Informe Lab2 – Prolog

Nombre: Víctor Duarte Arce.

Rut: 21.467.246-4

Tabla de contenidos:

Introducción:

El problema consiste en desarrollar un sistema para la creación, despliegue y administración de “chatbots” estructurados, específicamente “chatbots” de Respuesta de Interacción a Texto (ITR). Estos “chatbots” responden a preguntas basadas en opciones específicas y no involucran procesamiento avanzado del lenguaje natural ni inteligencia artificial. Los estudiantes deben implementar un ambiente que permita:

- Crear un entorno contenedor de “chatbots”.
- Crear “chatbots” individuales.
- Identificar “chatbots”.
- Agregar preguntas y opciones a los “chatbots”.
- Especificar enlaces para dirigir la interacción entre “chatbots”.
- Interactuar con los “chatbots” a través de opciones y palabras clave.
- Ofrecer una síntesis de las interacciones.

Los “chatbots” pueden estar interconectados, de modo que las opciones de un “chatbot” puedan dirigir a un usuario a otro “chatbot”. Los estudiantes deben cumplir con requisitos mínimos obligatorios y realizar autoevaluaciones de su trabajo.

Descripción del problema:

El problema consiste en la creación de un ambiente administrador de chatbots, el cual permita al usuario crear, modificar y hablar con un chatbot. También debe poder ofrecer una síntesis de lo escrito con el chatbot.

Los chatbots deben ser de ITR (Respuesta de Interacción con Texto), estos chatbots se caracterizan por ser muy estructurados, por responder pregunta en base a decisiones específicas (Paradigmas de Programación 2023).

Descripción del paradigma (Declarativo Lógico):

La base del paradigma declarativo lógico son las relaciones, las cuales se definen como proposiciones que pueden ser verdaderas o falsas.

“Un programa escrito en un lenguaje declarativo como Prolog, puede usarse como una especificación de un problema, en lugar de establecer pasos para llegar a la solución de un problema” (Flores V. 2021).

La programación dentro de este paradigma consiste en 3 etapas: Base de conocimientos: Aquí se agregan todos los hechos que se conocen del problema; reglas: En esta etapa se crean reglas de la forma consecuente :- antecedente; consulta: En esta etapa se consultan los predicados, en el caso de Prolog, a través de su intérprete.

(Flores V. 2021).

Análisis del problema:

Objetivos del Sistema:

1. **Crear un ambiente contenedor de chatbots:** Los usuarios deben poder crear un entorno que albergue múltiples chatbots.
2. **Crear chatbots individuales:** Se deben poder crear chatbots específicos, cada uno con su conjunto de opciones y preguntas.
3. **Identificar chatbots:** Los usuarios deben poder identificar y distinguir entre los chatbots disponibles en el sistema.
4. **Agregar preguntas y opciones:** Para cada chatbot, los usuarios deben poder agregar preguntas y definir opciones para que el chatbot responda a las preguntas con las opciones proporcionadas.
5. **Especificar enlaces para la interacción:** Deben poder definir cómo los chatbots interactúan entre sí, permitiendo que las opciones de un chatbot dirijan a los usuarios a otro chatbot.
6. **Interactuar con los chatbots:** Los usuarios deben poder interactuar con los chatbots, ya sea seleccionando opciones específicas o utilizando palabras clave que coincidan con las opciones.
7. **Ofrecer una síntesis de las interacciones:** Al finalizar la interacción con un chatbot, el sistema debe proporcionar una síntesis de la conversación y la acción tomada.

Las características clave son: Los chatbots son estructurados y no implican procesamiento avanzado del lenguaje natural o inteligencia artificial, los chatbots pueden estar interconectados, permitiendo una conversación fluida y dirigida por opciones, el sistema debe ser flexible y permitir la creatividad de los usuarios en la configuración de los chatbots y sus interacciones.

Diseño de la solución:

Enfoque de Solución:

La solución se basa en la descomposición y la abstracción del problema a través de distintos TDAs basados en listas. A continuación, se muestran más detalles de cómo se implementó la descomposición y sus razones.

Descomposición del Problema:

La solución se descompone en los siguientes componentes clave:

- **TDA System:** Este TDA consiste en una lista compuesta de los siguientes TDAs:
    - **name (string):** Nombre del sistema.
    - **InitialChatbotCodeLink (int):** ID del chatbot de bienvenida.
    - **chatbots:** Lista de 0 o más chatbots.
    - **users:** Lista de 0 o más usuarios (user: string).
    - **chatHistory:** Lista de 0 o más historiales de usuario (userHistory: usuario y registro de conversaciones).
    - **actCID (int):** ID del chatbot en conversación actual.

    - **actFId (int):** ID del flujo en conversación actual.
- **TDA Chatbot:** Es una lista compuesta por los siguientes TDAs:
    - **chatbotID (int):** Identificador del chatbot en el sistema.
    - **name (string):** Nombre del chatbot.
    - **welcomeMessage (string):** Mensaje de bienvenida.
    - **startFlowId (int):** ID del flujo de inicio.
    - **flows:** Lista de 0 o más flujos.
- **TDA Flow:** Es una lista compuesta por los siguientes TDAs:
    - **id (int):** Identificador del flujo en un chatbot.
    - **name-msg (string):** Nombre del flujo (puede ser una pregunta).
    - **options:** Lista de 0 o más opciones.
- **TDA Option:** Es una lista compuesta por los siguientes TDAs:
    - **code (int):** Identificador de la opción en un flujo.
    - **message (string):** Mensaje en pantalla (ejemplo: "1) Chile").
    - **ChatbotCodeLink (int):** ID del chatbot al que se enviará.
    - **InitialFlowCodeLink (int):** ID del flujo al que se enviará.
    - **Keywords:** Lista de 0 o más palabras clave.

En resumen, la descomposición del problema se basó en estructuras de listas (ver Figura 3), ya que a pesar de que estas puede que no sean la estructura más eficiente de todas, son las más fáciles e intuitivas de usar.

Algoritmos y Técnicas:

Para la solución, se usaron los siguientes algoritmos basados en listas:

- **Búsqueda secuencial:** se usó para verificar la existencia de elementos en una lista (Figura x).

```
option_member_list(O, [H|_]):-
    get_Option_code(H, C1), get_Option_code(O, C2), C1 = C2.
option_member_list(O, [_|T]):- option_member_list(O, T).
```

  Figura 1: "option_member_list(O, L)." Elaboración propia.

- **Filtración de elementos repetidos:** este algoritmo usó la búsqueda secuencial para filtrar elementos repetidos de la siguiente manera (Figura 2).

```
filter_users([], []).
filter_users([H|T], R):- user_member_list(H, T), filter_users(T, R), !.
filter_users([H|T], [H|R]):- filter_users(T, R), !.
```

  Figura 2: "filter_users(UsersIn, UsersOut)." Elaboración propia.

También se utilizaron las siguientes técnicas:

- **Descomposición de problemas:** Para abordar cada problema planteado, se dividió en varios subproblemas. Por ejemplo, al enfrentarnos al desafío de crear un flujo que no contenga opciones repetidas, fue necesario resolver el subproblema de filtrar una lista de opciones para asegurar que ninguna se repitiera. A su vez, esta tarea implicó resolver otro

subproblema: reconocer si una opción pertenece a una lista según su identificador (ver Figura 1).

- **Abstracción de la solución:** Para implementar un ambiente generador de chatbots, se abordó el problema de una forma abstracta, pensando en distintos TDAs, como por ejemplo el TDA Option, el cual tiene distintas operaciones que pueden ser utilizadas para la creación de la solución.

Recursos Empleados:

- **Lenguaje de Programación Prolog:** Se utiliza Prolog como el lenguaje principal para implementar el sistema de chatbot debido a su capacidad de manejar estructuras de datos y predicados de manera eficiente.
- **Estructuras de Datos**: Se emplean listas para representar los option, flow, chatbots y system.

Aspectos de la implementación:

Para la implementación de este proyecto, se utilizó el lenguaje de programación interpretado Prolog, ya que este es un lenguaje basado en el paradigma lógico y accesible.

El intérprete usado fue el de SWI-Prolog.

Estructura del proyecto:

Para implementar los distintos TDAs, se utilizaron varios archivos, como se muestra en la Figura 3, cada recuadro en rojo implicó la creación de un nuevo archivo.

Bibliotecas empleadas:

Se utilizó la biblioteca: “library(lists): List Manipulation” de Prolog. “Esta biblioteca proporciona predicados básicos comúnmente aceptados para la manipulación de listas en la comunidad Prolog.” (Traducción al español de SWI-Prolog -- Library(Lists): list manipulation, s. f.). El uso de esta biblioteca se justifica, ya que toda la solución está basada en el uso de listas.

Instrucciones de uso

Paso 1: Ejecución del Programa

Para ejecutar el programa, simplemente abra un entorno de Prolog y consulte el archivo TDASystem_214672464_DuarteArce.pl

Paso 2: Creación del Entorno

Para crear el entorno usted debe definir las options, los flows, los chatbots y el system que utilizará en este respectivo orden, ya que el system contiene muchos chatbots, que a su vez contienen flows, que a su vez contienen options.

Para definir las options, los flow, los chatbots y el system, debe consultar los predicados dejando una variable al final de cada uno para poder almacenarlos, luego separar por el conector “,”.

Ejemplo:

option(1, "1) Viajar", 2, 1, ["viajar", "turistear", "conocer"], OP1), option(2, "2) Estudiar", 2, 1, ["estudiar", "aprender", "perfeccionarme"], OP2), option(3, "3) Comer", 2, 1, ["comer", "aprender"], OP3), flow(1, "flujo1", [OP1, OP2], F10), flowAddOption(F10, OP3, F11), chatbot(0, "Inicial", "Bienvenido\n¿Qué te gustaría hacer?", 1, [F11], CB0), system("Chatbots Paradigmas", 0, [CB0], S0), systemAddUser(S0, "user1", S2), systemAddUser(S2, "user2", S3), systemAddUser(S3, "user3", S5), systemLogin(S5, "user1", S7), systemLogout(S7, S9), systemLogin(S9, "user2", S10).

Paso 3: Interacción con el Sistema

Para interactuar con el sistema, tiene los siguientes predicados:

- systemTalkRec(SystemIn, Message, SystemOut).
- systemSynthesis(SystemIn, User, String).
- systemSimulate(SystemIn, MaxInteractions, Seed, SystemOut).

Ejemplo:

systemTalkRec(S10, "hola", S11), systemTalkRec(S11, "1", S12), systemTalkRec(S12, "1", S13), systemTalkRec(S13, "Museo", S14), systemTalkRec(S14, "1", S15), systemTalkRec(S15, "3", S16), systemTalkRec(S16, "5", S17), systemSynthesis(S17, "user2", Str1), systemSimulate(S3, 5, 32131, S99), systemSynthesis(S99, "user", Str2), write(Str2).

Si quiere ver un ejemplo concreto de uso vea el anexo.

Posibles errores:

Un posible error que podría aparecer en el uso de este sistema es el uso de variables repetidas, ejemplo: “option(1, "1) Viajar", 2, 1, ["viajar", "turistear", "conocer"], OP1), option(2, "2) Estudiar", 2, 1, ["estudiar", "aprender", "perfeccionarme"], OP1).” Esto arrojará “false”, debido al paradigma utilizado en la implementación.

Resultados y autoevaluación

Predicado	Grado de alcance (0-1)	Descripción
option	1	El sistema es capaz de crear opciones
flow	1	El sistema es capaz de crear flows
flowAddOption	1	El sistema es capaz de agregar opciones si es que no existen ya en el flow
chatbot	1	El sistema es capaz de crear chatbots
chatbotAddFlow	1	El sistema es capaz de agregar flows si es que no existen ya en el chatbot
system	1	El sistema es capaz de construir systems.
systemAddChatbot	1	El sistema es capaz de agregar chatbots si es que no existen ya en el system.

systemAddUser	1	El sistema es capaz de agregar usuarios si es que no existen ya en el system.
systemLogin	1	El sistema es capaz de iniciar sesión de un usuario si es que ya se ingresó en el system.
systemLogout	1	El sistema es capaz de cerrar la sesión abierta.
systemTalkRec	1	El sistema es capaz de hablar con el chatbot.
systemSynthesis	1	El sistema es capaz de crear una síntesis.
systemSimulate	1	El sistema es capaz de simular una conversación con un límite de interacciones.

Tabla 1: *"Autoevaluación" Elaboración propia.*

Conclusiones del Trabajo

Alcances:

El sistema de chatbot implementado en Prolog permite la creación y gestión de múltiples chatbots, flujos de conversación y opciones de respuesta.

Los usuarios pueden interactuar con los chatbots a través de mensajes de texto, y el sistema responde según las opciones disponibles en cada flujo de conversación. También es posible simular una conversación con el chatbot.

Las funcionalidades básicas, como la creación de chatbots, flujos y opciones, así como la interacción con los usuarios, se implementaron de manera exitosa.

Limitaciones:

La implementación actual es una versión básica del sistema de chatbot y no incluye características avanzadas como el procesamiento de lenguaje natural (NLP) o la gestión de contextos complejos.

La interfaz de usuario es limitada y se realiza a través de consultas en el intérprete de Prolog, lo que puede ser poco amigable para usuarios no técnicos.

La falta de una interfaz gráfica de usuario (GUI) limita la accesibilidad y la usabilidad del sistema.

Dificultades y Desafíos:

El paradigma de programación lógico, aunque poderoso y elegante, puede ser menos intuitivo para aquellos familiarizados con enfoques más tradicionales como la programación orientada a objetos (POO).

La gestión de flujos de conversación y opciones requiere un diseño cuidadoso para garantizar la coherencia y la eficiencia.

Referencias

SWI-Prolog -- Library(Lists): list manipulation. (s. f.). https://www.swi-prolog.org/pldoc/man?section=lists

Flores, V (2021). *01 - INTRODUCCIÓN AL PARADIGMA LÓGICO.* Uvirtual. https://uvirtual.usach.cl/moodle/mod/hvp/view.php?id=156617.

Anexos

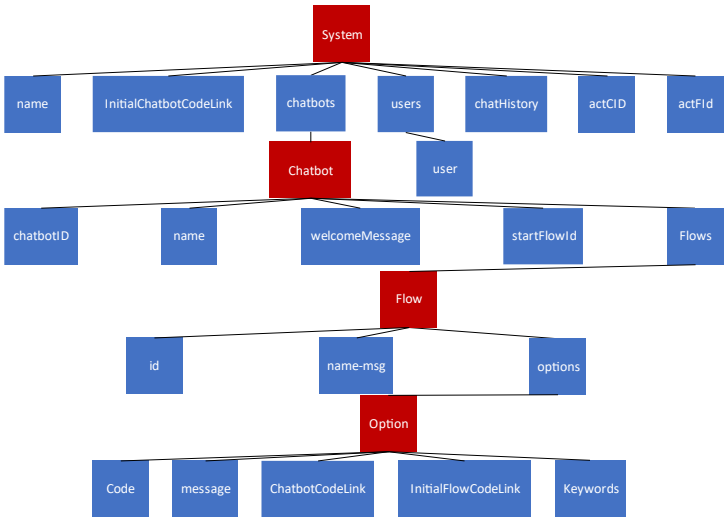

Figura 3: “Diagrama de estructura de system”. Elaboración propia.

Ejemplo de uso del sistema:

option(2, "2) Estudiar", 2, 1, ["estudiar", "aprender", "perfeccionarme"], OP2), option(1, "1) Viajar", 1, 1, ["viajar", "turistear", "conocer"], OP1), flow(1, "flujo1", [OP2], F10), flowAddOption(F10, OP1, F11), chatbot(0, "Inicial", "Bienvenido\n¿Qué te gustaría hacer?", 1, [F11], CB0), option(1, "1) New York, USA", 1, 2, ["USA", "Estados Unidos", "New York"], OP3), option(2, "2) París, Francia", 1, 1, ["Paris", "Eiffel"], OP4), option(3, "3) Torres del Paine, Chile", 1, 1, ["Chile", "Torres", "Paine", "Torres Paine", "Torres del Paine"], OP5), option(4, "4) Volver", 0, 1, ["Regresar", "Salir", "Volver"], OP6), option(1, "1) Central Park", 1, 2, ["Central", "Park", "Central Park"], OP7), option(2, "2) Museos", 1, 2, ["Museo"], OP8), option(3, "3) Ningún otro atractivo", 1, 3, ["Museo"], OP9), option(4, "4) Cambiar destino", 1, 1, ["Cambiar", "Volver", "Salir"], OP10), option(1, "1) Solo", 1, 3, ["Solo"], OP11), option(2, "2) En pareja", 1, 3, ["Pareja"], OP12), option(3, "3) En familia", 1, 3, ["Familia"], OP13), option(4, "4) Agregar más atractivos", 1, 2, ["Volver", "Atractivos"],

OP14), option(5, "5) En realidad quiero otro destino", 1, 1, ["Cambiar destino"], OP15), flow(1, "Flujo 1 Chatbot1\n¿Dónde te Gustaría ir?", [OP3, OP4, OP5, OP6], F20), flow(2, "Flujo 2 Chatbot1\n¿Qué atractivos te gustaría visitar?", [OP7, OP8, OP9, OP10], F21), flow(3, "Flujo 3 Chatbot1\n¿Vas solo o acompañado?", [OP11, OP12, OP13, OP14, OP15], F22), chatbot(1, "Agencia Viajes", "Bienvenido\n¿Dónde quieres viajar?", 1, [F20, F21, F22], CB1), option(1, "1) Carrera Técnica", 2, 1, ["Técnica"], OP16), option(2, "2) Postgrado", 2, 1, ["Doctorado", "Magister", "Postgrado"], OP17), option(3, "3) Volver", 0, 1, ["Volver", "Salir", "Regresar"], OP18), flow(1, "Flujo 1 Chatbot2\n¿Qué te gustaría estudiar?", [OP16, OP17, OP18], F30), chatbot(2, "Orientador Académico", "Bienvenido\n¿Qué te gustaría estudiar?", 1, [F30], CB2), system("Chatbots Paradigmas", 0, [CB0], S0), systemAddChatbot(S0, CB1, S01), systemAddChatbot(S01, CB2, S02), systemAddUser(S02, "user1", S2), systemAddUser(S2, "user2", S3), systemAddUser(S3, "user3", S5), systemLogin(S5, "user1", S7), systemLogout(S7, S9), systemLogin(S9, "user2", S10), systemTalkRec(S10, "hola", S11), systemTalkRec(S11, "1", S12), systemTalkRec(S12, "1", S13), systemTalkRec(S13, "Museo", S14), systemTalkRec(S14, "1", S15), systemTalkRec(S15, "3", S16), systemTalkRec(S16, "5", S17), systemSynthesis(S17, "user2", Str1), systemSimulate(S3, 5, 32131, S99), systemSynthesis(S99, "user", Str2), write(Str2).