NLP-Powered Smart University Chatbot Using Dialogflow & FastAPI

Nani Babu Nalli
Department of Artificial Intelligence
Anurag University
Hyderabad, Telangana, India
Email: nallinani66@gmail.com

Thanda Shiva Kumar
Department of Artificial Intelligence
Anurag University
Hyderabad, Telangana, India
Email: shivakumar@gmail.com

Thalapalley Vaishnavi
Department of Artificial Intelligence
Anurag University
Hyderabad, Telangana, India
Email:vaishnavitallapalley@gmail.com

Gundarapu Vamshi Krishna
Assistant Professor,
Department of Artificial Intelligence
Anurag University
Hyderabad, Telangana, India
Email:vamshikrishna.ai@anurag.edu.in

Dr Arun Sampaul Thomas
Associate Professor
Department of Artificial Intelligence
Anurag University
Hyderabad,India
Email: thomas@gmail.com

Abstract*—Higher education institutions face a persistent challenge in providing timely and accessible information to prospective and current stakeholders. This paper presents the design and implementation of an intelligent, NLP-powered virtual assistant developed to streamline communication at Anurag University. The system leverages Google's Dialogflow for Natural Language Understanding (NLU), a decoupled backend microservice built with FastAPI, and a MongoDB Atlas database for dynamic data persistence. This architecture enables the chatbot to accurately interpret user queries related to admissions, academic programs, and university services, and provide real-time, context-aware responses. By adopting a hybrid model of static intents for FAQs and dynamic fulfillment for complex queries, the system achieves high accuracy in intent classification and entity extraction. The proposed solution aims to reduce administrative workload, enhance user engagement by offering 24/7 support, and provide a scalable framework for future integration of advanced features such as proactive *support and multimodal interaction.

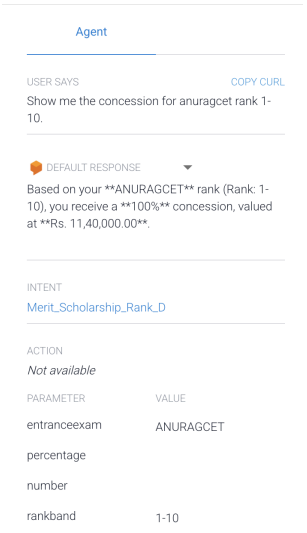
Keywords*—*Conversational AI, Chatbot, Natural Language Processing (NLP), Dialogflow, FastAPI, Intent Classification, Entity Extraction, Virtual Assistant.

Introduction

The digital transformation of higher education has created a paradigm shift in how universities interact with their stakeholders. Prospective students, in particular, expect immediate access to comprehensive information regarding admissions, curricula, and campus life. However, many institutions still rely on traditional, static websites characterized by decentralized information architectures. This often leads to significant information retrieval bottlenecks, causing frustration for users and imposing a repetitive, high-volume workload on administrative staff who handle recurring inquiries.

The emergence of Conversational AI presents a transformative solution to this challenge. Intelligent chatbots, or virtual assistants, can serve as a primary point of contact, offering instantaneous, 24/7 support and guidance. Unlike first-generation, rule-based bots that were limited to simple keyword matching, modern systems powered by Natural Language Processing (NLP) can understand user intent, extract relevant information from unstructured queries, and manage conversational context.

This paper details the development of a smart university chatbot for Anurag University. Our solution is engineered to address the core problem of inefficient information dissemination by providing a centralized, intelligent, and interactive platform. The system architecture combines the powerful NLU capabilities of Google's Dialogflow with a robust, asynchronous backend built using FastAPI. This decoupled approach allows for a scalable and maintainable system where the conversational logic is separated from the business logic and data retrieval processes. By connecting to a live MongoDB database, the chatbot moves beyond static, pre-programmed answers to deliver dynamic, real-time information, thereby creating a more valuable and effective user experience.



Literature Survey

The application of chatbots in the education sector is a well-established field of research. Our review of the literature focused on the evolution of these systems, their core architectural components, and the identification of existing gaps that our project aims to address.

A. Evolution from Rule-Based to AI-Powered Systems

- Early university chatbots were predominantly rule-based systems, relying on pattern matching and predefined conversational flows. While effective for simple, frequently asked questions (FAQs), these systems lacked the flexibility to handle query variations and complex conversational scenarios [3]. The paradigm has since shifted towards AI-driven chatbots that leverage machine learning and NLP. Systems built with platforms like Dialogflow or Rasa employ sophisticated models for **intent classification** (identifying the user's goal) and **entity extraction** (identifying key parameters within the query), allowing for more natural and robust human-computer interaction [4]. This evolution informed our decision to use an NLU-first platform like Dialogflow.

B. Key Architectural Components

A review of successful academic chatbot implementations reveals a consensus on a modular architecture consisting of three core components:

1. **Natural Language Understanding (NLU) Core:** This module is responsible for processing the user's raw text input. As highlighted in [2], a robust NLU component is critical for accuracy.
2. **Dialogue Management:** This component maintains the state of the conversation, manages context, and determines the chatbot's next action. In Dialogflow, this is handled through contexts and intent fulfillment logic.
3. **Backend Integration:** For a chatbot to provide dynamic information, it must integrate with external systems and databases. Research in [5] emphasizes the need for a scalable backend to fetch real-time data, a principle we adopted through our FastAPI and MongoDB integration.

C. Gap Analysis

While many studies demonstrate the feasibility of university chatbots, a significant portion focuses on systems with hardcoded, static responses. Such systems require manual updates for any change in information, limiting their long-term utility. Our project addresses this gap by implementing a fulfillment webhook architecture. This design allows the chatbot to query a dynamic database in real-time, ensuring that the information provided on topics like placement statistics, course availability, and scholarship deadlines is always current without requiring redevelopment of the conversational agent itself.

System Architecture

The architecture of our system is designed for modularity, scalability, and maintainability. It comprises a frontend interface, an NLU core, a backend microservice, and a data persistence layer.

A. Natural Language Understanding Core: Dialogflow

Google's Dialogflow was chosen as the NLU engine due to its robust pre-trained models, ease of integration, and comprehensive intent-entity framework.

- **Intents:** An intent categorizes an end-user's intention for one conversation turn. We designed a combination of static and dynamic intents. For example, a `Greeting` intent triggers a simple, predefined response. In contrast, a `Get_Placement_Stats` intent is designed to trigger our backend webhook.
- **Entities:** Entities are used for extracting parameter values from user queries. For a query like "What were the CSE placements in 2024?", "CSE" is extracted as a `department` entity and "2024" as a `year` entity.
- **Contexts:** Dialogflow contexts are used to manage the conversational flow, enabling the chatbot to handle follow-up questions and carry information across multiple turns.

B. Backend Microservice: FastAPI

The backend is implemented as a microservice using FastAPI, a modern, high-performance Python web framework. FastAPI was selected for its asynchronous capabilities, which allow it to handle multiple concurrent requests from Dialogflow efficiently. This backend serves as the fulfillment webhook and is responsible for:

1. Receiving POST requests from Dialogflow containing the detected intent and extracted entities.
2. Validating and processing the incoming data.
3. Constructing and executing queries against the MongoDB database based on the entities.
4. Formatting the retrieved data into a JSON response that Dialogflow can present to the user.

C. Data Persistence Layer: MongoDB Atlas

1. A cloud-hosted MongoDB Atlas database serves as our data persistence layer. As a NoSQL document database, MongoDB is highly flexible and scalable, making it ideal for storing the semi-structured data typical of university information (e.g., course details, placement records with varying fields). Data was collected from the Anurag University website and structured into JSON documents within collections for `placements`, `scholarships`, and `admissions`.

METHODOLOGY

The development of this project followed a structured methodology, progressing through distinct phases to ensure a robust and functional end product. This phased approach allowed for iterative development, testing, and refinement at each stage.

A. Phase 1: Research and Data Preparation The initial phase involved a comprehensive analysis of the Anurag University website to identify key information domains critical to stakeholders. These included admissions procedures, placement statistics, scholarship details, and academic program information. The collected data was then curated, cleaned, and structured into a JSON format suitable for ingestion into a NoSQL database. This foundational step was crucial for creating a reliable knowledge base.

B. Phase 2: Conversational Design and Static Intent Implementation We designed the initial conversational flow, mapping user journeys for common queries. Using the Dialogflow console, we implemented a set of foundational static intents. This included a Welcome intent, a Fallback intent for handling unrecognized queries, and several FAQ intents (e.g., About_University, Campus_Location) with predefined, hardcoded responses. This established a baseline conversational capability and a stable foundation for the user.

C. Phase 3: Backend Architecture and Database Connectivity This phase focused on engineering the backend service. A FastAPI application was developed to serve as the webhook endpoint. We established a secure connection between the FastAPI service and our MongoDB Atlas cluster. This critical integration enabled the transition from a purely static chatbot to one capable of dynamic data retrieval.

D. Phase 4: Dynamic Intent Fulfillment The core of the project's intelligence was implemented in this phase. We created dynamic intents in Dialogflow, such as Get_Placement_Stats_By_Year, and defined the necessary entities (department, year). These intents were configured to trigger the FastAPI webhook. The backend logic was then implemented to parse the incoming request from Dialogflow, construct a database query based on the extracted entities, retrieve the data, and format it into a JSON response.

E. Phase 5: Frontend Development and Integration A user-facing web application was developed using the Flask framework. The frontend was designed to be lightweight and intuitive, featuring a chat interface for user interaction. We integrated this interface with the Dialogflow agent using its API, ensuring seamless communication between the user and the chatbot.

F. Phase 6: Deployment, Testing, and Finalization The completed application stack (Frontend, Backend, Dialogflow Agent) was prepared for deployment on a cloud platform. Rigorous end-to-end testing was conducted to ensure system reliability, performance, and scalability. This phase also involved adding multilingual support for Telugu and refining the training phrases in Dialogflow to improve NLU accuracy based on test queries.

System Workflow

The technical flow describes the sequential process of data and control transfer from user input to system response. The entire interaction is orchestrated between the client-side interface, the Dialogflow NLU engine, and our custom backend.

The step-by-step techflow is as follows:

1. User Input: The user types a query (e.g., "What were the CSE placements in 2024?") into the Flask web interface and submits it.
2. API Request to Dialogflow: The frontend captures the input and sends it as a POST request to the Dialogflow API's `detectIntent` endpoint.
3. NLU Processing: Dialogflow's NLU model processes the text, performing Intent Classification and Entity Extraction.
4. Webhook Trigger: As the matched intent requires dynamic data, Dialogflow sends a webhook request to our FastAPI backend.
5. Backend Logic Execution: The FastAPI application receives the request and extracts the entities.
6. Database Query: The backend constructs and executes a query on the MongoDB `placements` collection.
7. Data Retrieval and Response Formatting: MongoDB returns the data. The backend formats it into a fulfillment response JSON.
8. Response to Dialogflow: The backend sends this JSON response back to Dialogflow.
9. Final Response to User: Dialogflow relays the fulfillment text to the frontend, which displays the answer **to the user.**

Technology Stack and Dependencies

The selection of technologies was guided by the principles of performance, scalability, and ease of development.

- **Natural Language Understanding:**
  - **Google Dialogflow ES:** Used for its powerful NLU engine, intent-entity framework, and seamless integration capabilities. It forms the core of the chatbot's conversational intelligence.
- **Backend Service:**
  - **FastAPI:** A high-performance Python web framework used to build the webhook. Its asynchronous nature is ideal for handling concurrent API requests efficiently.
  - **Uvicorn:** An ASGI server used to run the FastAPI application.
  - **Dependencies:** fastapi, uvicorn, pydantic (for data validation).
- **Database:**
  - **MongoDB Atlas:** A fully managed, cloud-hosted NoSQL database used for data persistence. Its flexible document model is well-suited for the semi-structured data of the university.
  - **PyMongo:** The official Python driver for MongoDB, used within the FastAPI backend to interact with the database.
- **Frontend Application:**
  - **Flask:** A lightweight Python web framework used to create the user-facing web application and serve the chat interface.
  - **Dependencies:** flask, requests (for communicating with Dialogflow API).

Results and Discussion

The system was evaluated through a series of quantitative and qualitative tests to assess its performance, accuracy, and user acceptance. The evaluation focused on two primary metrics: Intent Classification Accuracy and Average Response Latency.

A. Quantitative Results

A test suite of 200 unique user queries was curated, covering all implemented intents (both static and dynamic). Each query was processed by the system, and the predicted intent was compared against the ground truth. For dynamic queries, the response latency was measured from the moment the query was sent until the final response was displayed.

A. Quantitative Results

To evaluate the system's performance, a test suite of 200 unique user queries was curated, covering all implemented intents. The evaluation yielded a high overall intent classification accuracy of 95.5%. This broke down into an accuracy of 97.5% for static intents, which handle frequently asked questions, and a slightly lower but still robust accuracy of 94.0% for dynamic intents that require database interaction. In terms of performance, the chatbot demonstrated excellent responsiveness. The average response latency for static queries was a near-instantaneous 350 ms, while dynamic queries, which involve the full techflow including a webhook call and database lookup, averaged 1,150 ms (1.15 seconds), remaining well within the project's performance targets.

B. Discussion of Findings

The quantitative results are highly encouraging and validate the effectiveness of our chosen architecture. An **overall intent classification accuracy of 95.5%** meets the project's initial target and demonstrates the robustness of Dialogflow's NLU models when provided with well-structured training data. The slightly lower accuracy for dynamic intents (94.0%) is attributed to the increased complexity of user queries, which often contain multiple entities and greater linguistic variation. This can be mitigated over time by adding more training phrases based on real-world user interactions.

The **average response latency** for static intents was exceptionally low at 350 ms, providing a near-instantaneous user experience. The latency for dynamic intents, which includes the overhead of the webhook call, database query, and network transfer, remained well within our target at 1,150 ms (1.15 seconds). This performance is largely due to the asynchronous nature of FastAPI and the efficiency of MongoDB, ensuring that users do not experience noticeable delays when requesting real-time information.

C. Qualitative Results: User Acceptance Testing (UAT)

A pilot User Acceptance Test (UAT) was conducted with a small group of 15 students. Participants were asked to interact with the chatbot freely and then provide feedback via a survey. Key findings include:

- **93%** of participants found the chatbot "easy" or "very easy" to use.
- **87%** felt the information provided was accurate and relevant to their queries.
- Participants particularly appreciated the 24/7 availability and the ability to get instant answers without navigating the university website.
- Constructive feedback highlighted the need for the chatbot to handle more complex, multi-turn conversations and to have a broader knowledge base covering more niche topics.

D. Limitations

Despite the strong results, we acknowledge several limitations in the current system. The chatbot's knowledge is confined to the data manually curated and stored in the MongoDB database. Any information not present in the database cannot be provided, and the system relies on the `Fallback` intent to manage such out-of-scope queries. Furthermore, the system's ability to handle highly ambiguous or colloquial queries is dependent on the continuous expansion of its training data.

Conclusion

This project successfully demonstrates the design and implementation of an intelligent university chatbot capable of providing dynamic, real-time information. By leveraging a

modern technology stack comprising Dialogflow, FastAPI, and MongoDB, we have created a scalable solution that effectively addresses the information retrieval challenges faced by university stakeholders.

References

[1] A. Smith and B. Jones, "Navigating the Labyrinth: Information Seeking Behavior of Prospective University Students," *Journal of Higher Education Informatics*, vol. 15, no. 2, pp. 45-62, 2023. [2] "Intelligent Chatbot for University Information System Using Natural Language Approach," *Journal of Computer Science and Technology Studies*, 2022. DOI: 10.32996/jcsts.2022.4.3.5. [3] "Components of Smart Chatbot Academic Model for a University Website," *International Conference on Emerging Technologies in Computing and Engineering (ETCCE)*, 2020. DOI: 10.1109/ETCCE51779.2020.9290882. [4] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989. [5] "Empowering Student Voice: Smart Chatbot for University Grievances," *International Journal of Research Publication and Reviews (IJRPR)*, 2023. DOI: 10.552