MID-REPORT

CS-5368 Intelligent Systems Project Report
Spring 2023.

Project: Develop an Intelligent Agent

SUBMITTED BY:

James Ballari

Kuljeet Kaur

Soundarya Samala

Tishya Sohankumar Thakkar

Table of Contents

Introduction

GitHub Links

GitHub Repository – https://github.com/sosamala/CS5368-project-2023Spring

GitHub Page – https://sosamala.github.io/CS5368-project-2023Spring/

Problem Statement

The project aims to build a web application hosted on GitHub that offers intelligent behaviour for assisting users in selecting restaurants based on their preferences. The application domain focuses on food/restaurants and involves non-trivial automated reasoning through ASP/SPARC rules. The software should be able to understand and respond to English-spoken or written questions. The interface of the software should be user-friendly, easy to navigate, and appealing to the end user. The application should be able to provide recommendations based on user preferences and interests, location, and budget. The goal of the project is to deliver an intelligent software that provides personalized recommendations to users based on their unique needs and interests, thus making the process of selecting a restaurant easier and more enjoyable.

Application Domain

The application domain of this project revolves around food and restaurants. People have diverse dietary needs and preferences and often struggle to find the perfect restaurant that caters to their specific requirements. This project aims to assist users in selecting the best restaurant that suits their taste buds, and budget. The application will consider various factors such as the preferred location, cuisine, and budget of the restaurant to provide personalized recommendations. This addresses a common problem that people face when trying to find a suitable restaurant that meets their needs. We aim to solve this problem by creating an intelligent agent that will help the users to select a restaurant by answering their queries.

Intelligent Agent

The intelligent agent in this project is designed to assist users in selecting a restaurant based on their preferences. The agent utilizes non-trivial automated reasoning through ASP/SPARC rules to provide personalized recommendations to the user. The agent is equipped with a database of restaurants, which includes information such as the restaurant's menu, location, pricing, and ratings. The agent will use regex to understand and respond to user's queries in English, either through spoken or written questions. The agent will then apply reasoning techniques to analyse the user's inputs and provide relevant recommendations based on their preferences. The goal of the intelligent agent is to provide a seamless and intuitive experience for the user while delivering personalized restaurant recommendations based on their unique needs and interests.

Problem Description

Background knowledge

There are different restaurants each with a unique id. Each restaurant has a rating, price range, category, and menu items. The rating of a restaurant is a score that indicates the quality of the food and service provided by the restaurant, ranges from one to five stars. The price range of a restaurant refers to the cost of the menu items, ranging from low ($) to high ($$$). The category of a restaurant describes the type of cuisine served, such as Italian, Chinese, or Mexican. The menu items of a restaurant include various dishes and drinks that are offered each with a corresponding category, price and description.

Questions

Here are some of the questions that you can ask the agent and the expected response:

1. Suggest a random restaurant.
   Response: Name of a Random restaurant.
2. Show Top Rated Restaurants
   Response: List of top-rated restaurants.
3. Suggest Restaurants by [$$] ($ - Budget Friendly $$$ - Expensive)
   Response: List of restaurant for a given budget.
4. What are restaurants in [zipcode]
   Response: List of restaurants in a given zipcode.
5. Show me restaurants that are serving [Category]
   Response: List of restaurants that serves a given category.
6. Search for [Restaurant Name]
   Response: Details of a given restaurants such as cuisine, price range, menu items, ratings.

In this prototype we have worked on suggesting a restaurant – option 1 from above. The rest and many other functionalities will be added in the final submission.

Techniques

Web Technologies:

- ### HTML

Used HTML to create webpages for the Restaurant intelligent agent.

- ### JavaScript (jQuery, Ajax)

Used JavaScript and jQuery framework to create logic and to add various controls for the web site as listed below.

1. Process text/speech from user and translate it to the required SPARC query and respond with text/speech.

We have used the web speech API[1] to synthesize the responses of the user and also get the text input from user from voice commands. Then we have developed a multi-layer text processing function which first processes the input from user based on the context and expected response, if that fails then checks for the Regex Pattern for various questions and checks if the input matches any of those and processes the request accordingly, if that also fails then we check the input against a collection of strings to check the best word count match using percentage of hits to determine if the input matches any of those, if that also fails then we respond with a fall back text.

Once we get a regex, or pattern matched in the above process, in some cases we may make a query request to SPARC to get results for which we have used Ajax (Step 3) web requests and once we get the corresponding response from Sparc we again used regex to process the corresponding variables/unknowns, extract the required data from the parameter query results and translate the extracted data into a sentence to respond the user.

2. Various User interface actions such as keyboard and button click actions have been mapped to override defaults and perform a certain action, like enter key on text input send the message, the toggle for voice input/output etc.
3. We have used Ajax (Asynchronous JavaScript and XML) web technology to create asynchronous web applications and requests to pass the Sorts, Predicates, Knowledge/Rules along with the required query to http://wave.ttu.edu/ajax.php via the https://cors-anywhere.herokuapp.com/ to overcome the CORS(Cross-origin resource sharing) issue when accessing the ASP SPARC features from wave.ttu.edu through github.io domain. However, to enable this, one must request temporary access to the demo server at *https://cors-anywhere.herokuapp.com/corsdemo*.

- ### CSS(BOOTSTRAP)

We have used Bootstrap and other custom CSS style sheets to make the User Interface more appealing to the user.

ASP: SPARC

- ### Data source extraction Hashing required fields using Python

We have used a python script to generate unique Hash values for various fields of the Restaurant Data Source so that they can be used as value names for the ‘sorts’ in ASP program which are unique strings mapping to the real-world String data stored in the JavaScript maps. Since many values start with numeric or capital letters or have special characters etc, in them the hash is used instead to establish the relationships and other rules for different sorts whereas the JavaScript map has the actual values. The hash is an alphabet unique string code which can be used as names for different sorts.

- ### Generating relationships using SQLite DB queries

We have used the restaurant data source values in SQLite database and wrote queries to quickly generate the required SPARC sort definition to data mappings between objects. We have normalized/or rearranged the data where applicable in the restaurant data source to improve the performance of the DB queries and quickly generate the SPARC relationships which are straight forward.

- ### Default Values

We have written Sparc rules to generate default values wherever necessary and writer other rules to enhance the object relationships in SPARC.

GitHub

We have hosted the required source code and data source to GitHub repository and hosted the web application in GitHub [3].

References

1. *Using the web speech API - web apis: MDN*. Web APIs | MDN. (n.d.). Retrieved April 1, 2023, from https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API
2. Liyuan-Gao. (n.d.). *Liyuan-Gao/asp_example: A simple example of web application for ASP program*. GitHub. Retrieved April 1, 2023, from https://github.com/liyuan-gao/ASP_example
3. "Creating a GitHub Pages site," *GitHub Docs*. https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site (accessed Apr. 02, 2023).