing information before producing a final response.

From an end-to-end perspective, the interaction flow begins with a user request typically a spoken utterance during a phone call. The audio input is first transcribed into text, then processed by the orchestrator to identify intent and context. The relevant agent performs the necessary reasoning and actions, such as checking table availability or retrieving menu information, and produces a textual response. This response is subsequently converted back into speech and delivered to the user. This complete pipeline, from user input to system response, enables seamless and natural interactions while ensuring that all actions are grounded in reliable back-end systems.

3 Multi-Agent Architecture and Orchestration

A central design choice of this project is the adoption of a multi-agent architecture rather than relying on a single, monolithic large language model to handle all user requests. While monolithic conversational models can generate fluent responses, they often struggle to reliably perform structured tasks, interact with databases, and respect domain-specific constraints. In contrast, a multi-agent approach allows responsibilities to be clearly separated, improves system robustness, and makes the overall behavior easier to control, evaluate, and extend.

In the proposed system, each agent is specialized for a specific domain of restaurant operations, such as reservations, order management, or general inquiries. This specialization enables the use of tailored prompts, tools, and reasoning strategies adapted to each task. As a result, the system reduces the risk of incorrect actions, hallucinated information, or inappropriate tool usage, which are common challenges when using a single general-purpose model for complex workflows.

3.1 Central Orchestrator

At the core of the architecture lies a central orchestrator, which acts as the coordination layer between user inputs and the specialized agents. The orchestrator is responsible for three main functions: intent classification, conversational context management, and request routing.

First, intent classification determines the nature of the user's request. Incoming user input after speech transcription and optional translation is analyzed to identify whether it relates to a table reservation, an order, or a general inquiry. This step is crucial, as it ensures that each request is handled by the most appropriate agent, rather than relying on a single agent to infer both intent and execution logic.

Second, the orchestrator manages conversational context across multiple turns. In real-world interactions, users often provide information incrementally, correct previous inputs, or refer implicitly to earlier parts of the conversation. The orchestrator maintains a rolling history of recent exchanges, which is incorporated into each new request sent to the agents. This context management enables coherent multi-turn dialogues while preventing unbounded context growth.

Finally, based on the detected intent and the maintained context, the orchestrator routes the request to the corresponding specialized agent. This routing mechanism enforces a clear separation of concerns and allows agents to remain focused on their respective domains, simplifying both development and debugging.

3.2 Specialized AI Agents

The system is built around three specialized AI agents, each responsible for a well-defined category of user requests.

The *Reservation Agent* handles all interactions related to table bookings. Its responsibilities include checking table availability, creating new reservations, viewing existing bookings, and processing cancellations. This agent interacts directly with the relational database to ensure that reservations are consistent, conflict-free, and persistently stored.

The *Order Agent* manages food ordering workflows, including order creation, item addition or removal, order finalization, and status tracking. By interfacing with structured menu and order data stored in the database, this agent ensures accurate pricing, availability checks, and reliable order management.

The *General Inquiries Agent* is dedicated to answering informational questions about the restaurant, such as opening hours, location, menu details, dietary restrictions, or frequently asked questions. Unlike the other agents, it primarily relies on retrieval-augmented generation (RAG), combining semantic search over a knowledge base with language model reasoning to produce grounded and informative responses.

This division of labor ensures that each agent can be optimized independently and evaluated according to task-specific criteria, while collectively contributing to a coherent conversational experience.

3.3 ReAct Reasoning Pattern

All agents in the system follow the ReAct (Reasoning and Acting) pattern, which structures agent behavior into an explicit sequence of steps: *Thought*, *Action*, *Observation*, and *Final Answer*. This approach encourages the agent to reason about the problem before taking any action and to explicitly process the outcome of each action before proceeding.

In the *Thought* phase, the agent analyzes the user request and determines what information or operation is required. During the *Action* phase, it invokes a specific tool, such as a database query or a retrieval function. The result of this operation is captured in the *Observation* phase, which provides concrete feedback from the environment. Finally, the agent synthesizes this information into a coherent and user-facing response in the *Final Answer* phase.

This structured reasoning process significantly improves reliability compared to unconstrained text generation. By enforcing explicit tool usage and grounding responses in observed data, the ReAct pattern reduces hallucinations, prevents invalid actions, and ensures that responses accurately reflect the system's underlying state. Moreover, it enables better transparency and debuggability, as each intermediate step can be inspected during development and evaluation.

Overall, the combination of a central orchestrator, specialized agents, and the ReAct reasoning framework forms a robust and scalable architecture. This design allows the system to handle complex, real-world restaurant interactions while maintaining reliability, interpretability, and extensibility.

4 Data Layer and Knowledge Management

Reliable data management is a critical requirement for a conversational AI system that performs concrete actions such as creating reservations or processing orders. In the proposed architecture, the data layer is designed to ensure consistency, persistence, and traceability of all operations, while also supporting flexible access to unstructured informational content. To achieve this, the system combines a relational database for structured transactional data with a vector-based retrieval system for knowledge-oriented queries.

4.1 Relational Database: PostgreSQL

The relational database serves as the backbone of the system for all transactional operations related to restaurant management. It stores structured data required for reservations, orders, menu items, and client information. The database schema is designed to reflect real-world restaurant workflows while maintaining clear relationships between entities.

Key entities include *Client*, *Table*, *Reservation*, *MenuItem*, *Order*, and *OrderItem*. Clients can place multiple orders and make multiple reservations, while each reservation is associated with a specific table and time slot. Orders are composed of one or more order items, each linked to a menu item with its corresponding quantity and price. These relationships are enforced through foreign keys, ensuring referential integrity across the system.

To interact with the database, the system relies on the SQLAlchemy Object-Relational Mapping (ORM) framework. The use of an ORM abstracts low-level SQL queries into high-level Python objects, improving code readability, maintainability, and safety. SQLAlchemy enables parameterized queries and structured transactions, reducing the risk of SQL injection vulnerabilities and logic errors. Furthermore, it simplifies schema evolution and facilitates integration with the AI agents by providing a consistent interface for database access.

The relational database guarantees several essential properties. Consistency is ensured through constraints and transactional operations, preventing conflicting reservations or invalid orders. Traceability is achieved by persisting all actions with timestamps and status fields, allowing the system to track the lifecycle of reservations and orders over time. Persistence ensures that all customer interactions resulting in actions are durably stored, making the system robust to failures and enabling later analysis or auditing. Together, these guarantees make the relational database a reliable foundation for all action-oriented components of the assistant.

4.2 Retrieval-Augmented Generation with ChromaDB

While a relational database is well suited for structured transactional data, it is less appropriate for handling open-ended informational queries such as questions about opening hours, dietary restrictions, or general restaurant policies. To address this limitation, the system incorporates a Retrieval-Augmented Generation (RAG) component based on a vector database.

The RAG approach augments the language model's generative capabilities by grounding responses in retrieved knowledge. Instead of relying solely on the model's internal parameters, user queries are first transformed into vector embeddings and compared against a collection of embedded documents. The most relevant pieces of information are then retrieved and provided as context to the language model, which generates a response based on this grounded knowledge.

The system uses ChromaDB as its vector storage engine. The knowledge base includes multiple types of content, such as restaurant descriptions, frequently asked questions, opening hours, location details, dietary and allergen information, and semantic representations of menu items. This heterogeneous knowledge is embedded using a dedicated text embedding model and stored in a unified vector space, enabling semantic similarity search across different categories of information.

When an agent processes a general inquiry, it queries the vector database using the embedded user question. The retrieved documents are selected based on semantic relevance and passed to the language model as contextual input. This process ensures that generated answers are both relevant and grounded in verified information. By explicitly linking responses to retrieved content, the system reduces hallucinations and improves factual accuracy.

The combination of relational storage for structured actions and vector-based retrieval for informational queries results in a hybrid knowledge management strategy. This design allows the assistant to reliably execute transactional workflows while also providing flexible and informative responses to open-ended questions, making it well suited for real-world restaurant interactions.

5 Voice and Multilingual Processing Pipeline

A key distinguishing feature of the proposed system is its ability to support real-time, multilingual voice interactions over standard phone calls. Unlike text-based conversational agents, a voice-based assistant must handle additional constraints related to audio processing, latency, and telephony infrastructure. The system addresses these challenges through a carefully designed processing pipeline that integrates speech recognition, language translation, conversational reasoning, and speech synthesis.

5.1 Phone Integration with Twilio

The primary user-facing interface of the system is a phone-based interaction layer built on top of the Twilio telephony API. Twilio provides reliable access to the public switched telephone network (PSTN), enabling customers to interact with the assistant using a standard phone call without requiring any additional software. However, this integration introduces strict operational constraints, particularly in terms of response time and request handling.

Twilio webhooks impose time limits on HTTP responses, which makes synchronous processing of speech recognition, language models, and text-to-speech impractical. To address this constraint, the system employs an asynchronous call handling strategy. Incoming calls are immediately acknowledged, and audio recordings are processed in background threads while the caller

hears a waiting message. This design prevents call timeouts and ensures a smooth conversational experience despite the computational cost of AI processing.

5.2 Asynchronous Audio Processing

Once a user's voice input is recorded, the audio file is downloaded and processed asynchronously. This decoupling of telephony control from AI computation allows the system to perform resource-intensive operations such as speech transcription and language model inference without blocking the call flow. Temporary audio files are managed carefully to prevent storage issues, and processing state is tracked per call to ensure correct synchronization between user input and system response.

Asynchronous processing is essential for maintaining acceptable latency in real-world phone interactions. By overlapping waiting periods with computation, the system achieves a balance between responsiveness and processing accuracy, even when operating under variable network or model inference conditions.

5.3 Speech-to-Text, Translation, and Response Generation

The core voice interaction pipeline follows a sequential transformation process: Speech-to-Text (STT), translation, agent reasoning, translation back to the user's language, and Text-to-Speech (TTS). First, the recorded audio is transcribed into text using a speech recognition model. This transcription step converts spoken input into a format suitable for natural language processing.

Because the system is designed to be multilingual, a translation layer is applied immediately after transcription. User input, regardless of the original language, is translated into a common pivot language used internally by the agents. This design choice simplifies agent logic and prompt design, as all reasoning and tool usage occur in a single language.

The translated text is then processed by the orchestrator and routed to the appropriate agent, which performs reasoning and executes any required actions. Once a textual response is generated, it is translated back into the user's original language. Finally, the translated response is converted into synthetic speech using a text-to-speech model and returned to the caller as an audio stream.

5.4 Language Detection Strategy

Accurate language detection is a critical component of the multilingual pipeline. The system performs automatic language identification on the first user utterance of each call. This detected language is stored and reused for subsequent turns, ensuring consistency throughout the conversation. By fixing the interaction language early, the system avoids oscillations between languages and reduces the risk of incorrect translations in multi-turn dialogues.

This strategy also enables language-specific exit commands and polite expressions to be recognized correctly, allowing users to naturally terminate the conversation in their preferred language.

5.5 Online and Offline Processing Modes

To balance performance, cost, and deployment flexibility, the system supports both online and offline processing modes. In online mode, cloud-based APIs are used for speech recognition, language translation, and text-to-speech synthesis, offering higher accuracy and lower latency at the expense of external dependencies and usage costs. In offline mode, local models are

employed for these tasks, enabling the system to operate without internet connectivity and reducing operational costs.

The ability to switch between online and offline modes makes the system adaptable to different deployment contexts, from development and experimentation to production environments with varying resource constraints. This dual-mode design enhances the robustness and portability of the voice assistant while preserving a consistent user experience.

6 Analytics and Monitoring Interface

In addition to handling customer interactions, the proposed system provides an analytics and monitoring interface designed for restaurant operators and administrators. This interface is implemented as a web-based dashboard using the Streamlit framework and serves as a complementary component to the conversational AI system. Its primary purpose is to offer visibility into restaurant operations and system activity, without interfering with real-time customer interactions.

The dashboard enables users to explore and analyze key operational data stored in the relational database. It provides access to information related to clients, reservations, tables, menu items, orders, and overall restaurant performance. Through interactive visualizations and filters, operators can monitor activity trends, identify peak periods, and gain insights into customer behavior. This separation of operational concerns allows the conversational assistant to remain focused on real-time interactions, while analytical tasks are handled independently through the dashboard.

A set of key performance indicators (KPIs) is exposed to support data-driven decision making. These indicators include metrics such as the number of reservations over time, table occupancy rates, order volumes, revenue distribution, top-selling menu items, and most frequent customers. Temporal analyses allow operators to observe trends across days or weeks, while categorical breakdowns provide insights into menu performance and customer preferences. The dashboard also supports data export functionality, enabling further offline analysis when needed.

Analytics play a critical role in restaurant management, as they help transform raw operational data into actionable insights. By visualizing patterns and performance indicators, operators can optimize staffing levels, adjust menu offerings, and better anticipate customer demand. Moreover, access to historical data enables continuous improvement by allowing managers to evaluate the impact of operational changes or promotional strategies.

Importantly, the analytics interface is architecturally decoupled from the conversational AI pipeline. The Streamlit dashboard operates in a read-oriented mode, querying the database without modifying its state or affecting the behavior of the agents. This clear separation ensures that monitoring and visualization do not introduce latency or instability into customer-facing interactions. As a result, the system achieves both operational reliability and managerial transparency, addressing the needs of multiple stakeholders within the restaurant environment.

7 Evaluation Methodology and Results

Evaluating conversational AI systems presents significant challenges due to their open-ended nature, multi-turn interactions, and reliance on probabilistic language models. Unlike traditional software systems with deterministic outputs, conversational agents may produce multiple

valid responses to the same input, making evaluation inherently complex. Additionally, the integration of reasoning, tool usage, and external data sources further complicates the assessment of correctness and performance. As a result, a single metric is insufficient to capture the overall quality of such systems.

To address these challenges, the proposed system adopts a multi-level evaluation strategy that decomposes the assistant's behavior into distinct components. This approach allows each subsystem to be evaluated independently while also measuring the overall end-to-end performance of the complete conversational pipeline.

7.1 Multi-Level Evaluation Strategy

The first evaluation level focuses on *intent classification*. Since the orchestrator is responsible for routing user requests to the appropriate agent, incorrect intent detection can lead to cascading errors. Intent classification is evaluated using labeled test queries and standard classification metrics, including accuracy, precision, recall, and F1-score. These metrics provide insight into how reliably the system distinguishes between reservations, orders, and general inquiries.

The second level evaluates *agent success rates*. Each specialized agent is tested against task-specific scenarios to determine whether it successfully completes the intended action. For example, the reservation agent is evaluated on its ability to correctly check availability, create reservations, and handle cancellations, while the order agent is assessed on order creation, item management, and order finalization. Success rates are computed as the proportion of scenarios in which the agent completes the task without errors or missing information.

The third level targets the quality of *Retrieval-Augmented Generation (RAG)*. Since the general inquiries agent relies on retrieved knowledge, it is essential to evaluate both retrieval relevance and answer grounding. Traditional information retrieval metrics such as Mean Reciprocal Rank (MRR) and Precision@K are used to assess how effectively relevant documents are retrieved. In addition, RAG-specific metrics are employed to evaluate the faithfulness and relevance of generated answers with respect to the retrieved context.

Finally, the system is evaluated through *end-to-end conversational scenarios*. These scenarios simulate realistic user interactions, including multi-turn dialogues and incomplete inputs. Metrics at this level include task completion rate, number of turns required to complete a task, and the system's ability to recover from missing or ambiguous information. This holistic evaluation captures the combined effect of speech processing, orchestration, agent reasoning, and response generation.

7.2 Metrics and Experimental Results

A variety of quantitative metrics are used to assess system performance. For intent classification, overall accuracy and macro-averaged F1-scores provide a balanced view of performance across intent categories. Agent success rates measure the reliability of task execution, while RAG evaluation relies on MRR, Precision@K, and specialized metrics such as answer relevancy and faithfulness, commonly implemented through frameworks like RAGAS.

Experimental results indicate strong performance across all evaluation levels. Intent classification achieves high accuracy and F1-scores, demonstrating effective routing of user requests. Specialized agents show high task success rates, particularly for well-defined workflows such as reservations and order management. RAG evaluation results confirm that retrieved information is both relevant and effectively used to ground generated answers, reducing hallucinations and

improving factual consistency. End-to-end scenario testing shows that the system can successfully handle realistic conversations with a limited number of turns and robust recovery from incomplete inputs.

Overall, these results validate the effectiveness of the proposed multi-agent architecture and evaluation methodology. By combining component-level metrics with holistic scenario-based testing, the evaluation provides a comprehensive and academically sound assessment of the system's performance in real-world restaurant interaction settings.