Travel Agent Natural Language Processing Demo Chatbot

Ziad Atari

1. Introduction

Travel planning is a time consuming and complex process which involves multiple platforms and steps to achieve the desired outcome. To streamline this process a Travel Agent chatbot was developed. This demonstrates the interactive and dynamic abilities of chatbots in helping users efficiently meet their travel planning needs, including booking hotels and flights, cancelling reservations and engaging in lighthearted small talk.

This chatbot is designed to function as a dialogue-based system, leveraging the power of natural language processing (NLP) techniques and integrating key principles of Conversations User Interface (CUI) design [1]. By combining machine learning based intent recognition, database-backed transaction management systems and user-friendly conversational designs, the system delivers an insight into the capabilities of Chatbots to deliver robust and intuitive user experiences.

The report outlines the chatbots architecture, implementation and evaluation, highlighting the use of module programming, intent-matching via machine learning, and context aware conversational flows. Additionally, it discusses the integration of identity management to personalizes interaction and error handling strategies to ensure seamless recovery from errors and unexpected queries.

2. Chatbot Architecture

2.1 Core Functionalities

Intent detection

Intent detection is a foundational component, it holds the job of determining the user's goal from their input. The system deploys a machine learning based pipeline in intent_model.py using:

- TF-IDF Vectorization: Converts user input into numerical representation based on word frequency and importance in the context of the dataset. [2]
- Logistic regression Classifier: This is a robust and efficient classifier trained on a 169 sentence long labelled dataset to handle diverse user queries. [3]

The system classifies the inputs into the following predefined intent categories:

- flight_booking
- hotel_booking
- cancel_booking
- get_name
- small_talk
- name_introduction
- current_bookings
- goodbye
- unknown

This allows for a scalable and flexible system capable of recognizing broad range queries. The implementation technique is based on of similarity-based intent detection techniques described in lab 3 [4] supplemented by lecture material on intent-routing and classification.

Identity Management

Personalization is a vital aspect of dialogue-based interactions. The Chabot achieves this through an identity management system, leveraging Named Entity Recognition (NER) techniques [5] and regex based parsing the system in utils.py can extract and store the users name allowing for interactions such as:

```
Travel Agent Bot: I don't know your name yet. What's your name?

 You: Hello, my name is John.

Travel Agent Bot: Welcome back, John!
```

The system is also capable of retrieving stored names across multiple sessions through a local database, creating a personalized user experience aligning with the emphasis on enhancing conversational personalization.

Transactional dialogue

The chatbot facilitates structured, multi-turn conversations for:

- Flight Booking: Users provide departure and arrival destination which are verified against an IATA airport dataset to check if the airports exist. This is followed by the user specifying the subsequent dates in a multi-turn dialogue handled by flight_booking.py
- Hotel Booking: Users specify the city of choice, pick from a hotel list, then confirm details like check-in/check-out and room quantities in a multi turn dialogue handled in hotel_booking.py.
- Cancellations: Users can view a list of both hotel and flight bookings each labelled with an ID, to which the users provide if they wish to cancel that is then verified and deleted from the database through database.py and cancellation.py, an example of which is shown below:

```
You: I want to cancel my hotel

Travel Agent Bot: Here are your current bookings:

Flight Booking ID 7633: London from JFK to 2024-12-12

Hotel Booking ID 7257: Jakarta from Crown Plaza to 2024-11-15, Rooms: 2024-11-20

Travel Agent Bot: Please enter the booking ID you wish to cancel:

You: 7257

Travel Agent Bot: Your hotel booking ID 7257 has been cancelled.
```

Small Talk

To make the interactions more human and engaging, the chatbot includes a small-talk module. The module uses two question/answer datasets enriched with fuzzy matching using cosine similarity inspired by labs on string matching and token similarity to address non travel related queries such as:

- *"How are you?"*
- *"Tell me a Joke"*
- *"Who killed Julius Caesar?"*

This improves user satisfaction by handling off task conversational inputs, a critical element in improving the user experience.

Error Handling

The chatbot was reinforced with several error handling routines and fallback strategies to create a robust system capable of dealing with unexpected situations. The mechanisms can be classified in the following groups:

1. Input Validation: The chatbot includes several validation checks for prompts like dates, airports and formatting with their respective explicit error messages, to maintain the chatbots integrity through rejecting illogical inputs.
2. User Cancelation: The chatbot provides users with graceful exit options through keywords like *"stop"* and *"nevermind"* that trigger the cancellation intent and exits the transaction.
3. Exception Handling: Critical operations like database queries are wrapped in try-except blocks to isolate and prevent crashes. This is paired with extensive logging and fallback messages such as *"Something went wrong. Please try again."* To allow for debugging traceable errors while maintaining a user-friendly CUI.
4. Model and File Handling: The initialization process issued by main.py, includes a check of key components and dependencies such as the database and intent-model with fallback instructions to retrain the model and recreate the database before dialogue begins.
5. Fallback Responses: The chatbot features a fallback response protocol where all ambiguous inputs are checked against the intent-model for intent similarity in intent_model.py, if the inputs are below the maximum confidence threshold of 0.25 then it is classified as unknown intent input and the fallback response is triggered.

2.2 Implementation

NLP Pipeline

1. Preprocessing
   The first step in the tokenization of input text, this involves the breakdown of sentences into individual tokens, this ensures that input text is ready for downstream processing such as vectorization. This is followed by stop word removal with words like "is" and "the" being filtered to focus on meaningful content, this reduces noise in the text and improves performance of intent recognition systems. Finally, the input undergoes stemming to reduce words to their root forms such as "running" → "run" paired with lemmatization which considers linguistic context to return valid root words such as "better" → "good", these are implemented as foundational preprocessing steps inspired by work in Lab 1. [7]
2. Vectorization
   The input must be represented in numerical form with vectorization method of choice being the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. This method assigns weights to tokens based on their importance reducing the impact of commonly used words while emphasizing key-terms. The vectorization process is enhanced using an extended n-gram range of (1, 3) to capture multiword phrases and richer contextual information, paired with parameter tuning like min_dif (minimum document frequency) to optimize vectorization of the training dataset. [2]
3. Classifier Training
   The logistic regression classifier is tuned with GridSearchCV to identify the optimal parameters for intent recognition accuracy. These optimizations combined with the size of the dataset and low confidence threshold for unknown classifications enable the system to handle edge cases and ambiguities effectively. [6]

Database

The system integrates SQLite as a lightweight and persistent storage solution as recommended in Lab 0 [8], implemented in the database.py file. Where the database follows a simple schema layout comprised of 3 primary tables being:

Table 1: user_info

user_id	name	history	last_intent
INTEGER, Primary Key	TEXT	TEXT	TEXT

Table 2: flight_booking

booking_id	user_id	departure_city	Destination	departure_date	return_date
INTEGER. Primary Key	INTEGER, Foreign Key Referencing user_info.user_id	TEXT	TEXT	TEXT	TEXT

Table 3: hotel_booking

booking_id	user_id	city	hotel_name	check_in_date	check_out_date	num_rooms
INTEGER. Primary Key	INTEGER, Foreign Key Referencing user_info.user_id	TEXT	TEXT	TEXT	TEXT	INTEGER

Code Design

The chatbot follows a modular architecture with each main function being isolated in its respective component with the aim of enhancing maintainability and scalability of the software. A high-level view of the components and their interaction can be viewed in the Sequence Diagram seen in Figure 1:

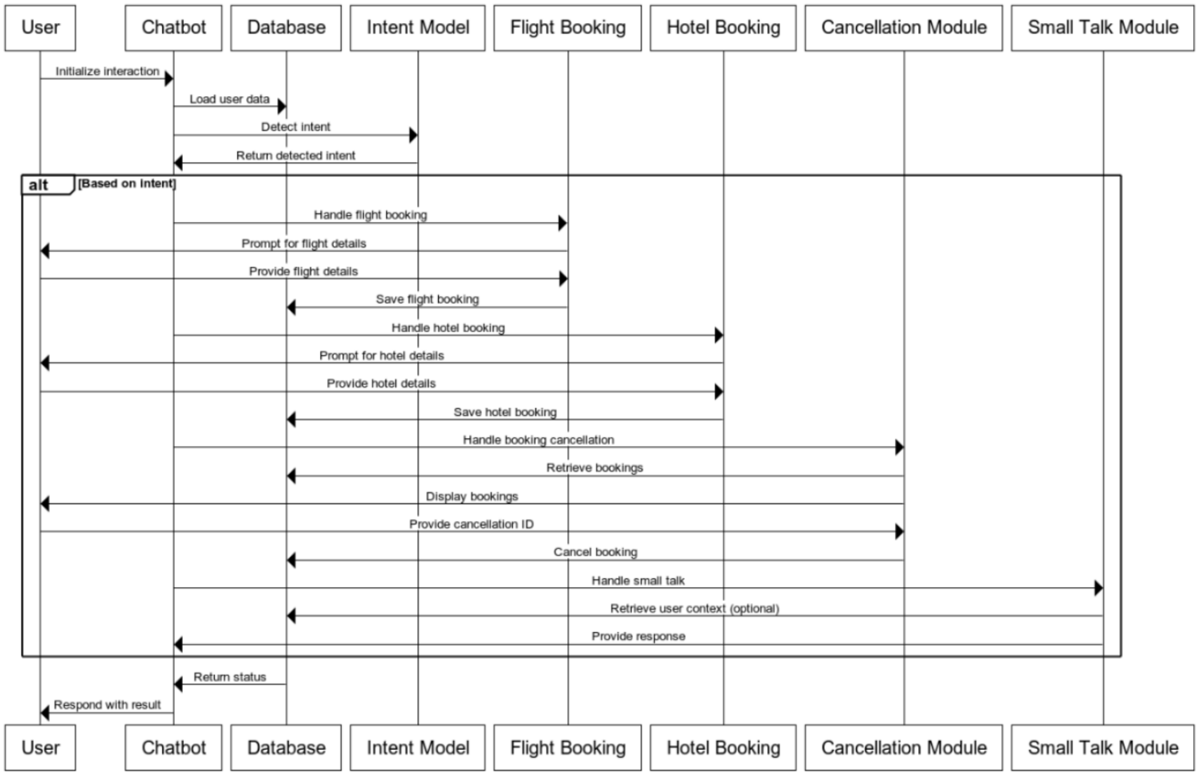

Figure 1: Sequence Diagram of Chatbot

The chatbot components are divided into 3 groups the Modular Components, Core loop and Foundational Components. The modular components hold functional aspects where each component is tasked with handling certain features, such as the intent_model.py which classifies the inputs and assigns the input to one of 9 intents handled across 6 modules such as cancellation.py, hotel_booking.py and small_talk.py. These modules are supported by foundational components such as database.py which handles database related queries and manages persistent memory of the chatbot. Finally, the chatbot features a control component called main.py which holds the control loop inspired by Lab 1's I/O loop [7] and command routing procedure, this drives the chatbots operations by taking the user input, preprocessing it, routing queries to the intent detection module all while managing the conversational flow and error handling.

2.3 Justifications

Similarity based intent matching

The decision to implement a similarity-based intent matching system via TF-IDF and Logistic regression reflects the following:

- Simplicity: This method avoids the complexity of deep learning models while still achieving high accuracy and usability
- Scalability: The chatbot will not be conducting long multiturn conversation across a wide range of possible topics, the use of a limited set of intents is sufficient for preforming the required tasks.

SQLite

This was chosen due to:

- Lightweight and efficient: Ideal for standalone application running natively on devices
- Ease of Use: Provides native python support eliminating the need for external dependencies which increase complexity of the program.
- Persistence: Enables the chatbot to data-recall regarding booking history and personalization using stored user profile data across multiple sessions.

Predefined fallback mechanisms

Fallback strategies are essential in ensuring the chatbot functioned in the intended manner complying with CUI principles through:

- Graceful Degradation: The chatbot can provide appropriate responses to ambiguous and unsupported responses in a natural manner.
- Error Logging: Extensive use of error logging improves the maintainability of the codebase by traceability for errors, allowing for comprehensive understandings of errors and their roots.

- Error Recovery: The implementation of try-except blocks and initialization checks allows for resolution of errors while the program is running, due to recovery instruction such as intent-model retraining to ensure the chatbot can function as intended regardless of minor environmental errors.

3. Conversational Design

3.1 Key Design Principles

Prompt Design

Prompt design plays a pivotal role in the effectiveness of any chat-based system, particularly in systems utilizing simpler machine learning algorithms with constrained topic breadth and comprehension. The chatbot facilitates user interactions by delivering clear, concise, and actionable prompts that guide users efficiently through tasks to achieve the intended outcome. For example:

- *"When do you want to leave for London? (Please enter the date in YYYY-MM-DD format)."*
- *"Please provide the booking ID for the reservation you wish to cancel."*

These prompts are designed aligning with the CUI principle [1] of minimizing user confusion. Additionally, the chatbot incorporates conversational markers, such as acknowledgments and affirmations like *"Got it"* and *"No problem",* which are integrated to improve conversational flow. These elements enhance the interactions natural flow improving engagement.

Discoverability and Context Awareness

Discoverability and Context awareness are key features in chatbot functionality, adding depth to dialogue through context-based suggestions improving the human-like aspect of dealing with a chatbot reducing ambiguity and improving user-experience. A main example of this is the booking suggestions:

```
Travel Agent Bot: Your flight has been booked successfully!

Travel Agent Bot: Would you like me to help you find a hotel for your trip?

You: Yes
```

The explicit suggestion of available actions keeps users informed about the capabilities of the bot in context of the action they are preforming, this ensures continuity of the conversation which is useful for multi-turn tasks such as bookings.

Error Handling

The chatbots features a robust error handling system that manages invalid and ambiguous inputs through checks and fallback replies that guides users toward correcting their input to reach their desired outcome. The first type of error correction are invalid input checks which are used for sensitive tasks like destination validation and date formats in flight bookings:

```
Travel Agent Bot: Which city are you departing from, John?

You: Narnia

Travel Agent Bot: No airports found for Narnia. Please try another city.
```

And ID number validation in cancellation tasks:

```
Travel Agent Bot: Please enter the booking ID you wish to cancel:

You: 2700

Travel Agent Bot: I couldn't find that booking ID, John.
```

The second type is fallback to conclusion, where the bot gives a general reply indicating failure to classify intent such as:

```
You: What is Burj Khalifa?

Travel Agent Bot: I'm not sure about that, but I can help you with your travel plans!
```

This guides users to rephrase to smoothly progress the dialogue and avoid unwanted errors.

Personalization

Personalization is used extensively throughout the chatbot to create a personal tone with the user, making the dialogue more natural. This is achieved through name recognition and storage as part of the users' details allowing it to be used across multiple sessions. The name is extracted through the Named Entity Recognition (NER) methods [5] and regex parsing in main.py and utils.py as such:

```
Travel Agent Bot: I don't know your name yet. What's your name?

You: Hello, my name is john doe

Travel Agent Bot: Welcome back, John!
```

This data is used throughout the dialogue such as the exit sentence:

```
You: Goodbye

Travel Agent Bot: Goodbye, John! Have a great day!
```

Confirmations

Explicit confirmation prompts are essential in chatbot design aligning with CUI practices of minimizing errors and maintaining user trust, this is especially vital in critical tasks such as bookings where the user should double-check before confirming any decision. An example of which is:

```
Travel Agent Bot: Confirm your flight details (yes/no):

You: Yes

Travel Agent Bot: Your flight booking ID is 3085.

Travel Agent Bot: Your flight has been booked successfully!
```

4. Evaluation

4.1 Usability Testing

Methodology

To evaluate the usability of the chatbot a structured user trial was conducted. This involved tasking participants with interacting with the chatbot to complete specific task including:

- Task 1: Booking a Flight and Hotel
- Task 2: Cancelling a reservation
- Task 3: Engaging in small talk

A total of 10 participants were chosen with varying levels of technical expertise for diverse perspectives. During the trials each participant was:

- Given a scenario to simulate such as *"book a flight to New York"* or *"Cancel your Hotel Booking"*
- Observations were made to measure how easily the participants completed these tasks and how effectively the chatbot guided them.

- Metrics recorded such as task success rates and errors encountered such as failure to recognize intent or provide proper response.
- After completing the task, participants rated their satisfaction and provided feedback on their experience.

The primary goal was to assess:

1. How intuitively users could interact with the chatbot
2. How accurately the chatbot interpreted the users' inputs and provided guidance

4.2 Performance Metrics

Quantitative Evaluation

Performance metrics were derived from both automated tests and user trials:

1. Intent Recognition Accuracy

   Automated tests were conducted to evaluate how accurately the chatbot recognized user intents, the models first iteration achieved a 47.06% accuracy on a test set, however this increased to 73.53% after optimization through hyperparameter training and expanding the training dataset. Despite this, the handling of edge cases involving overlapping statements and ambiguous intents remained challenging.

2. Task Completion Rates
   Booking Tasks: Users completed the bookings in 80% of attempts.
   Cancellation tasks: Cancelation had higher success rates with 90% of attempts being successful.
   Small Talk: The responses were not as consistent and diverse as participants initially though with 60% of attempts being successful on first attempt, this is due to the ambiguous nature of small talk.

Qualitative Evaluation

Participants provided qualitative feedback on their interaction with the chatbot highlighting:

1. Strengths
   a. Chatbots responses were clear, structured and concise
   b. Users appreciate the natural language style, which made the interaction feel natural
2. Weaknesses
   a. The chatbot struggled with ambiguous queries that lacked specificity such as *"can you help me?"* and *"I need assistance"*.
   b. Users noted that the bots' fallback responses were repetitive and did not provide guidance when the intent recognition failed

4.3 Results

The assessment showed that the chatbot given straightforward questions succeeded but began failing when the branches of the questions were given in an unconventional manner. Key results include:

- High Success Rates for Simple Tasks: Most of the booking and cancellation tasks were executed by the users independently, proving that the use case of the chatbot is suitable for functioning in a structured process.
- Satisfactory Small Talk Interaction: Some of the participants pointed out that the small talk response made the service more friendly enhancing user experience.
- Ambiguity as a Limitation: In such queries, there was some tendency to get it wrong or give fallback answers, proving that the management of such cases must be improved.

Areas for Improvement:

- Intent Recognition for Ambiguous Queries: Increasing the training data from the intent recognition model to incorporate the actual increases of user phrasing and, the applying of BERT or GPT-based embeddings may also help with the intent recognition of less formalized keyword queries.
- Dynamic Fallback Responses: Changing more of the fallback responses to be contextual that redirect users back on track (for example offering certain commands such as *"Would you like to book a flight?"*).
- Improved Error Handling: Using easy-to-read informational alerts in case of some intents or actions' failure.
- Ambiguity Resolution: Introducing repeat questions while working with the original message that has vague or ambiguous from the user. For example:
  - *User: "I need help."*
  - *Bot: "Sure! Would you like help with finding a flight, reservation at a hotel or do you need help with something different?*

To summarize, the evaluation process highlighted both strengths and limitations of the chatbot. While it excels at structured tasks and provides a friendly conversational experience, it requires further improvements in handling ambiguous queries and enhancing fallback mechanisms. Addressing these issues will enhance its overall usability, task success rates, and user satisfaction.

5. Discussion

5.1 Successes

The project demonstrated successes that underscore the chatbot's value as a digital travel assistant:

Modular Design: Another area of success of the chatbot was the modularity of the software. By separating the chatbot's core components, such as intent recognition, I/O loop, and database, the system achieved flexibility and scalability. This modular design allowed for debugging and updates without disrupting the system. For instance, extension of new APIs for travel services can be provided in the framework providing long term scalability.

Accurate Intent Matching: Its ability to match intents came out as a critical strength of the chatbot. Using NLP models in combination with customized training data, the system reached a suitable level of accuracy in the intent-recognition task. It was easy for users to undergo bookings, and other travel-related services. This precision greatly enhanced user satisfaction and streamlined their interaction with the chatbot.

5.2 Challenges

Despite successes, the project faced several challenges that highlighted areas for improvement:

- Edge Cases: Handling edge cases proved to be a hurdle. Users often phrased requests in unexpected ways, leading to instances where the chatbot struggled to provide accurate answers. For instance, region-specific slang and ambiguous queries fell outside the scope of the training data resulting in inaccuracies.
- Free-Form Small Talk: While the chatbot excelled at structured queries, it struggled with free-form small talk. Users engaged in casual conversations unrelated to travel. revealing the lack of extensive training data and limited system capabilities.

5.3 Future Work

To further enhance the chatbot's functionality, the following areas of future work are proposed:

Enhancing the chatbots capability of handling free-form multi-turn small talks is a priority for improving conversational depth and user engagement. This would involve greatly expanding small-talk dataset to widen topic diversity; this paired with using more advanced NLP engines would facilitate smoother and more natural conversations.

The integration of live API data would increase the chatbots utility as a travel agent and allow it benefit from massive datasets and expanded services providing a better experience to the user. However, this would require robust error handling and data synchronization operations.

In conclusion, the travel agent chatbot demonstrated impressive potential as a virtual assistant through its modular design and accurate intent matching. However, addressing challenges like edge cases and enhancing small-talk capabilities will be crucial for continued evolution. With targeted improvements, the chatbot can offer more comprehensive travel planning experiences.

6. References

[1] Rossouw, Amore & Smuts, Hanlie. (2023). Key Principles Pertinent to User Experience Design for Conversational User Interfaces: A Conceptual Learning Model. 10.1007/978-3-031-40113-8_17.

[2] Abubakar, H. D., Umar, M., & Bakale, M. A. (2022). Sentiment classification: Review of text vectorization methods: Bag of words, Tf-Idf, Word2vec and Doc2vec. SLU Journal of Science and Technology, 4(1), 27-33.

[3] LogisticRegression.(n.d.). Scikit-learn. https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] Clos, J. (2023, October). Lab 3: Text Classification [Lab work].

[5] Jeremie, Clos, & Fischer, J. (2024). Lecture 10: Advanced Topics [Slide show].

[6] GridSearchCV. (n.d.). Scikit-learn. https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GridSearchCV.html

[7] Clos, J. (2023, October). Lab 1: Modelling and Generating Language [Lab work].

[8] Clos, J. (2023). Lab 0: Preparatory work and basic string processing [Lab work].