Hybrid Chatbot

CIS 5590 – Topics in Computer Science (AGI)

Professsor name: Pei Wang,

Team members name: Belede Shiva Chandan,

Patel Maulesh Kalyan,

Patel Harshkumar Vinodbhai

1. Abstract

The system brings together three artificial intelligence principles which unite large language models (LLMs) with fuzzy logic and symbolic logic implemented through Prolog. A web-based interface aims to recreate multiple types of cognition through integration of neural reasoning and symbolic reasoning and probabilistic reasoning. The chatbot executes query responses through LLaMA-3 models hosted by Groq as well as performs linguistic uncertainty evaluation by scikit-fuzzy while using SWI-Prolog (accessed through pyswip) for logical inference. The combined system demonstrates how separate AI approaches generate human-level reasoning by working together.

2. Introduction

AI systems have experienced rapid progress as neural networks control natural language operations but symbolic AI maintains proficiency in dealing with rule-based logic. Fuzzy logic serves as an advantage between absolute reasoning methods by enabling the representation of uncertain reasoning systems. This project evaluates methods through which chatbot systems can combine their abilities regarding generative conversations and uncertainty detection with symbolic logical reasoning. The Flask web application allows user messages to experience parallel processing through a large language model (LLM) in addition to fuzzy logic engines and Prolog-based logic modules..

3. System Design Overview

Within its structure the chatbot system uses a three-layered reasoning framework. The system uses independent processing within different layers for identical input and merges their outputs for presentation to users through a web interface. A three-step reasoning system makes up the logic framework for the following system:

- **Groq Language Model (LLM)**: The system implements a large language model that connects to the Groq API for operation. The system creates contextual natural language responses which derive from patient input. This part demonstrates neural statistical processing functions.
- **Fuzzy Logic Engine**: The scikit-fuzzy library powers this module to evaluate linguistic uncertain statements entered by users according to triangular fuzzy membership functions. The implementation process grants membership values to expressions like "maybe" or "probably" to determine the uncertainty classification levels with triangular fuzzy sets.
- The Prolog Reasoning Engine functions through SWI-Prolog accessed through pyswip to apply logical rules for symbolic reasoning.
  The architecture operates as follows:

An input submission from the user leads to data being transferred to a backend server that utilizes Flask technology. The system sends the communicated input to every reasoning component. The server merges outputs from all modules before sending the prepared response to display on the web page.

A layered design structure enables the chatbot to operate through the combination of three AI methods including statistical fluency from LLM and probabilistic ambiguity detection through fuzzy logic and rule-based inference thanks to Prolog.

4. Reasoning Modules
   4.1 Neural Language Model (Groq API)
   The LLM module supports user input transmission to the Groq API through which it receives responses utilizing the llama3-8b-8192 model. Here's the Python code:

```
from groq import Groq
import os
client = Groq(api_key=os.getenv("GROQ_API_KEY"))

def basic_groq_response(prompt):
  chat_completion = client.chat.completions.create(
    messages=[
      {"role": "system", "content": "You are a helpful assistant."},
```

```
            {"role": "user", "content": prompt}
        ],
        model="llama3-8b-8192"  # Replace with your desired model
    )
    return chat_completion.choices[0].message.content
```

4.2 Fuzzy Logic Engine

We use skfuzzy to assess uncertainty based on keyword scoring. If the input contains words like “maybe” or “probably”, it’s scored higher on an uncertainty scale from 0–10.

```
import numpy as np
import skfuzzy as fuzz

def fuzzy_response(user_input):
    uncertain_words = ["maybe", "probably", "not sure", "possibly"]
    score = sum(word in user_input.lower() for word in uncertain_words)

    x = np.arange(0, 11, 1)
    fuzziness = fuzz.trimf(x, [0, 5, 10])

    fuzzy_value = fuzz.interp_membership(x, fuzziness, score)

    if fuzzy_value > 0.7:
        return "High Uncertainty detected."
    elif fuzzy_value > 0.3:
        return "Moderate Uncertainty detected."
    else:
        return "Low Uncertainty."
```

5. Web Application Interface

App.py

```
from flask import Flask, request, render_template
from fuzzy_logic import fuzzy_response
from logic_reasoning import logic_response
```

```
app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    response = ""
    if request.method == "POST":
        user_input = request.form["user_input"]
        groq_reply = basic_groq_response(user_input)
        fuzzy_reply = fuzzy_response(user_input)
        logic_reply = logic_response(user_input)

        response = f"Groq: {groq_reply}\nFuzzy Logic: {fuzzy_reply}\nProlog Logic:
{logic_reply}"
    return render_template("index.html", response=response)

if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hybrid AI Chatbot</title>
</head>
<body>
    <h1>Hybrid AI Chatbot</h1>
    <form method="post">
        <input type="text" name="user_input" placeholder="Ask me anything..."
style="width: 400px;">
        <input type="submit" value="Send">
    </form>
    <h2>Response:</h2>
    <pre>{{ response }}</pre>
</body></html>
```

After executing command python app.py . There you will get a url link in terminal, just copy that url link and paste in browser. So that you can interact with hybrid_chatbot and it will give response to your questions. I have already pasted some screenshots of hybrid_chatbot responses at last in the report.

6. Technologies Used
   - Python 3.12
   - Flask (web framework)
   - Groq API (LLM)
   - scikit-fuzzy (fuzzy logic)
   - SWI-Prolog + pyswip (symbolic logic)
   - HTML/CSS (interface)
   - dotenv (for secure key management)

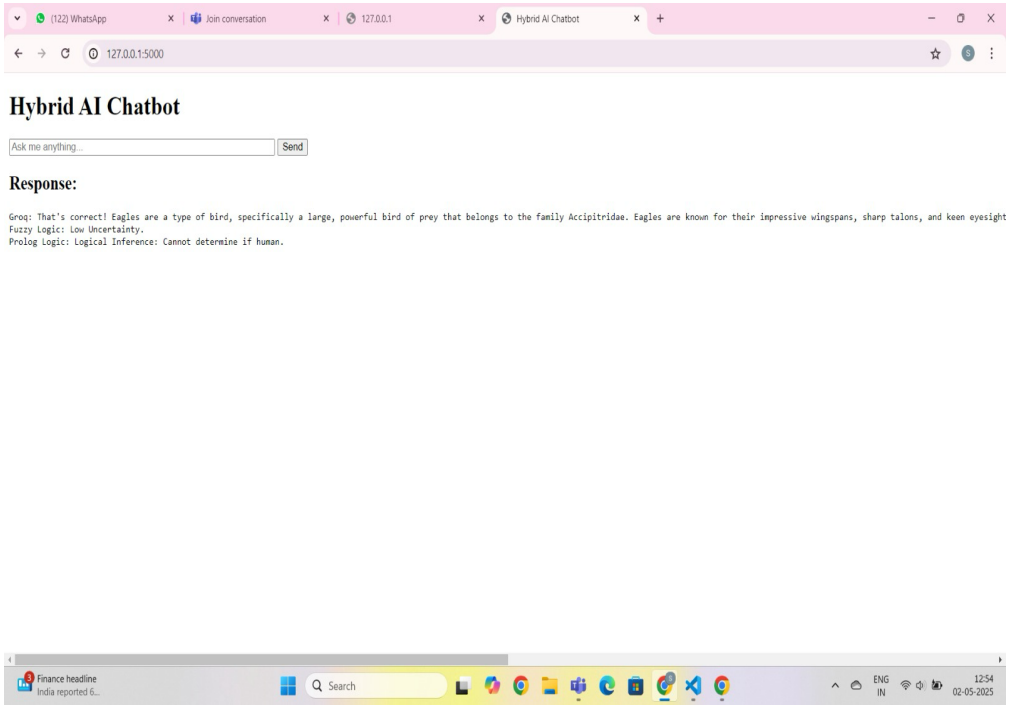
7. Challenges Encountered
   - Users must set SWI_HOME_DIR correctly while adding swipl.exe to their PATH environment variable.
   - Secure key loading with .env files was mandatory to set up the Groq API.
   - Python 3.12 users faced dependency problems when installing scikit-fuzzy.
   - The process of matching asynchronous LLM results to deterministic logic outputs

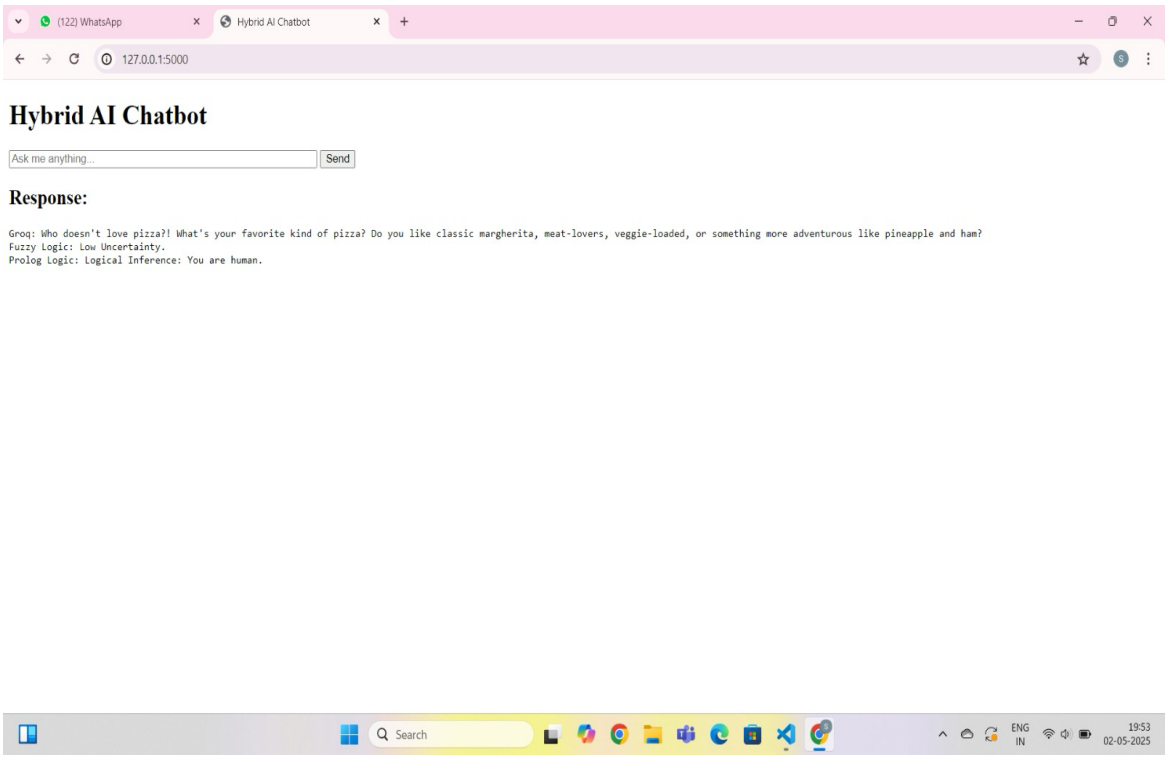
8. Future Work
   - The system should save user interactions for later use in follow-up processing.
   - Users can enhance LLM responses by adjusting its parameters for specific domains.
   - The system requires an addition of meta-reasoning capability to both rank and synthesize the three responses.
   - The system should operate in the cloud environment with voice and vision integration capabilities.

9. Conclusion

This project combines three reasoning systems through neural, fuzzy and symbolic elements to shape a chatbot. The reasoning systems operate together by filling in the gaps of one another to create an expanded framework of intelligent interaction. The prototype demonstrates a possible emergence of AGI from combining different reasoning systems instead of developing a single comprehensive architecture.

10. Results(Output):




(122) WhatsApp
Hybrid AI Chatbot
127.0.0.1:5000
Hybrid AI Chatbot
Ask me anything...
Send
Response:
Groq: Who doesn't love pizza?! What's your favorite kind of pizza? Do you like classic margherita, meat-lovers, veggie-loaded, or something more adventurous like pineapple and ham?
Fuzzy Logic: Low Uncertainty.
Prolog Logic: Logical Inference: You are human.
Search
ENG
IN
19:53
02-05-2025