Documentación Tarea 2 Programada 1.

Lenguajes compiladores e intérpretes.

Logico Funcional.

CallCenterLog_Prolog

Profesor:

Marco Rivera Meneses.

Integrantes:

Carlos Andrés Mata Calderón

David Robles

Victor Cruz Jiménez.

Grupo 6.

Descripción de algoritmos utilizados.

Para dar inicio a esta sección, primero se debe de explicar que es un BNF, entiéndase como BNF (Backus-Naur Form), es una técnica de notación meta sintáctica utilizada para definir la sintaxis de lenguajes de programación, conjuntos de comandos e instrucciones, formatos de documentos y protocolos de comunicación

En este caso, el BNF se planteó de modo que gracias a él se va a poder descomponer las oraciones en secciones, las cuales pueden ser analizadas para ejecutar una instrucción dependiendo de lo solicitado por el usuario, a su vez, gracias al BNF, es posible el plantear parámetros para el correcto funcionamiento de los algoritmos necesarios para la ejecución del programa.

Como primer algoritmo desarrollado, se encuentra

Algoritmo para validar la oración:

El algoritmo es capaz de detectar la entrada de texto del usuario y convertirla en una lista de palabras utilizando "input_to_list". Posteriormente, la lista de palabras es procesada por la regla "oración", la cual verifica si la oración cumple con la estructura definida en el BNF. Si la oración es correcta, el algoritmo devuelve true y procede con la siguiente regla. Sin embargo, si la oración es incorrecta, se le informa al usuario que la oración que escribió no es gramaticalmente correcta y se espera por la siguiente oración. Es importante destacar que este algoritmo se utiliza para procesar preguntas o consultas del usuario, pero no revisa el saludo ni el nombre.

Algoritmo para brindar la solución:

En este caso, el Sistema Experto tendrá que analizar la respuesta brindada por el usuario, en caso de que sea una respuesta afirmativa, el árbol de decisiones planteado avanza a la siguiente instrucción, en caso de que sea una respuesta negativa, toma una decisión y escribe la posible solución al problema específico.

Descripción de los hechos y reglas.

Para mayor facilidad a la hora de comprensión de los hechos y reglas, estas se van a dividir en secciones, una correspondiente al archivo main.pl y otra para db.pl

Hechos en main.pl.

- Validacion_gramatical. Recibe una lista de palabras dada por el usuario y verifica si esta corresponde a una oración gramaticalmente correcta utilizando una estructura previamente establecida. La función "Input_to_list" convierte la entrada de texto del usuario en una lista de palabras. La función "Input_to_string" convierte la entrada de texto del usuario en una cadena de texto.
- Sintagma Nominal. Recibe una lista de palabras y elimina el primer sintagma nominal encontrado, devolviendo el resto de las palabras. De manera similar, la función "sintagma_verbal" recibe una lista de palabras y elimina el primer sintagma verbal encontrado, devolviendo el resto de las palabras.

- Sintagma Saludo. verifica si una lista de palabras forma un saludo al programa. La función "inicio()" inicia el programa, responde al saludo del usuario, le pregunta su nombre, lo saluda nuevamente con su nombre y le pregunta acerca del problema.
- Eliminar Primeros. Elimina las partes innecesarias de una oración para reconocer el problema, devolviendo una lista de palabras que representan el problema en sí. La función "Obtener_causa" recibe una cadena de texto con el problema escrito por el usuario y devuelve las posibles causas, mientras que la función "Obtener_referencias" recibe una cadena de texto con el problema escrito por el usuario y devuelve enlaces de referencia.
- Causas y Referencias. Devuelven las posibles causas y las referencias, respectivamente, para un problema dado. La función "conversación" inicia una conversación con el usuario, utilizando todas estas funciones para proporcionar una experiencia de usuario completa y satisfactoria.

Hechos de db.pl

- Saludo. establece una lista de respuestas que el programa puede dar cuando se le saluda. Si el usuario utiliza alguna de las palabras establecidas como "hola", "tuanis", "pura vida", "eso todo bien", "buenos días", "buenas tardes" o "buenas noches", el programa responderá con la lista de palabras en minúsculas que siguen y almacenará el resto de la entrada del usuario en la variable S.
- Nombre Programa. Establece una lista de palabras que pueden usarse para referirse al programa, como "callCenterLog" o "servicio". Si un usuario utiliza alguna de estas palabras en una conversación, el programa responderá de acuerdo a la lista de palabras establecida y almacenará el resto de la entrada del usuario en la variable S.

- Determinante Masculino, Femenino y Neutro. establecen listas de determinantes para su uso en sintagmas nominales. Por ejemplo, si el usuario dice "la impresora", el programa reconocerá "la" como un determinante femenino y "impresora" como un sustantivo femenino, y almacenará el resto de la entrada del usuario en la variable S.
- Negacion. establece una lista de palabras que indican negación, como "no" o "nunca". Si el usuario utiliza alguna de estas palabras en una conversación, el programa lo reconocerá y tomará en cuenta esa negación en su respuesta, almacenando el resto de la entrada del usuario en la variable S.
- Sustantivo Masculino y Femenino. Establecen listas de sustantivos masculinos y femeninos, respectivamente, que se utilizan para referirse a los dispositivos y problemas específicos que el programa está diseñado para solucionar, almacenando el resto de la entrada del usuario en la variable S.
- "Verbo. establece una lista de verbos que pueden usarse en la conversación. El programa utilizará esta lista para determinar qué acción se necesita para resolver el problema identificado, almacenando el resto de la entrada del usuario en la variable S.
- Problema y Causa. establecen listas de problemas y sus causas asociadas. El programa utiliza esta información para identificar la causa del problema que el usuario está experimentando.
- Solucion. establece una lista de soluciones para las causas de los problemas identificados. El programa utiliza esta información para ofrecer soluciones específicas al problema identificado.
- "pregunta(X, Y)" toma dos argumentos: la pregunta y la respuesta negativa. Por ejemplo, la pregunta "¿Ya se aseguró de que la computadora está cargada?" tiene como respuesta

negativa "La computadora no está cargada". Cada pregunta se relaciona con un problema específico en un dispositivo determinado.

- Referencias. proporciona referencias de soluciones para cada problema en los diferentes dispositivos. Cada referencia tiene un enlace a un sitio web que proporciona información para solucionar el problema específico.
- Raiz. se encarga de realizar el diagnóstico a partir de la respuesta negativa proporcionada por el usuario. Esta función utiliza la información de las preguntas y referencias para determinar la raíz del problema y proporcionar una solución adecuada. En resumen, estos hechos y funciones conforman un sistema experto capaz de identificar y resolver problemas específicos en dispositivos tecnológicos.

Descripción de estructuras utilizadas.

El árbol de decisión en este sistema experto fue utilizado para modelar las diferentes soluciones posibles para los problemas presentados por el usuario. Cada problema se convierte en un nodo raíz del árbol, y los nodos hijos son las posibles causas que pueden estar provocando el problema.

Por ejemplo, si el problema es que la computadora no prende, el árbol de decisión tendrá un nodo raíz con este problema y nodos hijos que representan las posibles causas de que la computadora no prenda, como "la computadora no está cargada", "el botón de encendido no funciona" y "la batería está agotada o no funciona".

Cada nodo hoja en el árbol de decisión representa una solución específica para el problema que se está tratando. Estas soluciones son descritas en el código Prolog del programa como reglas, y cada regla representa una solución específica que se puede dar para el problema que se ha identificado.

Por lo tanto, el árbol de decisión se utiliza para determinar las posibles causas de los problemas que el usuario ha presentado, y luego se utiliza el código Prolog para proporcionar una solución para cada posible causa. Cuando se ejecuta el programa, el usuario puede responder preguntas específicas para determinar la causa raíz del problema, y luego se le proporciona una solución específica para ese problema.

Problemas conocidos con solucion.

Listas de verbos. A la hora de agregar verbos a la base de datos, en algunos casos no era suficiente con agregarlos de manera individual, es decir, verbos como “enciende” suelen venir acompañados de pronombres reflexivos que indican que el verbo recae sobre el mismo objeto. Por lo que en casos como “se enciende” es necesario un hecho específico en donde se incluye “se” seguido de “enciende” en una misma lista.

Problemas sin solución.

Respuestas complejas. Cuando el programa hace preguntas al usuario, es necesario recibir una afirmación o negación para continuar con el flujo de mensajes acorde a dicha respuesta del usuario, sin embargo, se encontró un problema en el caso de que el usuario de una respuesta compleja, en la que además de afirmar o negar la pregunta del programa, agregue información adicional o descriptiva. Por lo que en su estado actual, el aplicativo solo recibe respuestas simples tanto afirmativas como de negación.

Plan de actividades.

Estudiante a cargo:	Tarea asignada:	Descripción de tarea:	Tiempo estimado:	Fecha de entrega:
Carlos Andrés Mata Calderón	Lógica de prolog	Realizar de manera óptima la lógica del problema propuesto.	6 Horas	22/3/2023
	application_db	Escribir hechos y casos para el funcionamiento correcto de la base de datos	5 Horas	21/3/2023
David Robles	application_db	Escribir hechos y casos para el funcionamiento correcto de la base de datos	5 Horas	21/3/2023

	Logica de db	Realizar de manera óptima la lógica del problema propuesto.	7 Horas	22/3/2023
Victor Cruz Jiménez	main.pl	Encargado de plantear el archivo main, para el funcionamiento del programa, plantea casos y procedimientos necesarios para el correcto funcionamiento del mismo	7 Horas	22/3/2023
	Administración	Se encargará de la realización de la documentación necesaria para el correcto entendimiento de la tarea	4 Horas	24/3/2023

Conclusiones.

El sistema experto presentado es un ejemplo de cómo el paradigma de programación lógico, en particular Prolog, puede ser una excelente opción para desarrollar aplicaciones de procesamiento de lenguaje natural. En este caso, la aplicación se enfoca en brindar información a los usuarios sobre las causas y referencias relacionadas a un tema dado.

Prolog es un lenguaje de programación que se basa en la lógica y se utiliza comúnmente para implementar sistemas expertos y otros tipos de aplicaciones basadas en el conocimiento. En Prolog, se definen hechos y reglas que se utilizan para inferir nuevos conocimientos. Esto hace que sea una excelente opción para este tipo de aplicaciones, ya que permite la representación del conocimiento de una manera intuitiva y natural.

El sistema experto utiliza una variedad de reglas para validar la gramática de las oraciones y extraer información relevante de ellas. También utiliza una base de conocimiento para

proporcionar información específica sobre un tema en particular, incluyendo sus causas y referencias.

Además, Prolog tiene la ventaja de ser un lenguaje interpretado, lo que significa que los programas escritos en Prolog se ejecutan en tiempo real. Esto hace que el proceso de desarrollo y depuración sea más rápido y efectivo.

En conclusión, el paradigma de programación lógico, y Prolog en particular, resultan ser una excelente opción para el desarrollo de sistemas expertos de procesamiento de lenguaje natural. La facilidad para representar el conocimiento, validar la gramática y procesar la información de manera eficiente hace que Prolog sea una herramienta valiosa para el desarrollo de aplicaciones basadas en el conocimiento y la inteligencia artificial.

Recomendaciones.

A continuación, se presentan algunas recomendaciones para aquellos que deseen desarrollar sistemas expertos basados en Prolog para el procesamiento de lenguaje natural:

1. Familiarizarse con el paradigma lógico: es importante tener una comprensión sólida del paradigma de programación lógico antes de comenzar a desarrollar sistemas expertos en Prolog. La lógica y el razonamiento basado en reglas son fundamentales en este paradigma y son esenciales para el desarrollo de aplicaciones de procesamiento de lenguaje natural.
2. Diseñar una base de conocimiento adecuada: es importante tener una base de conocimiento clara y organizada para que el sistema experto pueda proporcionar información relevante y precisa a los usuarios. La base de conocimiento debe ser fácil de entender y actualizarse con regularidad para mantener la información actualizada.
3. Validar y preprocesar las entradas: antes de procesar cualquier entrada, es importante validarla y preprocesar para asegurarse de que esté en un formato adecuado. Esto puede incluir la validación gramatical y la eliminación de caracteres no deseados.

En general, el desarrollo de sistemas expertos basados en Prolog para el procesamiento de lenguaje natural puede ser una herramienta valiosa para proporcionar información precisa y relevante a los usuarios. Con una base de conocimiento adecuada, reglas claras y concisas, y una validación y depuración regular, es posible crear sistemas expertos altamente efectivos y precisos.

Referencias.

Wiki Uqbar. (s.f.). Errores frecuentes al programar en lógico. Recuperado el 26 de marzo de 2023, de http://wiki.uqbar.org/wiki/articles/errores-frecuentes-al-programar-en-logico.html

Wikipedia. (s.f.). Sistema experto. En Wikipedia, la enciclopedia libre. Recuperado el 26 de marzo de 2023, de https://es.wikipedia.org/wiki/Sistema_experto

UNIR. (s.f.). Sistema experto. Revista UNIR Ingeniería. Recuperado el 26 de marzo de 2023, de https://www.unir.net/ingenieria/revista/sistema-experto/

SWI-Prolog. (s.f.). SWI-Prolog Reference Manual. Recuperado el 26 de marzo de 2023, de https://www.swi-prolog.org/pldoc/doc_for?object=manual

La Web del Programador. (2007). Ejemplo de sistema experto. Recuperado el 26 de marzo de 2023, de https://www.lawebdelprogramador.com/foros/Prolog/629511-ejemplo-de-sistema-experto.html

Bitácora.

Carlos.

- **15/03/2023: Duración: De 7:00 a.m. a 9:00 a.m.** Creé el problema "la compu no prende" con el hecho "problema('la compu no prende')." y la función raíz correspondiente. También comencé a escribir la función "solucion(X, Y)." para proporcionar soluciones al problema.
- **18/03/2023: Duración: De 2:00 p.m. a 5:00 p.m.** Agregué una nueva causa al problema "la compu no prende" con el hecho "causa('la compu no prende', 'falla de alimentación eléctrica')." También completé la función "solucion(X, Y)." para el problema "la compu no prende".
- **22/03/2023: Duración: De 10:00 p.m. a 1:00 a.m.** Agregué un nuevo problema "no reproduce sonido" con el hecho "problema('no reproduce sonido')." También comencé a escribir la función raíz correspondiente y agregué el hecho "causa('no reproduce sonido', 'problema de hardware en la tarjeta de sonido')."
- **24/03/2023: Duración: De 6:00 a.m. a 10:00 a.m.** Agregué un nuevo problema "la impresora no imprime" con el hecho "problema('la impresora no imprime')." También agregué el hecho "causa('la impresora no imprime', 'atascamiento de papel')."

David.

- **16/03/2023: Duración: De 11 p.m. a 1:30 p.m.** Creé el problema “Internet no conecta” y añadí la función raíz asociada.

- **18/03/2023: Duración: De 7:30 p.m. a 11:00 p.m.** Agregué causas para el problema de internet, además de agregar otro problema relacionado “pagina no carga” y algunas de sus causas.
- **21/03/2023: Duración: De 8:00 p.m. a 11:00 p.m.** Creé los problemas relacionados al televisor “Televisor no enciende” y “televisor no tiene imagen”, con sus respectivas secciones, incluyendo la función raíz.
- **22/03/2023: Duración: De 10:30 p.m. a 12:00 a.m.** Agregué y cambie enlaces de referencia para los problemas relacionados al internet y a los televisores.
- **24/03/2023: Duración: De 6:30 p.m. a 8:30 p.m.** Revisé y trabajé en algunas secciones de la documentación.

Victor.

- **20/03/2023: Duración: De 3:00 p.m. a 7:30 p.m.** Se comenzó con la creación del archivo main.pl, se plantearon los primeros parámetros y se realizó un inicio simple para la ejecución del programa.
- **23/03/2023: Duración: De 3:00 p.m. a 8:00 p.m.** Se continuó con el desarrollo del archivo main.Se implementaron funciones para el correcto funcionamiento del sistema. }
- **24/03/2023: Duración: De 5:00 p.m. a 6:30p.m.** Se finalizó el archivo main..