Curso:

Lenguajes, Compiladores e intérpretes

Proyecto MrTrainer

Documentación

Profesor:

Marco Rivera Meneses

Estudiantes:

Adrián Gómez Garro

Gretchell Ochoa Quintero.

Marco Rivera Serrano

I semestre del 2021

MrTrainer

MrTrainer es el sistema experto creado para comportarse como un entrenador personal a partir de programacion logica.

Descripción de los hechos y reglas implementadas

- startConversation(List,Empty): Aqui se reconoce el saludo.
- startConversationAux(List,[]): Aqui empieza la recaudación de los datos del usuario para su rutina
- userSport(Sport): Se obtiene el deporte que el usuario quiere practicar
- userSportAux(Sport): Se obtiene el deporte si la respuesta no fue válida
- getSport(List,Empty,Sport): Aquí aparece la lista de deportes
- listSport(List,Empty,Sport): Se verifica la existencia del deporte
- userIllness(Illness): Se obtiene la patología del usuario
- getIllness(List,Empty,Illness): Aquí se obtiene la patología mostrando una lista de ser necesario.
- illnessList(Illness): Se muestran las patologías y se verifica su existencia
- userLevel(Level): Se le pregunta al usuario su frencuencia de ejercicio
- getRoutine(Number,Sport,Illness,Level,Days,Rutine): Aqui se muestra la rutina detallada al usuario si cumple con los parámetros.
- sport([(deporte)|S,S): si el deporte existe.
- illness([(patología)|S,S): si la patología existe.
- level([(nivel)|S,S): si la frecuencia existe.
- greeting([(saludo)|S,S): si el saludo existe.
- time([(tiempo)|S,S): si el tiempo existe.
- yes([(afirmación)|S,S): confirmación.
- no([(negación)|S,S): negación
- rutine(número de rutina, deporte, patología, nivel, días de ejercicio, rutina)

Descripción de las estructuras de datos desarrolladas

Para el proyecto en cuestión, como estructuras de datos, se trabajó únicamente con listas dada su accesible forma de manipular la información. Estas mismas fueron utilizadas en las reglas y hechos necesarios como sport, illness y similares. Su uso más notable fue en la rutina explicada anteriormente para un uso adecuado de sus elementos y fácil manipulación de estos.

Al utilizar estas estructuras con el fin mencionado se logró eficientar el código de mejor manera, ya que evitó la definición innecesaria de una cantidad de hechos abrumadora.

Descripción detallada de los algoritmos desarrollados

Se implementó el algoritmo de miembro enseñado en clase. Este mismo se utilizó para poder obtener el nivel del usuario y ajustar el plan de ejercicios si es principiante, intermedio o avanzado.

Este algoritmo utiliza recursividad llamando la función hasta encontrar el elemento si este existe dentro de la lista, de lo contrario, tira false.

Problemas sin solución

Si la aplicación está en uso y se presiona x, entra en un bucle sin poder salir de Prolog, por lo que se recomienda presionar Ctrl + C lo más rápido posible.

Tampoco, se pudo implementar algún comando de despedida como “Adios” o similares que pare por completo la aplicación.

Por último, no se pudo implementar el lenguaje español en su totalidad. Por ejemplo, no se pueden utilizar tildes, signos de interrogación o exclamación iniciales, diéresis entre otros caracteres especiales de la lengua española

Actividades realizadas por estudiante

** Estas fechas son aproximadas y fueron la primera idea que se tuvo en el desarrollo.*

Actividad	Descripción	Tiempo	Responsable
Interfaz	Desarrollo de la interfaz con la que el usuario interactúa.	9 días	Gretchell
Hechos	Base de Datos.	4 días	Adrian
Reglas	Funciones que trabajarán según el conocimiento que posea.	4 días	Adrian
Vocabulario	Crea el posible vocabulario que se puede usar.	3 días	Marco
BNF	Desarrollar un reconocedor de oraciones según el vocabulario.	4 días	Marco
Fecha de entrega	Revisar que todo se entregue	11/06	Gretchell, Marco, Adrian

Problemas solucionados

Uno de los principales problemas fue obtener la entrada del usuario y poder manipularla a nuestro favor. Para esto, utilizamos primeramente la función implementada en Prolog conocida como read(). Sin embargo, esta función nos limita demasiado si existen caracteres especiales dentro de la respuesta del usuario. Por lo que se decidió cambiar a read_string(). Esta otra, permite leer todo el input del usuario hasta cierto carácter, en nuestro caso el enter, representado como “\n”. Ahora, podemos utilizar todo el string.

Por otra parte, para obtener la frecuencia si el usuario escribe de más, se utilizó la función sub_atom(). Esta nos permite obtener los caracteres que queramos, en nuestro caso, el primero de la respuesta.

Por último, tenemos el caso general donde se prefería utilizar minúsculas para todas las respuestas, pero sin la necesidad de escribir cada una de ellas. Por lo que el equipo optó por utilizar la función string_lower(), esta toma un string y convierte todos sus caracteres a minúsculas sin importar el orden, permitiendo reducir bastante la cantidad de reglas necesarias.

Conclusiones

Prolog es un lenguaje muy potente para recrear diálogos o acciones humanas. Sin embargo, su introducción y práctica pueden resultar confusos para principiantes, dado que su principal enfoque es el lógico. Entonces se deben escribir los hechos varias veces para las posibles respuestas del Chatbot, esperando abarcar todas las que se puedan.

Además, no existen muchos tutoriales actuales para su uso, por lo que se recurre a la documentación propia de Prolog cayendo en la prueba y error si no se consigue lo deseado, perdiendo tiempo haciendo debugging por dichas razones.

Por último, crear una inteligencia artificial puede ser un muro bastante fuerte contra algunos estudiantes, dado que se necesita tener en cuenta cada posible escenario que se pueda llegar para evitar cualquier tipo de errores o sin algún sentido lógico.

Recomendaciones

Principalmente se recomienda mucha práctica abarcando los hechos, para que a la hora de utilizarlo profesionalmente, no caer en errores de falta de algunos casos o la completa ausencia de los mismos.

Además, se recomienda indagar bastante sobre el uso de inputs y manejo de los mismos. Sobre todo, con el manejo de strings para saber cuál función se necesita específicamente y no caer en ciclo de prueba y error hasta encontrar la adecuada.

Por último, en la administración del tiempo, se recomienda tener un calendario online o alguna aplicación parecida como Notion para mantener el equipo organizado y evitar atrasos innecesarios o que otro compañero tenga que realizar la tarea.

Bibliografía

Moraes, F. (14 de Mayo de 2021). *Prolog Programming: Give Your AI Bot It's Best Life. - Digital.com*. Obtenido de Prolog Programming: Give Your AI Bot It's Best Life. - Digital.com: https://digital.com/custom-software-development-companies/prolog-programming/

SWI Prolog. (s.f). *Predicate read_string/5*. Obtenido de Predicate read_string/5: https://www.swi-prolog.org/pldoc/man?predicate=read_string/5

SWI Prolog. (s.f). *Predicate string_lower/2*. Obtenido de Predicate string_lower/2: https://www.swi-prolog.org/pldoc/man?predicate=string_lower/2

SWI Prolog. (s.f). *Predicate sub_atom/5*. Obtenido de Predicate sub_atom/5: https://www.swi-prolog.org/pldoc/man?predicate=sub_atom/5

Wikipedia. (6 de Enereo de 2021). *ELIZA*. Obtenido de ELIZA: https://es.wikipedia.org/wiki/ELIZA

Bitácora

- Sábado 29 de mayo del 2021

Se realizó la división de tareas para ver que hacia cada persona y así disminuir la carga de trabajo.

- Domingo 30 de mayo del 2021

Se creó un nuevo repositorio en GitHub para subir todos los archivos de la tarea.

- Lunes 31 de mayo del 2021

Se realizó una búsqueda en internet de información para tener un mayor entendimiento de la que tiene que hacer en la tarea.

- Martes 1 de junio del 2021

Se realizó una búsqueda de ejemplos de rutinas y que ejercicios eran recomendados en cada disciplina.

- Miércoles 2 de junio del 2021

Se realizó una búsqueda de ejemplos de implementaciones del BNF para entender como es el funcionamiento del mismo.

- Jueves 3 de junio del 2021

Se realizó una búsqueda más específica de ejemplos de implementaciones del BNF en el ambiente de Prolog como guía para implementar el BNF.

- Sábado 5 de junio del 2021

Se logró hacer la implementación del barebones del BNF para agregarle el resto de características.

- Lunes 7 de junio del 2021

Se realizó una lista de rutinas para la organización de los datos y poder agregarlos a la base de datos del BNF.

- Martes 8 de junio del 2021

Se implementaron las rutinas en la base de datos del BNF.

- Miércoles 9 de junio del 2021

Se realizó la planeación de la interfaz en consola.

- Jueves 10 de junio del 2021

Se realizó la interfaz en consola y se empezó a trabajar en la integración de todo el código y en la verificación de que tenga un funcionamiento correcto.