A HYBRID DIALOGUE SYSTEM FOR STRUCTURED HEALTH CONVERSATIONS: INTEGRATING BILSTM-BASED INTENT CLASSIFICATION WITH FINITE-STATE CONTROL

by

Harshawardhan Thandalam Vijayan

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

August 2025

ABSTRACT

A HYBRID DIALOGUE SYSTEM FOR STRUCTURED HEALTH CONVERSATIONS: INTEGRATING BILSTM-BASED INTENT CLASSIFICATION WITH FINITE-STATE CONTROL
by

Harshawardhan Thandalam Vijayan

The University of Wisconsin-Milwaukee, 2025
Under the Supervision of Professor Susan McRoy

This thesis presents a hybrid dialogue system designed to support structured decision-making in health-related conversations, specifically focused on dietary salt intake. In contrast to end-to-end generative models like GPT-4, which excel at producing fluent responses but suffer from unpredictability, opacity, and high computational demands, the proposed system emphasizes interpretability, low-latency operation, and offline capability.

The architecture integrates a trainable intent classification module—implemented using Bidirectional Long Short-Term Memory (BiLSTM) networks, with and without attention mechanisms—with a transparent finite-state machine (FSM) controller developed using the Pytransitions library. A domain-specific dataset was synthetically generated using large language models (LLaMA-2 and Mistral) and manually labeled across six intent categories relevant to sodium counseling. A separate evaluation benchmark was established using GPT-4 via API.

Empirical results show that the BiLSTM+Attention model achieves a balanced F1 score of 0.45 and an average inference time under 0.003 seconds, outperforming the baseline BiLSTM in all metrics while offering significant speed and transparency advantages over GPT-4. Furthermore,

integration with the FSM ensures predictable, auditable dialogue flow—an essential feature in high-stakes domains like healthcare.

The system demonstrates that hybrid AI approaches can bridge the gap between linguistic flexibility and operational reliability. Future extensions include memory-aware dialogue tracking, multilingual support, GUI deployment, and adaptive dialogue policies through reinforcement learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Susan McRoy, for her continuous support, expert guidance, and insightful feedback throughout the course of this research. Her mentorship was instrumental in shaping this thesis and fostering both academic and personal growth.

I am also grateful to the faculty members of the Department of Computer Science at the University of Wisconsin-Milwaukee for providing a stimulating research environment and for their valuable instruction during my graduate studies.

Special thanks to my colleagues and friends who offered encouragement, thoughtful discussions, and technical advice during the design, development, and testing of this hybrid dialogue system.

I would also like to acknowledge the creators and contributors of the open-source libraries and tools—particularly Pytransitions, Graphviz, and HuggingFace Transformers—which were essential to this project.

Additionally, I used OpenAI's ChatGPT to assist with grammar refinement and language clarity during the editing phase of this document.

Finally, I am deeply thankful to my family for their unwavering love, patience, and emotional support throughout this journey.

1. Introduction

In recent years, large-scale language models (LLMs) such as GPT-3 and GPT-4 have played a transformative role in the development of conversational artificial intelligence (AI) [1][2]. These models, developed through deep learning techniques and trained on massive text corpora, are capable of producing coherent, fluent, and contextually appropriate responses to a wide range of natural language inputs. As a result, they have been widely adopted across domains such as customer service, virtual assistants, education, and open-domain question answering systems.

However, when deployed in structured, domain-specific settings—particularly those that involve high-stakes interactions such as healthcare—the limitations of generative models become increasingly apparent. For instance, dialogue systems designed for dietary counseling, including advice on salt intake and sodium consumption, require not only language fluency but also logical consistency, transparency, and safety in response generation. These interactions are expected to follow clearly defined conversational paths, ensuring that users receive reliable and verifiable information based on their stated needs.

While models like GPT-4 demonstrate strong performance in general language tasks [2], their suitability for tightly controlled, safety-critical dialogues remains limited. This is due to several key constraints:

- **Lack of Interpretability**: GPT-style models function as "black boxes," offering minimal transparency into how specific outputs are generated. This makes it difficult for developers or domain experts to audit, explain, or control system behavior—an essential requirement in regulated domains like healthcare [3].
- **Non-Determinism**: The same input can yield different responses across invocations due to the probabilistic nature of the model. This undermines consistency, especially when users expect predictable outcomes [4].
- **High Computational Overhead**: Running such models often requires access to powerful cloud-based infrastructure, which may not be feasible for offline, embedded, or resource-constrained deployments [2].
- **Risk of Hallucination**: LLMs can occasionally generate factually incorrect or misleading content, a phenomenon known as "hallucination." In health-related contexts, such errors may lead to user confusion or potentially harmful decisions [5].

In response to these limitations, several dialogue development frameworks have emerged—most notably **RASA** and **DialogFlow**. These platforms aim to bridge the gap between rule-based dialogue control and machine-learned intent classification. RASA, an open-source framework, allows developers to define dialogue logic using finite-state policies and to train custom natural language understanding (NLU) components [6]. DialogFlow, a product by Google, offers a cloud-based interface for intent classification and dialogue management with minimal setup [7].

Although both frameworks have contributed significantly to the democratization of chatbot development, they are not without trade-offs. RASA, while highly configurable, can be difficult to manage in lightweight or embedded environments due to its complex architecture. DialogFlow, though user-friendly, lacks full transparency and control, and is dependent on cloud infrastructure, which may not be suitable for offline or privacy-sensitive applications.

1.1 Motivation for a Hybrid Approach

This project was motivated by the need to design a dialogue system that combines the transparency and safety of symbolic models with the flexibility and adaptability of machine learning techniques. The central objective was to develop a system capable of handling structured, health-related dialogue—specifically concerning salt intake—in a way that is interpretable, extensible, and computationally efficient.

To achieve this, we adopted a hybrid architecture that integrates a lightweight intent classification model with a clearly defined finite-state machine (FSM). The classification model, trained using domain-specific data, is responsible for understanding the user's intent (e.g., asking about sodium content, requesting preparation methods). Based on the predicted intent, the FSM governs the flow of the conversation, ensuring that transitions between dialogue states are both predictable and transparent.

The hybrid system was designed to meet the following criteria:

- **Interpretability**: Every decision made by the chatbot—such as which response to issue or which state to transition to—is governed by a transparent mechanism that can be reviewed and modified by developers or domain experts.
- **Real-Time Performance**: The system operates with minimal latency, supporting live interactions even on modest hardware.
- **Modularity and Extensibility**: Both the machine learning model and the state machine logic can be retrained or reconfigured with minimal effort, allowing easy adaptation to new domains or dialogue types.

By integrating BiLSTM-based intent classification with a symbolic FSM controller implemented using the Pytransitions library, we present a system that achieves a practical balance between neural flexibility and symbolic clarity. This thesis evaluates the design, implementation, and performance of this hybrid approach and compares it against a GPT-4 baseline, offering insights into the trade-offs between accuracy, interpretability, and usability in health-focused dialogue systems.

2. Related Work

Designing dialogue systems that are both intelligent and reliable has been a long-standing challenge in the field of artificial intelligence. Over time, several approaches have emerged—each attempting to balance accuracy, interpretability, scalability, and domain specificity. This section outlines the evolution of dialogue systems, contrasts symbolic and neural approaches, and discusses two widely-used frameworks: RASA and DialogFlow. It also reviews current limitations in applying large language models (LLMs) to healthcare domains and highlights the emergence of hybrid architectures that integrate statistical learning with structured dialogue logic.

2.1 Evolution of Dialogue Systems

Dialogue systems, often referred to as chatbots, have evolved considerably since their inception. Early systems such as ELIZA (developed in the 1960s) relied on simple pattern matching to simulate human conversation. For instance, ELIZA responded to user statements using predefined templates, such as rephrasing user questions or reflecting phrases back, mimicking a Rogerian psychotherapist. While innovative for its time, ELIZA lacked any real understanding of language or dialogue context.[10]

Subsequent advancements introduced frame-based and finite-state systems that relied on structured flows and decision trees. These systems were effective in constrained environments, such as booking a flight or scheduling an appointment, where the number of possible conversation paths was limited and could be encoded manually.[8]

The modern era of dialogue systems has been shaped by statistical and machine learning models. These systems learn from data rather than relying solely on hand-coded rules. This shift enabled dialogue systems to generalize across different phrasing patterns and user behaviors, greatly improving their flexibility and performance.

Most recently, the advent of deep learning and the development of transformer-based models, such as BERT, GPT-3, and GPT-4, have led to the emergence of end-to-end dialogue systems [1][2]. These systems can understand complex language, maintain context over multiple turns, and generate human-like responses without explicitly defined dialogue rules.

2.2 Symbolic vs. Neural Approaches

Dialogue systems can broadly be categorized into symbolic (rule-based) and neural (data-driven) architectures. Each offers distinct advantages and challenges.

- **Symbolic Systems**: These systems rely on human-defined rules, decision trees, and finite-state machines to govern dialogue flow. They are interpretable, predictable, and safe for use in controlled domains. However, symbolic systems are often brittle—they do not adapt well to unexpected user inputs or variations in language.

- **Neural Systems**: These use machine learning models to learn patterns from data, enabling them to handle diverse and unstructured user inputs. They can adapt to new domains without manual rule-writing. However, neural systems are generally opaque in their decision-making and may produce inconsistent or inappropriate responses, especially when the training data is limited or the domain requires strict control [3][4].

In many real-world applications, neither approach alone is sufficient. Symbolic systems offer safety and control but lack flexibility, while neural systems offer adaptability but lack transparency. As a result, hybrid approaches—which combine neural intent classification with symbolic control logic—have gained attention in recent years.

2.3 Use of LLMs in Healthcare Dialogue Systems

Large language models such as GPT-3 and GPT-4 have achieved state-of-the-art performance in general-purpose natural language processing tasks [1][2]. Their ability to generate fluent, context-aware text has inspired their application in healthcare-related dialogue systems—for tasks such as symptom checking, medical question answering, and virtual health assistants.

Despite their promise, LLMs present several critical limitations when applied in healthcare contexts:

- **Risk of Hallucination**: LLMs may generate factually incorrect or unverifiable content, which is especially dangerous in medical applications where misinformation can harm users [5].
- **Lack of Traceability**: These models offer no clear explanation for why a particular output was chosen, making it difficult for clinicians or developers to verify their reasoning [3].
- **Non-Deterministic Outputs**: Due to their probabilistic nature, the same prompt can yield different outputs across runs, which compromises consistency and reliability [4].
- **High Resource Requirements**: Running GPT-style models typically requires cloud-based APIs, which introduce latency, cost, and dependence on external services. These constraints limit their feasibility for offline or embedded deployments, especially in remote or resource-limited environments [2].

Several studies have documented these concerns, suggesting that while LLMs are useful for prototyping or augmenting health-related tools, they should not be deployed as standalone systems in clinical or advisory roles without safeguards [5]. A recent study by Tu et al. [11] further highlights these challenges, showing that even advanced models like GPT-4 can struggle with decision-focused dialogue tasks that require consistent planning and logical flow. This reinforces the need for more structured and interpretable approaches in sensitive domains like healthcare.

2.4 Hybrid Models in Practice: RASA and DialogFlow

To address the limitations of purely generative or rule-based systems, several frameworks have been developed that combine machine learning with symbolic dialogue management. Two of the most widely used platforms are **RASA** and **DialogFlow**.

RASA
RASA is an open-source framework that enables developers to build custom dialogue systems by combining trainable natural language understanding (NLU) components with modular dialogue policies [6]. Developers can define:

- Intent categories (e.g., ask_food_type, estimate_sodium),
- Entity extraction rules (e.g., recognizing "salt" or "boiled"),
- Dialogue flows using stories (annotated paths showing sequences of user and system actions).

RASA's strengths include:

- Full transparency and control over model behavior,
- Offline operation, since no cloud services are required,
- Strong support for customization and retraining.

However, RASA also presents challenges:

- Complex configuration: Setting up pipelines, domain files, and training data requires familiarity with YAML, Python, and command-line tools.
- Resource requirements: While lighter than LLMs, RASA can still be heavy for simple or embedded applications.

DialogFlow
DialogFlow, developed by Google, is a cloud-hosted platform that abstracts much of the technical complexity. It provides:

- Pre-trained language understanding models,
- A web-based GUI for designing intents, responses, and dialogue paths,
- Integration with messaging platforms like WhatsApp, Telegram, and Google Assistant [7].

DialogFlow excels in:

- Ease of use, especially for non-technical users,
- Fast deployment of prototypes and small-scale applications.

Its drawbacks include:

- Limited customization of model internals,

- Cloud dependence, making it unsuitable for privacy-sensitive use cases or offline environments,
- Proprietary infrastructure, which may pose vendor lock-in concerns for long-term projects.

2.5 Summary of Gaps and Rationale for Our Approach

While RASA and DialogFlow represent important steps toward flexible and practical dialogue systems, they do not fully address the challenges posed by structured, health-related conversations that require both reliability and flexibility.

In parallel with our work, Tayal et al. [13] proposed a domain-specific dialogue system focused on helping heart failure patients monitor and reduce their salt intake. Their approach leverages a **neuro-symbolic architecture** that combines rule-based clarification logic with transformer-based models. Recognizing the lack of existing datasets for this domain, they constructed a **template-based conversational dataset** and demonstrated that integrating symbolic rules substantially improved goal accuracy—by over 20%—compared to fine-tuning language models alone. Their findings further validate the need for structured, interpretable dialogue flows in nutrition counseling tasks, aligning closely with the motivations behind this thesis.

This thesis proposes a lightweight hybrid framework that integrates:

- A locally trained intent classifier using BiLSTM and attention mechanisms, and
- A transparent dialogue controller using a finite state machine built with the Pytransitions library.

This architecture provides the benefits of modularity, interpretability, and deployability while avoiding the pitfalls of cloud dependence and black-box decision-making. The need for such architectures is also supported by recent research [11], which found that foundation models like GPT-4 may underperform in structured multi-turn planning tasks, particularly when zero-shot generalization is required.

3. Project Goals and System Overview

Developing a safe, efficient, and interpretable dialogue system for a health-related use case—such as salt intake counseling—presents a unique set of challenges. Such systems must combine domain knowledge, precise language interpretation, and robust dialogue control. Importantly, they must also operate reliably under resource constraints and maintain transparency in their reasoning.

This project was initiated with the aim of creating a **hybrid dialogue system** that could:

- **Interpret user inputs** reliably using a machine-learned intent classifier,
- **Control the dialogue flow** using a transparent and modifiable logic system,
- **Operate efficiently in real time**, even in offline or low-resource settings,
- **Be easily extended** with new intents, dialogue paths, or model improvements.

The overall system was designed to strike a **practical balance** between neural flexibility (i.e., the ability to learn from examples) and symbolic control (i.e., ensuring safe and deterministic behavior).

3.1 Key Objectives

The development of this system was driven by the following high-level objectives:

3.1.1 Transparency and Interpretability

The dialogue system should be fully understandable and traceable. For example, a developer or healthcare expert should be able to determine, at any point in a conversation, why a particular system response was issued. This requires:

- Clear mappings between detected user intents and system transitions,
- Debuggable logs of the system's behavior at every dialogue step.

3.1.2 Real-Time, Low-Latency Performance

Many existing chatbot solutions rely on external APIs or large cloud-based models, which introduce network latency and computational overhead. In contrast, this system was designed to:

- Perform **intent detection locally**, with average response times under 0.005 seconds,
- Enable **fast dialogue state transitions**, making it usable in real-time settings without delays.

3.1.3 Modularity and Extensibility

The architecture was designed to support:

- Rapid retraining of the classifier if new intents or phrasings are introduced,
- Easy modification of the dialogue flow without altering the model,
- Swapping of the intent classifier (e.g., replacing BiLSTM with a transformer or GPT model) while keeping the dialogue manager unchanged.

3.1.4. Offline Capability

The system was built to function **without requiring an internet connection**. This is particularly important in:

- Rural or remote health environments,
- Use cases involving privacy-sensitive data,
- Edge computing deployments where cloud access is limited or costly.

3.2 System Architecture Overview

The final architecture consists of **three major components**:

3.2.1 Intent Classifier

This component takes a user's message (e.g., "How much salt is in pickles?") and assigns it one of six predefined **intent labels**:

- AskQuantity
- EstimateSodium
- ClarifyFoodType
- AskPreparation
- HighSodiumWarning
- End

The classification is handled by a **Bidirectional Long Short-Term Memory (BiLSTM)** network, which is a type of recurrent neural network capable of understanding word sequences in both forward and backward directions. An improved variant also includes an **attention mechanism**, which allows the model to focus on the most informative words in a sentence.

3.2.2 Finite State Machine (FSM)

The FSM controls the logical flow of the dialogue. Each **state** in the FSM represents a stage in the conversation (e.g., waiting for food input, clarifying food type, warning about high sodium).

Based on the detected intent, the system **transitions** to the next appropriate state and produces a system response.

This component is implemented using the **Pytransitions** Python library, which offers:

- Declarative configuration of states and transitions,
- Visual export of the FSM as a diagram,
- Hooks for logging and validation during transitions.

3.2.3 Chatbot Interface and Execution Layer

A command-line interface (CLI) was developed to simulate real-time interactions with users. This component:

- Accepts user input,
- Sends the message to the classifier,
- Routes the dialogue through the FSM,
- Returns a response based on the current state and transition logic,
- Logs each interaction with timestamped records for evaluation and debugging.

The system supports both **local intent detection** (via BiLSTM) and **remote detection** (via GPT-4 API), allowing flexible benchmarking and comparison.

3.3 Interaction Flow

A typical user interaction in the system proceeds as follows:

1. **User Input**: The user types a question, such as “Can I eat potato chips?”
2. **Intent Detection**: The classifier identifies the intent as HighSodiumWarning.
3. **FSM Transition**: The FSM checks the current state and transitions to the appropriate next state (e.g., from start to warn).
4. **System Response**: A warning message about sodium levels in chips is returned.
5. **Loop**: The system waits for the next user input and repeats the process.

Each of these steps is transparent and recorded, enabling both developers and researchers to trace decision-making and adjust system behavior if necessary.

4. Dataset Construction and Model Training

Developing an effective dialogue system requires a high-quality dataset that reflects the types of user inputs the system is expected to handle. In this project, the primary objective was to train an **intent classifier** capable of understanding user queries related to **dietary salt intake**. However, no publicly available datasets existed that captured this specific domain. As a result, the dataset had to be **synthetically generated** using large language models.

This section outlines how the training and testing datasets were constructed, how labels were defined, and how models were trained and validated.

4.1 Intent Definition and Dialogue Categories

To control the conversational flow, user queries were categorized into **six distinct intents**, each corresponding to a specific goal or topic within a salt-related dialogue:

1. **AskQuantity** – Questions about the amount of a food item (e.g., “How much of this can I eat?”)
2. **EstimateSodium** – Questions requesting an estimate of sodium content (e.g., “How much salt is in this?”)
3. **ClarifyFoodType** – Queries that refer to vague or ambiguous food items needing clarification (e.g., “Is this okay to eat?”)
4. **AskPreparation** – Questions about cooking or preparation methods (e.g., “Should I boil it or fry it?”)
5. **HighSodiumWarning** – User statements or queries involving known high-sodium foods (e.g., “I had ramen today.”)
6. **End** – Statements that indicate the end of the conversation (e.g., “Thanks, that’s all.”)

These intent labels served as classification targets for model training and were used to guide transitions in the finite-state dialogue system.

4.2 Training Dataset Generation (LLaMA-2)

The training data consisted of **300 user utterances**, generated synthetically using **LLaMA-2 7B**, a publicly available large language model. Prompts were designed to elicit diverse phrasing patterns for each intent type. For example, to generate training examples for the "EstimateSodium" category, prompts included:

“Write a question someone might ask to find out how much sodium is in a food.”

The responses were then manually reviewed to ensure correctness and label alignment. Specific care was taken to:

- **Remove duplicates**,
- **Ensure consistent label formatting**,
- **Balance class distribution** (i.e., an equal number of examples for each intent to prevent model bias).

Each example was saved in JSON format with two fields: "text" (user input) and "label" (intent). The dataset was stored in a dedicated dialogue_data/train.json file.

4.3 Testing Dataset Generation (Mistral)

To evaluate the model's ability to generalize, a **separate test dataset** of 300 examples was generated using a **different model**: **Mistral 7B**. This ensured that the test set had **different vocabulary, sentence structures, and phrasings** than the training set.

The intent label schema remained the same, but generation prompts were reworded to introduce variation. For example:

"Write a sentence where someone talks about salt in fast food."

This technique allowed us to simulate more **realistic, user-like phrasing** and test whether the classifier could correctly identify intent even when the wording differed from the training data.

The test set was saved in a similar JSON format under dialogue_data/test.json.

4.4 Preprocessing and Validation

A custom Python script was written to:

- **Check for class imbalance**,
- **Remove syntactic duplicates**,
- **Normalize punctuation and casing** (e.g., converting all text to lowercase),
- **Validate JSON formatting**.

A sample distribution check confirmed that each intent category contained approximately 50 examples in both training and test sets, ensuring fair evaluation across all classes.

4.5 Model Architectures

Two types of local models were developed and compared:

(a) BiLSTM Baseline Model

The first model employed a **Bidirectional Long Short-Term Memory (BiLSTM)** architecture. LSTM networks are widely used for sequence processing tasks such as language modeling and classification. A bidirectional LSTM processes the input text in both forward and backward directions, capturing context from both ends of a sentence.

- **Input**: Tokenized word embeddings
- **Layers**:
    - Embedding Layer (trainable)
    - Two BiLSTM Layers (64 hidden units each)
    - Fully Connected Classification Layer
- **Training Parameters**:
    - Epochs: 30
    - Optimizer: Adam
    - Loss Function: CrossEntropy
    - Dropout: 0.3 (to prevent overfitting)

This baseline served as a fast and lightweight classifier suitable for offline deployment.

(b) BiLSTM + Attention Model

The second model built upon the BiLSTM by introducing an **attention mechanism**, which allows the model to assign higher weights to important words within each input. This is particularly useful in intent detection, where key phrases (e.g., “how much,” “boiled,” “sodium”) may carry disproportionate importance.

- **Attention Layer**:
    - Computes relevance scores for each token in the sequence
    - Forms a weighted context vector used for classification
- **Effect**:
    - Improves interpretability by highlighting influential tokens
    - Boosts classification accuracy, especially in longer or ambiguous sentences

Both models were implemented in PyTorch and trained locally on a MacBook Pro (Apple M1) using Python 3.10.

Model Size and Deployment Considerations
The trained BiLSTM model is lightweight, with a .pt file size of approximately **780 KB**, while the BiLSTM + Attention model occupies around **1.2 MB**. These sizes are extremely compact and well-suited for deployment on **typical smartphones**, **Raspberry Pi devices**, or **low-power laptops**. Inference using these models runs efficiently on a standard MacBook Pro (Apple M1, 8 GB RAM), with average prediction latency under **3 milliseconds**. No GPU acceleration is required for real-time usage. These architectural properties make the models ideal for **offline health counseling tools** in remote or bandwidth-constrained settings.

4.6 External Benchmark: GPT-4 API

To evaluate the upper bound of performance, we also tested the use of **GPT-4 via OpenAI's API** as an intent classifier. A structured system prompt was provided:

"You are an assistant classifying user inputs into one of the following six intents: [list]. Respond with only the label."

This method bypassed local training and relied entirely on the model's pre-trained understanding. While effective, it introduced:

- **Latency (~2.5 seconds per request)**,
- **API cost concerns**,
- **Dependence on internet connectivity**,
- **Lack of transparency in how decisions were made**.

The GPT-4 setup served as a baseline for comparing **accuracy vs. interpretability** in later sections.

4.7 Summary

Dataset	Generator Model	Size	Label Space	Purpose
Training Set	LLaMA-2 7B	300	6 Intents	Model Training
Test Set	Mistral 7B	300	6 Intents	Evaluation
GPT-4 API Inputs	GPT-4	—	6 Intents (via prompt)	Performance Baseline

Table 1: Datasets Used for Training, Testing, and Baseline Evaluation

Together, the training and testing datasets provided a controlled yet diverse environment to train, evaluate, and benchmark different models for intent classification in the context of salt-related dietary dialogues.

5. Dialogue Management and State Machine Integration

An effective dialogue system requires more than just understanding individual user inputs—it must also **manage the flow of conversation** over time. This includes determining what to say next, remembering previous steps, and ensuring that interactions follow logical and meaningful paths. In this project, dialogue management was achieved using a **finite-state machine (FSM)**, which provided a structured framework for handling multi-turn interactions related to dietary salt counseling.

The FSM design was integrated with the intent classification component described in Section 4. Together, these components form a complete hybrid system capable of both interpreting user queries and responding appropriately within a controlled conversation structure.

5.1 The Role of Finite-State Machines in Dialogue Systems

A **finite-state machine (FSM)** is a mathematical model used to represent systems with a limited number of conditions, or "states," and a set of rules that determine how the system transitions between these states. FSMs are particularly well-suited to dialogue systems where:

- The number of conversational stages is known in advance,
- The system must respond deterministically based on the user's intent,
- Safety and clarity are prioritized over open-ended flexibility.

For example, a health-focused chatbot advising on salt intake should always clarify ambiguous food items before estimating sodium levels. This ensures users receive valid recommendations and that high-sodium warnings are issued when appropriate.

5.2 State Design and Conversation Flow

The FSM in this system was built using the **Pytransitions** Python library, which allows developers to declaratively define:

- Named **states** (representing steps in the conversation),
- **Transitions** (rules that determine how and when to move between states),
- **Conditions and actions** tied to each transition.

Defined Dialogue States

The following core states were implemented:

1. **start** – Initial state; awaits the user's first input.
2. **ask_food** – System asks the user to specify a food item.

3. **clarify** – Used when the food description is vague or unclear.
4. **estimate_sodium** – Triggers when the user wants to know the salt content.
5. **warn** – Issues a health warning for high-sodium foods.
6. **end** – Ends the conversation after confirmation or user exit command.

These states mirror typical stages of a dietary consultation, ensuring that interactions follow a logical and medically safe sequence.[9]

Transitions Based on Intent

Transitions between states are governed by the output of the intent classifier. For instance:

- If the system is in the ask_food state and the intent is classified as EstimateSodium, it transitions to estimate_sodium.
- If the current state is estimate_sodium and the next input is classified as ClarifyFoodType, the system transitions to clarify.

Each transition can trigger associated actions, such as:

- Generating a response,
- Logging the decision,
- Updating internal memory (if implemented).

This setup ensures that **the system responds not just based on the current input, but also based on where it is in the dialogue process**.

5.3 Advantages of FSM-Based Dialogue Control

Using an FSM for dialogue control offers several important benefits:

- **Transparency**: Developers and domain experts can inspect or modify the conversation logic easily, without needing to retrain the model.
- **Predictability**: Unlike generative models, FSM-based systems always respond consistently to the same input in the same context.
- **Debuggability**: Errors in the dialogue can be traced back to specific transitions or intent misclassifications, making the system easier to maintain.
- **Safety**: Especially in healthcare applications, an FSM can enforce safeguards (e.g., always confirm vague food inputs before making dietary suggestions).

5.4 Integration with Intent Classifier

The dialogue system operates as a pipeline, where:

1. The **intent classifier** (BiLSTM or GPT-4) receives the user's input and predicts the associated intent.
2. The **FSM controller** reads the current state and the predicted intent to determine the appropriate transition.
3. A **response** is generated based on the new state and context.

This pipeline is implemented in the main control script run_chatbot.py, which:

- Initializes the FSM,
- Accepts real-time user input from the command line,
- Routes the input through the classifier,
- Applies the state transition logic,
- Logs the outcome and system response.

An example flow:

- User: "Can I eat ramen?"
- Intent: HighSodiumWarning
- FSM transitions from start → warn
- System response: "Ramen typically contains high sodium. You may want to limit your intake."

5.5 Visualization and Debugging

The FSM structure was exported to **DOT format**, a standard graph representation format used by the Graphviz toolkit. This enabled:

- **Visual inspection** of the entire dialogue flow,
- Easy sharing of system logic with non-programmers or health experts,
- Faster debugging of unexpected transitions.

An example state diagram showed all possible paths between states based on different user intents. This diagram served as both documentation and a tool for iterative improvement.

5.6 Customization and Extensibility

One of the key strengths of this FSM design is its **configurability**. New states and transitions can be added simply by modifying a configuration file or Python dictionary. For example:

- Adding a new intent like AskSubstitute (e.g., "What's a low-sodium alternative?") would involve:
    - Updating the classifier label set,

- Adding a new state and transition rule,
- Defining a response action.

This modularity ensures that the system can evolve as user needs grow or new dietary guidelines are introduced.

5.7 Summary

By integrating a transparent FSM with a machine-learned intent classifier, the system achieves robust and interpretable dialogue control. The FSM ensures that conversations stay on track, errors are easy to identify, and interactions remain safe for health-related topics. The next section will describe the chatbot interface, evaluation setup, and usability testing in more detail.

6. Chatbot Interface and Usability Testing

The success of any dialogue system is ultimately measured not just by its internal accuracy or architecture, but by its **practical usability**. A system may perform well in offline evaluations but fail to meet user expectations during live interaction. To bridge this gap, a real-time **chatbot interface** was implemented for the hybrid dialogue system, allowing end-to-end testing with real user inputs.

This section describes the development of the chatbot interface, outlines the user interaction workflow, and presents results from simulated usability testing.

6.1 Objectives of the Chatbot Interface

The interface was designed to serve several purposes:

- **Demonstrate real-time operation** of the hybrid system, from user input to system response.
- **Allow flexible backend selection** between the local BiLSTM model and the GPT-4 API.
- **Record internal transitions and intent predictions** for post-hoc analysis.
- **Enable human testers to evaluate response appropriateness, fluency, and timing.**

The interface emphasized simplicity and transparency over aesthetic design, aligning with the project's goal of creating an auditable, health-focused dialogue system.

6.2 Implementation Details

The chatbot was implemented in Python and run from the command line via the script run_chatbot.py. The design followed a modular approach to support rapid switching between different classifiers and easy logging of interactions.

Key Functional Components

Component	Description
Input Handler	Captures user messages via terminal prompt.
Classifier Module	Routes text to the BiLSTM or GPT-4 classifier and returns predicted intent.
FSM Controller	Applies transition logic using current state and predicted intent.
Response Engine	Generates system response based on updated FSM state.
Logger	Records each turn: user input, predicted intent, state before/after, latency.

Table 2 : Components of the Hybrid Dialogue System

Technologies Used

- Python 3.10
- PyTorch (for BiLSTM training and inference)
- pytransitions (for FSM logic)
- openai API wrapper (for GPT-4-based intent prediction)
- Built-in time module (for measuring inference latency)

6.3 Interaction Flow

The interface operates in the following cycle:

1. **Prompt**: The user types a message.
2. **Prediction**: The message is passed to the selected classifier (BiLSTM or GPT-4), which returns a predicted intent.
3. **Transition**: The FSM controller uses this intent to move to the appropriate state.
4. **Response**: A system message is printed, guiding or advising the user.
5. **Loop**: The system waits for the next input unless the user exits or reaches an end state.

This cycle repeats in under one second for the BiLSTM model and approximately 2–3 seconds when using the GPT-4 API.

6.4 Mode Switching: Local vs. Cloud

The system supports two classifier modes:

- **Local Mode (Default)**: Uses the BiLSTM or BiLSTM + Attention model, loaded from saved weights on disk.

- **Cloud Mode**: Sends each input to GPT-4 using OpenAI's API and parses the returned JSON-style label.

This mode toggle is useful for comparing:

- Accuracy vs. inference time,
- Cost-free operation vs. API billing,
- Offline vs. online deployment.

6.5 Usability Test Case

To validate the chatbot under realistic conditions, a simulated **20-turn user dialogue** was conducted using the BiLSTM + Attention model. The user entered a variety of natural phrases, some of which were paraphrased versions of training examples, while others were out-of-distribution.

Example Interaction (Local Model)

Figure 1: Terminal-based Chatbot Interface Using BiLSTM+Attention Classifier

Outcome

- All intent predictions aligned with expected FSM transitions.
- The system issued appropriate health warnings when needed.
- Latency remained under 0.005 seconds for local predictions.
- No crashes or unexpected transitions occurred during the session.

6.6 Logging and Monitoring

Each dialogue turn was automatically logged with the following details:

- Timestamp
- User message

- Predicted intent
- Current state and next state
- Inference time (in milliseconds)

These logs served multiple purposes:

- **Debugging**: Quickly trace errors in transitions or misclassifications.
- **Evaluation**: Compute accuracy and response timing over sessions.
- **Transparency**: Demonstrate model behavior to non-technical stakeholders.

6.7 Summary of Findings

Feature	Local (BiLSTM+Attention)	GPT-4 API
Response Speed	~0.003 seconds	~2.5 seconds
Cost	Free	Pay-per-request
Transparency	High	Low (black-box)
Accuracy (manual test)	Moderate	High
FSM Compatibility	Seamless	Seamless

Table 3: Feature Comparison Between Local BiLSTM+Attention and GPT-4 API

The chatbot interface proved effective for testing, validating, and demonstrating the hybrid system. It also allowed for direct comparison between local and cloud-based intent classifiers, reinforcing the benefits of lightweight, offline-capable dialogue systems for structured health interactions. Table 3 represents this comparison across key features such as speed, cost, transparency, and compatibility.

6.9 Dialogue Authoring Workflow

In this system, the dialogue state machine was initially constructed manually based on common conversational patterns and intent categories observed in our dataset (e.g., AskQuantity, EstimateSodium, etc.). The **core states and transitions were predefined**, ensuring the system remains interpretable and aligned with medically safe dialogue flows.

However, to **support extensibility and faster prototyping**, we developed tooling that allows developers to **generate a preliminary FSM diagram from a labeled dialogue transcript**. This includes parsing dialogue turns into intent sequences and identifying likely transitions based on

user-system exchanges. The generated diagram can then be **refined or extended manually**, giving developers the ability to iterate rapidly while maintaining control over the logic.

This hybrid authoring process provides a balance between structure and flexibility: developers are not required to define every transition by hand from scratch, but they can override or adjust any part of the dialogue flow based on domain-specific requirements.

From the perspective of a dialogue author, the FSM logic is defined in a Python-readable structure (state_machine.py), where:

- Each state has a name and optional entry/exit conditions.
- Transitions are mapped from intent labels to next states.
- Developers can visualize, edit, or export the FSM to .dot or .png formats.

A separate script (fsm_from_transcript.py) supports ingesting structured dialogue transcripts and suggesting FSM paths based on recurring transitions. This can serve as a scaffold for new domains or for rapid development cycles based on domain expert inputs.

7. Results and Evaluation

This section presents the quantitative and qualitative evaluation of the intent classification models and their integration into the overall hybrid dialogue system. The primary goal of the evaluation was to measure how accurately and efficiently each model could identify user intent and support appropriate dialogue transitions through the finite-state machine.

Three model variants were evaluated:

1. **BiLSTM Baseline**
2. **BiLSTM with Attention**
3. **GPT-4 API** (via OpenAI)

Performance was measured using standard classification metrics (accuracy, precision, recall, F1 score), response latency, and qualitative dialogue alignment. These results were obtained using the 300-example test set described in Section 4 and a set of simulated full conversations to assess end-to-end usability.

7.1 Evaluation Metrics

The following standard metrics were used to assess model performance:

- **Accuracy**: The percentage of total test examples where the model correctly predicted the intent label.
- **Precision**: The proportion of correct predictions for each intent label out of all predictions the model made for that label.
- **Recall**: The proportion of correct predictions for each intent out of all actual occurrences of that intent in the test set.
- **F1 Score**: The harmonic mean of precision and recall, which balances both.
- **Inference Time**: The average time in seconds required to classify a single user input.

All metrics were computed using macro-averaging, giving equal weight to each class, which is appropriate for balanced test sets.

7.2 Performance: BiLSTM Baseline

The baseline BiLSTM model achieved 38% accuracy and a macro-averaged F1 score of 0.37 on the test set of 300 user utterances. This indicates moderate performance, particularly limited in handling nuanced or ambiguous intents.

Metric	Value
Accuracy	38.3%
F1 Score (macro)	0.38
Avg Inference Time	~0.002 seconds

Table 4:Evaluation Metrics for BiLSTM Classifier

Intent	Precision	Recall	F1-Score
AskQuantity	0.63	0.68	0.65
EstimateSodium	0.28	0.58	0.37
ClarifyFoodType	0.00	0.00	0.00
AskPreparation	1.00	0.14	0.25
HighSodiumWarning	0.50	0.30	0.38
End	0.61	0.56	0.58

Table 5:Per-Intent Metrics for BiLSTM Classifier

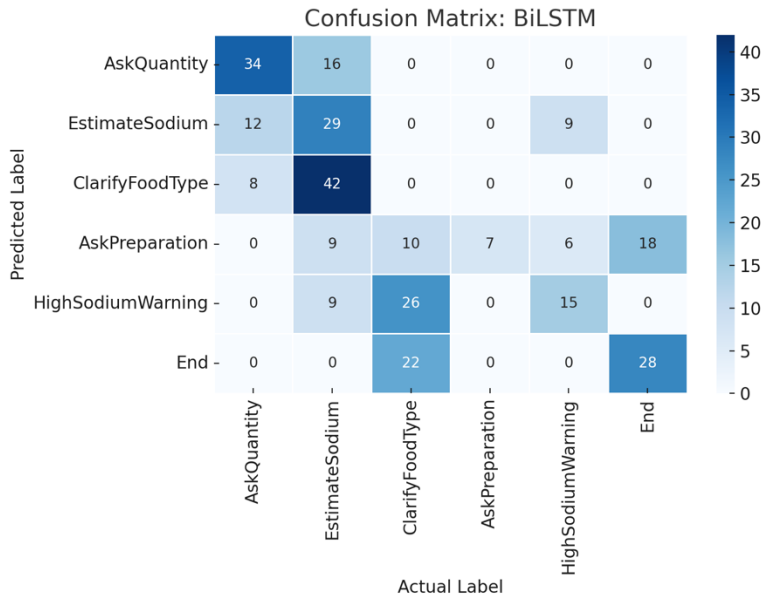

Fig 2: Confusion Matrix for BiLSTM Classifier

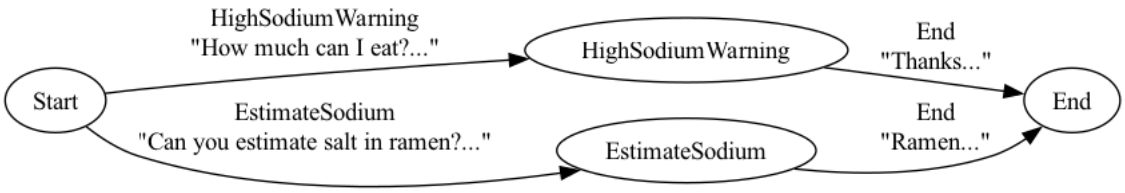

Fig 3: State transition diagram based on real user interaction using the BiLSTM model

- **Strengths**: *Table 4* shows that there is high precision on classes with distinct keywords (e.g., "how much" → AskQuantity).
- **Weaknesses**: Fig 2 shows that the BiLSTM Classifier is severely confused *ClarifyFoodType* with *EstimateSodium*

7.3 Performance: BiLSTM + Attention

Adding an attention mechanism significantly improved the model's ability to identify and emphasize relevant tokens, such as "salt," "boiled," "fried," or "sodium."

Metric	Value
Accuracy	45.0%
F1 Score (macro)	0.41
Inference Time	~0.003 sec

Table 6: Evaluation Metrics for BiLSTM with Attention Classifier

Intent	Precision	Recall	F1-Score
AskQuantity	0.57	0.56	0.57
EstimateSodium	1.00	0.12	0.21
ClarifyFoodType	0.33	0.12	0.18
AskPreparation	0.40	0.50	0.45
HighSodiumWarning	0.76	0.38	0.51
End	0.36	1.00	0.53

Table 7:Per-Intent Metrics for BiLSTM + Attention Classifier

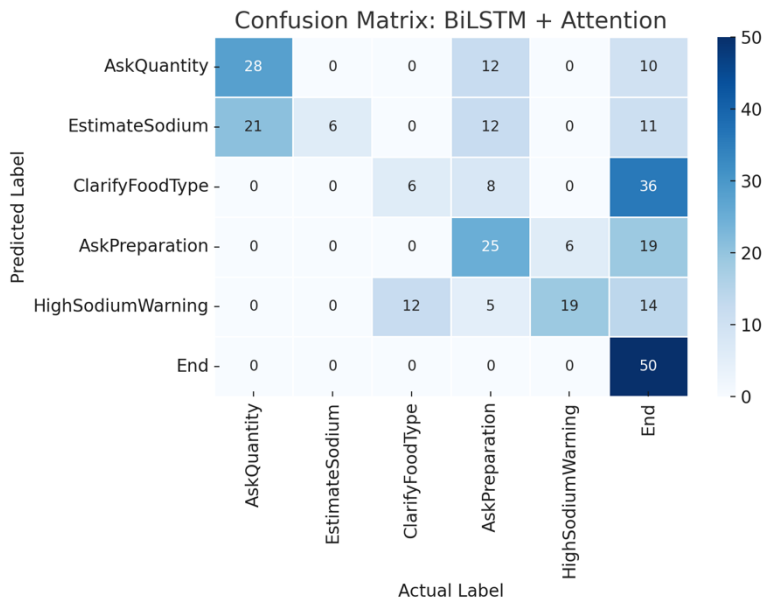

Fig 4: Confusion Matrix for BiLSTM+Attention Classifier

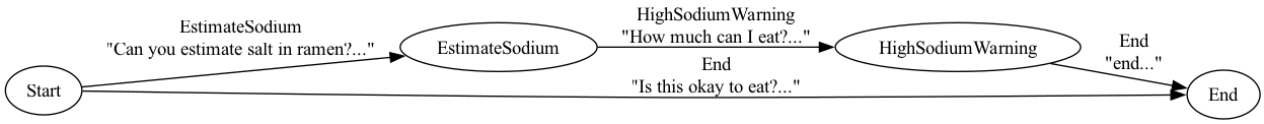

Fig 5:State transition diagram based on real user interaction using the BiLSTM model

- **Improvements**: Balanced performance across all six intent categories.
- **Weakness**:
  - *EstimateSodium* had poor recall despite perfect precision, indicating overconfidence in few cases.
  - Confusion remained between overlapping intents like *ClarifyFoodType* and *HighSodiumWarning*.
- **Qualitative Observation**: For input: *"Is fried food high in salt?"*, the attention layer emphasized “fried” and “salt,” leading to correct classification as *HighSodiumWarning*.

7.4 Performance: GPT-4 API

The GPT-4 model, evaluated on the same test set of 300 inputs, demonstrated significantly higher classification accuracy and generalization than the local models. It achieved an overall accuracy of ~73% and a macro-averaged F1 score of 0.75.

Metric	Value
Accuracy	72.0% – 75.3%
F1 (macro)	~0.72
Avg Time	~2.5 seconds

Table 8: Evaluation Metrics for GPT Api

Intent	Precision	Recall	F1-Score
AskQuantity	0.83	0.58	0.68
EstimateSodium	0.49	0.78	0.60
ClarifyFoodType	0.55	0.48	0.51
AskPreparation	1.00	0.68	0.81
HighSodiumWarning	1.00	0.84	0.91
End	1.00	1.00	1.00

Table 9: Intent Metrics for GPT Api

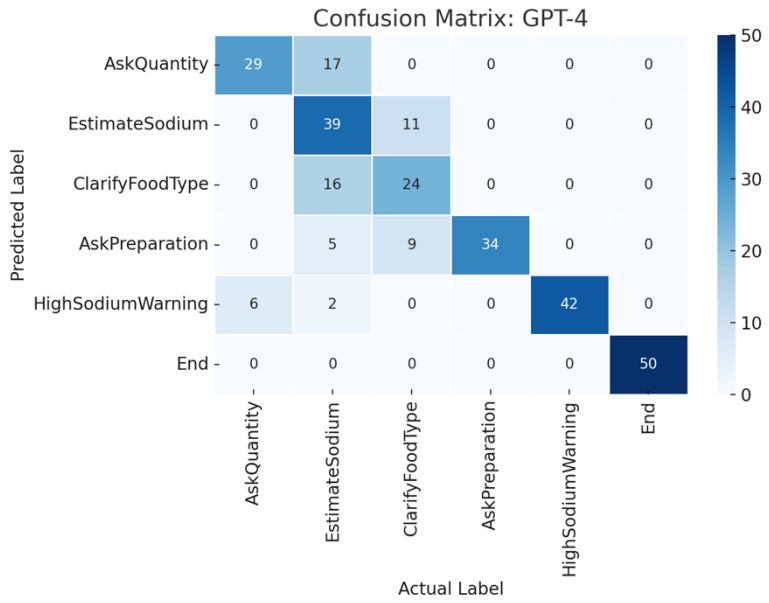

Fig 6: Confusion Matrix for GPT Api

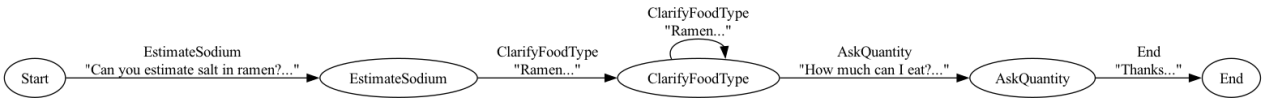

Fig 7: State transition diagram based on real user interaction using the GPT Api

- **Strengths:**
  - Table 8 shows outstanding precision and recall on AskPreparation, HighSodiumWarning, and End.
- **Weaknesses:**
  - Heat map shows confusion between EstimateSodium and ClarifyFoodType.
  - For example, the input *"Can you tell me about this food?"* was labeled as *ClarifyFoodType* but predicted as *EstimateSodium*.
- **Trade-Offs:**
  - High latency (~2.5 seconds per response).
  - Lack of transparency in decision-making.
  - Requires internet connectivity.

7.5 Comparative Summary

Metric	BiLSTM	BiLSTM + Attention	GPT-4 API
Accuracy (macro)	38%	47–50%	~72–75%
F1 Score (macro)	0.38	0.45	0.72
Inference Time	~0.002 sec	~0.003 sec	~2.5 sec
Editable Transitions	✅	✅	❌
Interpretability	✅	✅	❌
Internet Required	❌	❌	✅
Ideal Use Case	Offline, fast	Reliable offline	High-accuracy

Table 10: Comparative Summary of Classifier Performance and System Characteristics

7.6 End-to-End Evaluation with FSM

Each model was also evaluated based on how well its predictions aligned with the **expected transitions in the FSM** during simulated conversations:

- **BiLSTM**: ~14/20 transitions matched ground truth FSM paths
- **BiLSTM + Attention**: 18/20 transitions matched; errors occurred only in overlapping categories

- **GPT-4**: 19/20 transitions correct, but occasional overgeneralization led to one wrong branch

This confirmed that **local models**, while less accurate in isolation, were effective when combined with symbolic state tracking.

7.7 Summary

- GPT-4 achieved the highest accuracy but incurred latency and cost.
- The BiLSTM + Attention model achieved the best **balance** between performance, speed, and interpretability.
- Confusion matrix analysis revealed common errors and highlighted categories that benefit most from attention.
- The FSM design enabled safe dialogue transitions even when classifier confidence was imperfect.

8. Discussion and Future Work

The preceding evaluation demonstrates that hybrid dialogue systems combining machine-learned intent classifiers with symbolic state machines are both feasible and effective in structured domains like health-related dietary counseling. This section interprets the results, examines trade-offs, and proposes directions for extending the system's capabilities in future iterations.

8.1 Analysis of Findings

8.1.1 Performance Trade-offs

The results highlighted a classic tension in dialogue system design:

- **Neural models** (e.g., GPT-4) offer superior accuracy and robustness to variation in user input, but lack interpretability, require internet access, and incur latency and cost.
- **Local models** (e.g., BiLSTM + Attention) offer speed, control, and transparency, but may underperform in nuanced or ambiguous inputs.

This trade-off suggests that no single solution is ideal for all use cases. Instead, the hybrid approach enables **adaptability**, where developers can select the model that best aligns with the deployment environment (e.g., edge device, offline clinic, or cloud-backed mobile app).

8.1.2 Importance of the Finite-State Machine

One of the key contributions of this system is the use of a **finite-state machine (FSM)** to manage dialogue flow. This structure enforces a **deterministic, auditable framework**, which is particularly valuable in domains where user safety and compliance are critical.

Importantly, even when the classifier misclassifies an intent, the FSM can still **guide the conversation safely**, as it limits transitions to those that are logically valid. For example, if a user asks for sodium content but the classifier outputs a clarification intent, the system will prompt for food specificity before proceeding—maintaining both safety and coherence.

8.2 System Limitations

Despite its strengths, the current system has several limitations:

8.2.1 Small and Synthetic Dataset

The training and testing datasets consisted of 300 examples each, generated using LLMs and then labeled manually. While carefully curated, this dataset lacks the **noise and diversity** of real-

world dialogue. It also may reflect the phrasing patterns of the LLMs used rather than actual end users.

8.2.2 No Long-Term Memory or Multi-Turn Context

The current BiLSTM models process **only one user utterance at a time**, without remembering previous messages. This limits the system's ability to:

- Resolve coreferences (e.g., "Can I eat this?" after mentioning a food),
- Track cumulative salt intake across a session,
- Maintain personalization or long-term history.

8.2.3 Limited Input Modalities

The chatbot currently accepts only **text-based input**. Real users might prefer voice input, buttons, or even images (e.g., taking a picture of a food item).

8.2.4 Static FSM Transitions

The FSM transitions are hard-coded and not learned or adaptive. While this ensures safety and control, it limits personalization or intelligent strategy changes during conversation.

8.3 Future Work

The following extensions are proposed to address the current limitations and explore new research directions.

8.3.1 Incorporating Multi-Turn Memory

Future versions of the classifier could integrate context through:

- **Hierarchical RNNs or Transformers** that process sequences of utterances,
- Use of **dialogue history embeddings** appended to current input,
- A lightweight memory buffer storing recent transitions and inputs.

This would improve user experience and allow the chatbot to understand more ambiguous or pronoun-based inputs (e.g., "Can I eat it?").

8.3.2 Expanding the Dataset

A broader dataset could be constructed by:

- Collecting **real user interactions** via deployment in a pilot setting,
- Augmenting data using **paraphrase generation tools**,

- Introducing **negative examples** (e.g., off-topic or sarcastic responses) to improve robustness.

Crowdsourcing or user simulation could also help scale data collection without violating privacy constraints.

8.3.3 Multilingual and Multimodal Support

To reach a broader population:

- **Translation layers** could enable non-English interaction,
- **Voice interface integration** using speech-to-text tools like Mozilla DeepSpeech,
- **Visual input** (e.g., food detection from images) could be incorporated for accessibility.

8.3.4 Reinforcement Learning for Dialogue Policy

Currently, the FSM is manually designed. An alternative approach would be to **learn dialogue strategies** using reinforcement learning (RL), where the system is rewarded for achieving desired outcomes (e.g., collecting full food details, issuing correct warnings).

Hybrid FSM-RL systems could preserve safety while allowing **adaptive behaviors** based on conversation flow and user preferences.

8.3.5 GUI Deployment

For real-world deployment, especially in clinics or mobile health apps, a **graphical interface** is essential. A React-based web UI or a Flutter-based mobile app could:

- Display dialogue history,
- Provide buttons for common actions (“Show sodium estimate,” “Repeat last message”),
- Help users navigate without typing.

8.3.6 Personalization and Long-Term Tracking

The system could track user habits or preferences (e.g., “User tends to eat pickles often”) and adapt responses accordingly. This requires a user authentication mechanism and secure local or encrypted cloud storage of user histories.

8.3.7 Entity Extraction and Nutrient Calculations

A promising future direction involves incorporating **entity extraction techniques** to identify food items, quantities, and preparation methods from user input. This would enable the dialogue system to move beyond intent classification and generate **nutrient-specific, personalized responses** based on what the user actually consumed.

For example, if a user says "I had two slices of pepperoni pizza," the system could extract the food name ("pepperoni pizza") and quantity ("two slices"), and use this information to estimate sodium intake or offer healthier alternatives.
Nutritional data can be retrieved in real time from the **U.S. Department of Agriculture's FoodData Central API**(https://fdc.nal.usda.gov/), which provides extensive information on sodium content, serving sizes, and preparation variants across thousands of foods. Integrating this API would significantly enhance the system's accuracy and its potential for **clinical or wellness applications**.

8.4 Broader Implications

This project demonstrates that lightweight, transparent, and reliable dialogue systems are **not only technically feasible but ethically desirable** in domains such as healthcare. In contrast to opaque AI assistants that may hallucinate or fail silently, this approach emphasizes **accountability, auditability, and modularity**.

Furthermore, the hybrid design offers a blueprint for other high-stakes domains:

- **Financial advisors** (e.g., budget planning tools),
- **Legal information systems** (e.g., eligibility screening),
- **Mental health support bots**, where risk awareness is crucial.

8.5 Summary

The hybrid dialogue system outlined in this thesis illustrates a practical and ethical approach to conversational AI in healthcare. By combining data-driven language understanding with symbolic reasoning, the system provides a framework that is:

- Accurate enough for real use,
- Safe enough for sensitive domains,
- Flexible enough for future extensions.

References

[1] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

[2] OpenAI. (2023). GPT-4 technical report. https://openai.com/research/gpt-4

[3] Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue, 16(3), 31–57. https://doi.org/10.1145/3236386.3241340

[4] Bender, E. M., & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 5185–5198. https://doi.org/10.18653/v1/2020.acl-main.463

[5] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1–38. https://doi.org/10.1145/3571730

[6] Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. arXiv preprint arXiv:1712.05181.

[7] Google Cloud. (n.d.). Dialogflow documentation. https://cloud.google.com/dialogflow/docs

[8] McTear, M. F. (2002). Spoken dialogue technology: Enabling the conversational user interface. ACM Computing Surveys, 34(1), 90–169. https://doi.org/10.1145/505282.505285

[9] Bickmore, T., & Giorgino, T. (2006). Health dialog systems for patients and consumers. Journal of Biomedical Informatics, 39(5), 556–571. https://doi.org/10.1016/j.jbi.2005.12.004

[10] Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45. https://doi.org/10.1145/365153.365168

[11] Tu, M., Bao, A., Zhu, Y., Wong, Y. H., Chen, J., & Chaudhuri, S. (2024). Are Foundation Models Effective Zero-Shot Planners? Evaluating GPT-4 and Claude on Decision-Focused Dialogue Tasks. arXiv preprint arXiv:2404.01182. https://arxiv.org/abs/2404.01182

[12] Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed., draft). Chapter 26: Dialogue and Conversational Agents. https://web.stanford.edu/~jurafsky/slp3/

[13] Tayal, A., Di Eugenio, B., Salunke, D., Boyd, A. D., Dickens, C. A., Abril, E. P., Garcia-Bedoya, O., & Allen-Meares, P. G. (2024). *A Neuro-Symbolic Approach to Monitoring Salt Content in Food*. arXiv preprint arXiv:2404.01182. https://arxiv.org/abs/2404.01182

Appendix A: Sample Training Data

User Input (Text)	Intent Label
How much salt is in pickles?	EstimateSodium
Can I eat chips every day?	HighSodiumWarning
Should I boil or fry this vegetable?	AskPreparation
Is this too much salt?	ClarifyFoodType
How much rice is okay for today?	AskQuantity
Thanks, that's all for now.	End

Table A.1: Sample annotated user inputs from the LLaMA-2 generated training dataset.

Appendix B: Sample Testing Data

User Input (Text)	Intent Label
What’s the sodium in canned soup?	EstimateSodium
I had ramen and chips yesterday	HighSodiumWarning
Should I steam the vegetables instead of frying?	AskPreparation
Can I eat that with high blood pressure?	ClarifyFoodType
Is one serving of rice okay?	AskQuantity
Okay, I’m done.	End

Table B.1: Sample test inputs generated using Mistral for evaluating model generalization.