Contents

1 Introduction

Intelligent Q&A systems have seen growing use in education, customer service, and online consultation. With advances in NLP, machine learning, and large language models (LLMs), chatbots are now more capable of understanding complex queries and managing multi-turn conversations.

In university settings, students often face practical questions related to course schedules, administrative procedures, and campus facilities. Traditional methods of information access such as browsing websites or asking peers can be time-consuming and inefficient.

This project aims to design and implement a chatbot tailored to student inquiry scenarios. It accurately detects user intent, recognizes key entities, and generates responses based on a built-in campus information knowledge base. By integrating multiple NLU components, including intent classification, entity recognition, and sentiment analysis, the system enhances user experience and improves overall information availability on campus.

2 Project Methodology

2.1 Research Design and Paradigm

This project adopts the Constructive Research / Design Science (DSR) research paradigm. The core objective is to design, build, and evaluate a novel intelligent system—a hybrid chatbot that integrates both retrieval-based and template-based features—to address real-world problems such as low information retrieval efficiency and lack of emotional interaction in campus settings(Yang et al., 2019).

The overall research process follows an iterative development cycle:

1. Requirement Analysis and Data Collection: Identify key information needs in campus scenarios and gather relevant raw data.

2. System Design and Implementation: Design and build a system architecture comprising a Natural Language Understanding (NLU) module, a response generation module, and an interactive front-end interface.

3. Model Training and Development: Train the core NLU model using synthetic data generated by LLM and inspected by human.

4. System Integration and Evaluation: Integrate all modules into a working prototype

and evaluate the system using both quantitative and qualitative methods.

5. Iterative Optimization: Refine the system by enriching training data and optimizing components based on evaluation feedback.

This design-oriented research approach ensures that the outcomes are not only theoretically grounded but also practically valuable. The iterative methodology further enables rapid feedback incorporation and continuous system improvement in real-world usage scenarios.

2.2 System Architecture

The chatbot system follows a reasonable architecture as Mohammed and Aref (2022) adopted in their research, consisting of a frontend user interface and a backend processing core. The backend adopts a modular design, primarily comprising a Natural Language Understanding (NLU) module and a response generation module. The overall architecture is shown in fig. 2.1. The functions of each component will be described in detail in the following subsection.

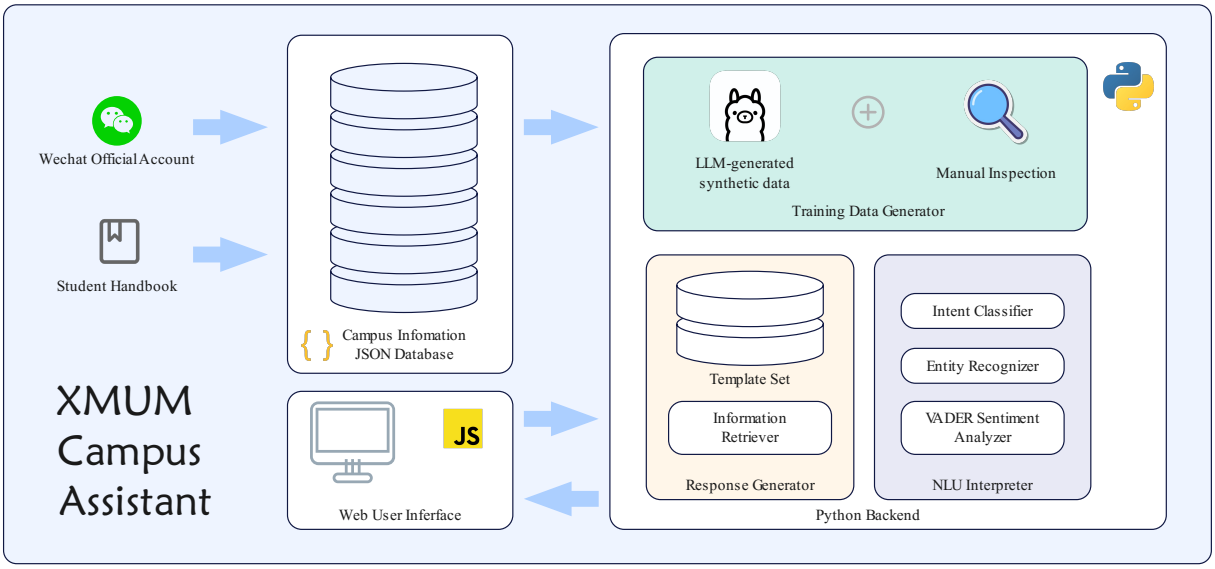

Figure 2.1: *Overall System Architecture*

2.3 Data Collection and Preparation

High-quality data is fundamental to the performance of natural language processing systems. In this study, data collection was conducted on two levels: (1) organizing raw information for knowledge base construction, and (2) generating training corpora for the NLU model. Both types of data underwent structured processing and manual verification to ensure their usability and reliability in system development.

2.3.1 Knowledge Base Data

We manually collected information on campus facilities, dining options, academic procedures, and more by consulting the student handbook, official public accounts, and other campus information sources, and structured it into a searchable `JSON` file as the knowledge base of the intelligent system. Here is a snippet of the knowledge base:

```json
"restaurant": {
        "possible_intent": ["ask_restaurant_location", "ask_restaurant_time", "
          ask_restaurant_recommendation"],
        "starbucks": {
                "restaurant_name": "Starbucks",
                "possible_name": ["Starbucks", "starbucks", "STARBUCKS"],
                "location": "Bell Avenue Roadside",
                "opening_hours": "9:00-21:00",
                "recommendation_reason": "It's a coffee shop with various drinks and
                   snacks"
        },
        ...
}
```

2.3.2 Model Training Data

Since we will adopt supervised learning to train the NLU model (which will be introduced in section 3.1.2), a large amount of training data is needed. However, preparing them completely manually will be extremely time-consuming. Thus, we adopted a *synthetic data* strategy. Specifically, we first designed a data generator (`data_generator.ipynb`) that fills predefined intents and entities into prompt templates. Then, a locally deployed `Ollama` LLM is used to generate many feature-label pairs. Finally, all generated data must undergo **manual sampling, review, and correction** to ensure quality and accuracy before being used for model training as Li, Bonatti, Abdali, Wagle, and Koishida (2024) stressed in their research.

2.4 Evaluation Methods

To comprehensively evaluate the effectiveness of this project, we adopted a combination of quantitative and qualitative assessment methods(Alfrink, 2024).

2.4.1 Quantitative Evaluation

The evaluation primarily focuses on the performance of the NLU model. We used an independent test set and employed scikit-learn's `classification_report` tool to auto-

matically calculate a range of model evaluation metrics, including precision, recall, and F1-score—derived from the confusion matrix—to assess the model's performance across each class(scikit-learn, n.d.).

2.4.2 Qualitative Evaluation

The evaluation primarily targets the overall user experience and practicality of the chatbot system. We conducted multi-turn, open-ended conversations with the chatbot, with a particular focus on its ability to handle edge cases beyond the training data. This allowed us to assess the fluency of the dialogue, the accuracy of the responses, and the effectiveness of emotional interaction.

3 Design of the Intelligent System

3.1 Core Modules and Their Functions

3.1.1 User Input Interface

This module is responsible for capturing user input and sending it to the back-end interpretation module. We adopted a Web UI architecture, with the frontend built using `JavaScript` and `HTML`, and connected to the Python backend via the `Flask` framework.

3.1.2 Natural Language Understanding (NLU) Module

The Natural Language Understanding (NLU) module serves as the system's initial processing stage, responsible for transforming users' free-form text input into structured semantic information(Smutek, Golec, Rymarczyk, Jarmuł, & Hernas, 2024). It consists of the following three subsystems:

- **Intent Classification**
  This model aims to identify the user's query intent within a specific context. The task is formulated as a multi-class text classification problem, which we adopt supervised learning to solve it:

  1. Prepare labeled data for model training and use split it into train set and test set.
  2. Construct a Scikit-learn `Pipeline` consisting *TF-IDF vectorizer* and *logistic regression* and train the model(Batra, 2023).
  3. Check the `classification_report` and generate more training data to improve the performance of the intent classifier.
     For example, we once get the classification report below:

Table 3.1: *Classification Report of Intent Classifier (sorted by number of training data)*

Category	Precision	Recall	F1-score	Support	Training Data
ask_handbook_info	0.965	0.991	0.978	110	482
ask_restaurant_time	0.982	0.991	0.987	112	452
ask_restaurant_location	0.949	0.970	0.960	135	419
ask_facility_time	0.812	0.897	0.852	29	155
ask_facility_info	0.857	0.667	0.750	36	139
ask_facility_location	0.730	0.794	0.761	34	123
ask_building_location	0.786	0.647	0.710	17	67
greet_welcome	0.385	0.625	0.476	8	58
greet_sorry	1.000	0.917	0.957	12	56
ask_building_include	1.000	0.667	0.800	12	50
greet_hello	0.455	0.385	0.417	13	45
greet_thanks	1.000	1.000	1.000	11	41
greet_goodbye	0.800	0.364	0.500	11	39
Accuracy		0.910		641	
Macro Avg	0.833	0.779	0.793	641	
Weighted Avg	0.911	0.910	0.907	641	

We can observe from table 3.1 that the more training data we have, the better the performance of the intent classifier is. Also, it's worth noting that some categories, such as `greet_thanks` seem to have a very high precision score but have low amount of training data. This may be attributed to the limited size of the test dataset, potentially leading to **sampling bias** or **overfitting**.

Eventually, we obtain more synthetic data for those categories with small amount of training data and enhance the overall performance of the intent classifier.

- **Entity Recognition**
  To further extract key information from user input (such as location, time, or objects), we need to recognize entities. This task is modeled as Named Entity Recognition (NER) in sequence labeling problem(Singh, 2024). We adopt the following procedures to solve it:

  1. Prepare labeled data for model training and convert it to the tuple data format required by `spaCy`. Example:
     `[('Where is Xia Yi Cheng located?', 'entities': [(9, 21, 'restaurant')])]`
  2. Train a `spaCy` model to recognize it.

3. Check the model performance using a self-built classification report and increase training data amount accordingly.

- **Sentiment Analysis**
  To enhance the system's human-centered and emotional awareness capabilities, the *VADER sentiment analysis* tool is integrated to classify the emotional tone and intensity of user input(Hutto & Gilbert, 2014). The output includes positive, negative, and neutral sentiment categories, along with a compound score, which informs template selection and interaction tone adjustment.

3.1.3 Response Generation Module

The response generation module is responsible for producing user-understandable and naturally styled text replies based on the intent, entities, and sentiment data output by the NLU module. It comprises the following two core components:

- **Knowledge Base Retriever**
  The system's knowledge is organized in a structured JSON format, covering common campus inquiry scenarios. The retriever receives keywords and labels from the NLU module and performs matching queries within the knowledge base to locate the most relevant information entries.

- **Response Templating Engine**
  A hierarchical response templating strategy is employed, with standardized language templates designed for different intent types and sentiment labels.Based on the user's identified intent, extracted entities, and sentiment scores, the engine dynamically selects the appropriate template and fills in placeholders with information retrieved from the knowledge base, ultimately generating coherent and complete natural language responses.

3.1.4 Interactive Front-end Interface

The front-end interface is built using a web framework and provides a chat-like, turn-based interaction experience.

3.2 Data Flow and Workflow

The system operates based on a single-turn interaction model following the sequence: *Input → Interpretation → Generation → Output.* For each user query, the system sequentially invokes its modules and returns a structured, readable response. The following describes a typical data flow during system operation.

3.2.1 Input Stage

System operation begins when the user submits a natural language query through the input interface, such as ”Where is Starbucks located?”. Before entering the main processing pipeline, the input is first passed through a sentence segmentation step to enable sentence-by-sentence analysis for multi-sentence queries.

3.2.2 Natural Language Understanding Stage

Once the input text enters the Natural Language Understanding (NLU) module, it undergoes the following processing steps in sequence:

- The intent classifier detect the intent of user input, such as `ask_facility_location` or `ask_restaurant_time`.
- The entity recognizer extracts key entities from the input, such as `Starbucks: restaurant` or `library: facility`, to support content generation in the response module.
- The VADER sentiment analyzer detect the sentiment score of user input and categorize it into three tags: positive, neutral, and negative.

After each clause is processed, the system organizes the output into a unified intermediate format. For example, given the input ”Where is Starbucks located?”, the structured output is:

```json
{
        "query": "Where is Starbucks located?",
        "intent": "ask_restaurant_loaction",
        "entities": {"Starbucks": "restaurant"}
        "sentiment_tag": "neutral"
}
```

The whole NLU process is shown in fig. 3.1.

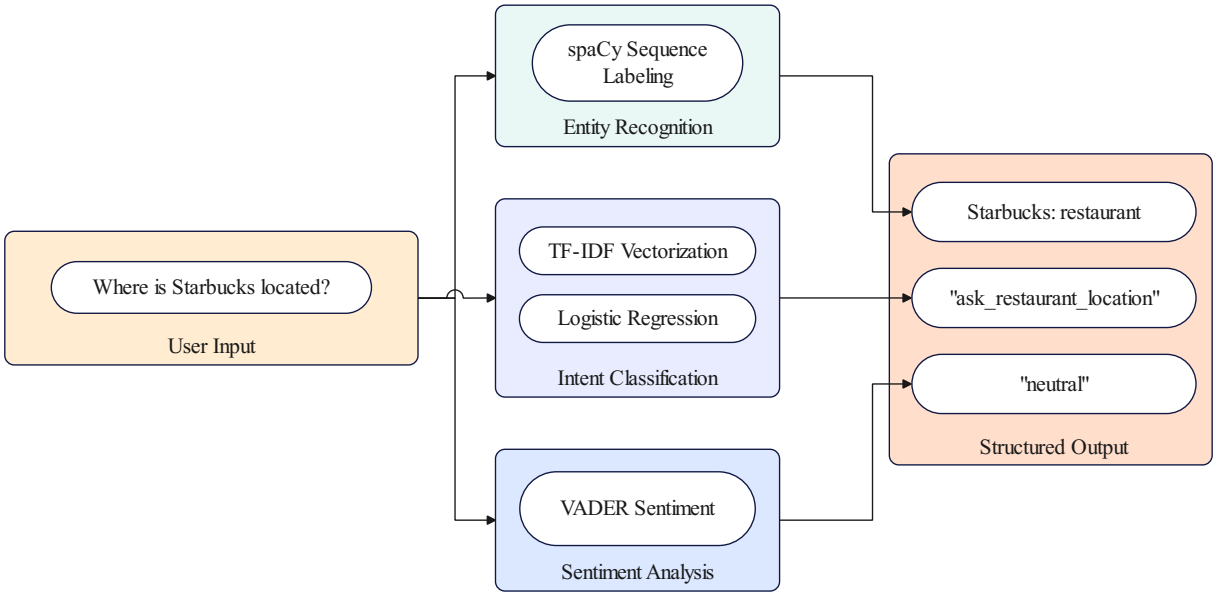

Figure 3.1: *NLU Process Flow Chart*

3.2.3 Response Generation Stage

Based on the structured output, the system invokes the response generation module. It will first match the correct output template according to the intent and sentiment and retrieve in the knowledge base for information to be filled in the template.

For instance, if the module receive the output above, it will find the following template:

JSON

```
      {
      "ask_restaurant_location":{
              "info":["restaurant_name","location"],
              "neutral": "{restaurant_name} can be found at {location}.",
      }
      }
```

And it will retrieve in the knowledge base for `restaurant_name` and `location` of Starbucks until it finds:

JSON

```
       "starbucks": {
    "restaurant_name": "Starbucks",
    "possible_name": ["Starbucks", "starbucks", "STARBUCKS"],
    "location": "Bell Avenue Roadside",
    "opening_hours": "9:00-21:00",
    "recommendation_reason": "It's a coffee shop with various drinks and snacks"
  }
```

The whole response generation process is shown in fig. 3.2.

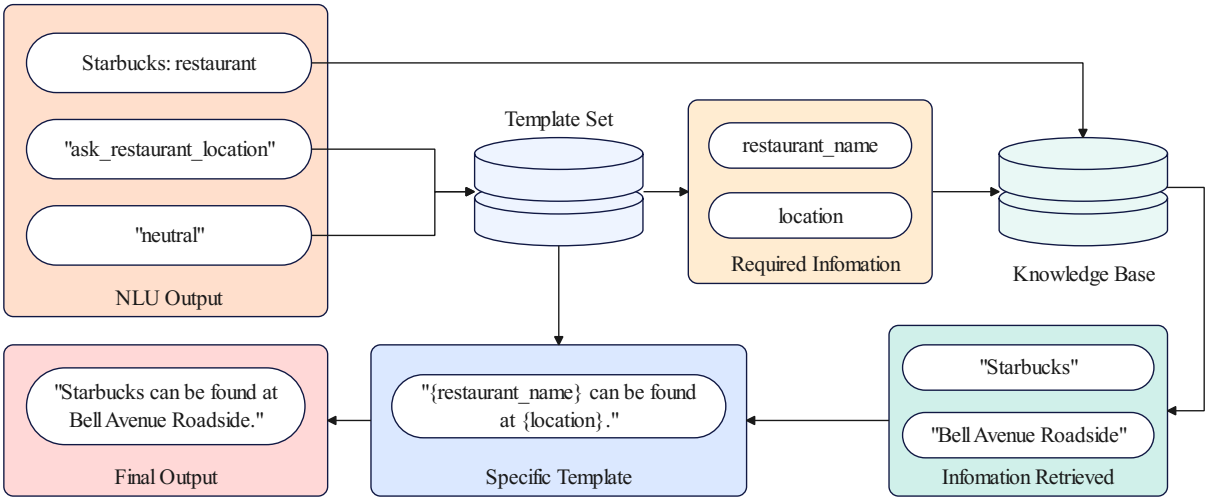

Figure 3.2: *Response Generation Process Flow Chart*

3.2.4 Output Stage

The final response will be transmitted from the Python backend to the Web UI for display, using the `Flask` framework.

4 Implementation

Before the implementation, kindly refer to appendix A for the runtime dependencies of this intelligent system.

5 Discussion

5.1 Summary of Overall Effectiveness

This system is designed to improve information access for XMUM students and enhance the accessibility of campus-related content. During development, a modular architecture was adopted to facilitate system construction and deployment.

Evaluation results indicate that the system performs reliably in intent classification and entity recognition, effectively covering most common campus-related queries. The sentiment analysis module also contributed to tone adjustment in responses, enhancing the naturalness and user-friendliness of interactions. Although the current version does not yet support multi-turn dialogue or context modeling, it has demonstrated task completion and user experience in single-turn scenarios, meeting the intended design goals.

5.2 Advantages of Models and Systems

The system demonstrates several key strengths in both core modeling and overall architectural design. Firstly, the NLU module performed well on the test set, showing high accuracy and stability, which provides a reliable foundation for information retrieval. The response generator can match template and retrieve information accurately, ensuring the quality of responses. Additionally, the use of Web UI enhances the overall usability and accessibility of the intelligent system.

5.3 Issues and challenges

During the development and deployment of the system, we also encountered several practical challenges and limitations. One of the major issues was the limited timeliness of the raw data, which made it difficult to keep up with real-time changes in campus operations and nearby businesses. In certain query scenarios, this could lead to outdated information, affecting the accuracy and credibility of the responses.

5.4 Directions for Improvement

Although the current system has demonstrated stable performance in single-turn question-answering scenarios, there are several areas for improvement in future iterations. Firstly, incorporating a context-tracking mechanism would enable the system to understand multi-turn conversations, enhancing its ability to handle complex queries with greater coherence and continuity. Secondly, the database used for model training and information retrieval can be expanded so that the system can meet more of students' usage needs. Lastly, on the front-end side, further optimization of the interface design and response feedback mechanisms could improve usability and provide a smoother user experience.

References

Alfrink, K. (2024). *Contestable artificial intelligence: Constructive design research for public artificial intelligence systems that are open and responsive to dispute* (Dissertation (TU Delft), Delft University of Technology). doi: 10.4233/uuid:0b014ee6-67a8-4c59-8598-4bfd771595a3

Batra, R. (2023, 1). *Multi-Class Text Classification with Scikit-Learn using TF-IDF model.* Retrieved from https://medium.com/%40rohit_batra/multi-class-text-classification-with-scikit-learn-using-tf-idf-model-161d395ce374

Hutto, C., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Confer-*

ence on Web and Social Media, *8*(1), 216-225. Retrieved from https://ojs.aaai.org/index.php/ICWSM/article/view/14550 doi: 10.1609/icwsm.v8i1.14550

Li, Y., Bonatti, R., Abdali, S., Wagle, J., & Koishida, K. (2024). *Data generation using large language models for text classification: An empirical case study.* Retrieved from https://arxiv.org/abs/2407.12813

Mohammed, M., & Aref, M. M. (2022). *Chatbot system architecture.* Retrieved from https://arxiv.org/abs/2201.06348

scikit-learn. (n.d.). *classification_report.* Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html (Accessed June 26, 2025)

Singh, A. (2024, 5). *How to train Custom Named Entity Recognition [NER] model with SPACY.* Retrieved from https://www.newscatcherapi.com/blog/train-custom-named-entity-recognition-ner-model-with-spacy-v3

Smutek, T., Golec, E., Rymarczyk, P., Jarmuł, M., & Hernas, A. (2024, 08). Enhancing conversational ai with the rasa framework: intent understanding and nlu pipeline optimization. *Journal of Modern Science*, *57*, 531-548. doi: 10.13166/jms/191223

Yang, L., Hu, J., Qiu, M., Qu, C., Gao, J., Croft, W. B., … Liu, J. (2019). *A hybrid retrieval-generation neural conversation model.* Retrieved from https://arxiv.org/abs/1904.09068

A Runtime Dependencies for This Intelligence System

This system consists of a `Python` backend and a `JavaScript` frontend, and needs certain dependencies before implementation.

For the `JavaScript` frontend:

- `Node.js v20.18.0`
- `npm v10.8.2`

The rest of frontend dependencies will be automatically installed by the command `npm install` when the system starts.

For the `Python` backend:

- `Python v3.12.6`
- `pip v25.1.1`
- Required Python Packages:

- `flask`
- `flask-cors`
- `joblib`
- `spacy`
- `thinc`
- `pandas`
- `scikit-learn`
- `vaderSentiment`
- `tqdm`
- `ollama` (optional, for generating training data. An NVIDIA discrete GPU with at least 8GB VRAM and a locally deployed `Ollama` LLM are required.)

Kindly run the following command to install the required Python packages:

Windows Command Prompt

```
pip install flask flask-cors joblib spacy thinc pandas scikit-learn
  vaderSentiment tqdm ollama
```