**GuardIAn: Un agente inteligente encargado de la seguridad en las calles de Santa Fe.**

**Apellidos, Nombres (dejar esta parte en blanco en la versión de evaluación, pero sí escribirlos en la versión final)**

***Universidad Tecnológica Nacional, Facultad Regional Santa Fe***

**Abstract**

*En la cátedra IA se solicita a los alumnos que desarrollen un agente inteligente capaz de brindar seguridad a su usuario.*

*Se considera que el usuario se encuentra en una situación de riesgo y el agente puede escuchar a través de un micrófono y transformarlo a texto para interpretar la situación del usuario. Si el agente piensa que el usuario está en peligro, emprende acciones para rescatarlo. En este trabajo se desarrolló el agente usando sistemas de producciones para la toma de decisión. Para que este sistema sea más eficiente se implementó un algoritmo basado en la red RETE. El sistema de producciones corre en Java y el micrófono por el que el agente percibe es de un dispositivo con Android.*

**Palabras Clave**

Inteligencia artificial. Sistema de producciones. Red RETE. Seguridad. Agente inteligente.

**Introducción**

La presente contribución se desarrolló como un práctico de la cátedra Inteligencia Artificial de la Facultad Regional Santa Fe, en donde se solicitó el desarrollo de un agente que interprete las frases de un usuario o persona y descubra en la conversación patrones que estén asociados a situaciones de peligro tales como "ayuda", "me están pegando", etc. Para dar solución al problema planteado por la catedra, se implementó un agente inteligente [1] que transforma el sonido de la voz a texto y que, a través de un sistema de producciones, toma acciones que ayuden a la persona que está en problemas como, por ejemplo, llamar a la ambulancia, a la policía, etc.

1.  La reescritura cubre una amplia gama de métodos para reemplazar subtérminos de una fórmula por otros términos.

Este trabajo está relacionado con un trabajo previo realizado para la misma cátedra que permite encontrar un camino óptimo entre dos puntos de un mapa. Así la ambulancia, los bomberos o la policía utilizan estos algoritmos para encontrar el mejor camino a la ubicación del usuario.

En la implementación del sistema de producciones, para que el mismo sea eficiente, se desarrolló un algoritmo basado en el de red RETE de manera de mejorar los tiempos de respuesta [2, 3, 4]. Este algoritmo permite implementar el emparejamiento de patrones en un sistema de producción de manera eficiente.

Inicialmente solo se solicitó que el sistema fuera capaz de recibir las palabras como texto plano. Luego de la entrega del trabajo a la cátedra, se le adicionó la capacidad de captar audio del ambiente e interpretar palabras habladas utilizando el micrófono de un teléfono móvil y una aplicación de captura de voz.

Los sistemas de producciones cuentan con una base de reglas de comportamiento compuestas, cada una, por una condición y una acción asociada. Tales reglas se conocen como *producciones* y la base que las almacena es la *memoria de producciones* (MP)[5]. Sus aplicaciones varían desde sistemas de reescritura de cadenas1 hasta sistemas expertos. Su estado está dado por una base de hechos llamada *memoria de trabajo* (MT). La *máquina de inferencia* (MI) verifica si los hechos cumplen las condiciones de las reglas y luego, elige una y ejecuta su acción.

Una red RETE mejora la eficiencia de un sistema de producciones al recordar qué condiciones eran satisfechas en ciclos anteriores y actualizar esa información cuando cambian los hechos, sólo para las reglas afectadas. El algoritmo fue diseñado por Charles L. Forgy en 1983 [6].

**Elementos del Trabajo y metodología**

Para resolver el problema presentado por la cátedra, se siguieron los siguientes pasos:

* Se identificaron los tipos de hechos que se almacenan en la memoria de trabajo y las reglas que forman la memoria de producciones, usando la sintaxis de la lógica de primer orden.
* Se creó un listado de ‘palabras clave’ que al ser oídas advirtieran al sistema de un peligro potencial.
* Se programó un preprocesador de percepciones para filtrar las palabras claves del diálogo escuchado.
* Se programó el sistema de producciones.
* Se programó la interoperabilidad entre el sistema que es sujeto de este paper y la inteligencia artificial desarrollada en el trabajo previo de la cátedra.

El framework FAIA [7] se usó como herramienta para la implementación del agente.

A continuación, se detalla cómo se llevaron a cabo las distintas etapas.

**Representación de reglas y hechos**

Se abstrajo el problema de la siguiente forma:

* Se eligieron seis tipos de incidentes ante los cuales el agente debiera reaccionar: delito callejero, delito en un hogar o tienda, violencia doméstica, emergencia médica, incendio y explosión. Cuando el agente reconoce alguno de estos incidentes, toma las acciones apropiadas como, por ejemplo, llamar a la policía en caso de detectar un delito callejero.
* Hay palabras y frases que se suelen escuchar en contextos específicos como, por ejemplo, durante un robo. Se listaron, entonces, palabras y frases que pudieran relacionarse a los seis incidentes elegidos y se les llamó *palabras claves*. A cada palabra clave se le asoció un valor numérico que representa cuán fuertemente está ligada a un tipo de incidente.
* Se define un subconjunto de palabras clave que están muy fuertemente asociadas a un tipo particular de incidente. Se dice que estas palabras son *críticas* para identificar ese tipo de incidente, porque su percepción indica una gran probabilidad de que el incidente esté ocurriendo.
* Se reconoce con certeza que un incidente de un tipo particular está ocurriendo cuando se escucharon suficientes palabras clave asociadas a ese tipo de incidente y al menos una palabra crítica relacionada al mismo.
* Se considera que se escucharon suficientes palabras clave relacionadas a un incidente cuando la suma de los riesgos asociados a ellas supera un valor límite predefinido.

A partir de esta abstracción se identificaron reglas y clases de hechos. En la figura 1 se puede ver una de estas reglas.

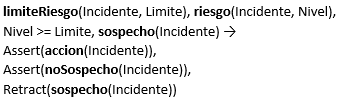


Figura 1: Regla que le indica al sistema que está ocurriendo un incidente y que debe actuar en respuesta.

**Palabras clave**

Las palabras clave y el tipo de incidente con el que se las puede relacionar se introducen al sistema de producciones como hechos, usando el formato de la figura 2. Éstas son las palabras que el sistema puede reconocer.

Sin embargo, el sistema no es provisto con una completa familia de palabras de cada palabra clave, dado que eso resultaría en una cantidad desmedida de hechos en la memoria de trabajo. Por ello se seleccionó una sola forma de cada palabra: de los verbos, se eligió su forma infinitiva y, de los sustantivos, su forma singular. Nos referiremos a ésta como la forma *normalizada* de las palabras.



Figura 2: Formato de los hechos introducidos al sistema de producciones. Este se lee como “La palabra clave ‘Palabra’ está relacionada al tipo de incidente ‘TipoIncidente’ con una fortaleza de ‘Valor’”.

**Preprocesador de entrada**

El sistema de producciones reconoce, entonces, palabras clave en su forma normalizada. Por lo tanto, no sería capaz de reconocer sinónimos de las palabras clave ni palabras no normalizadas, por lo que se vio la necesidad de acoplar un módulo de preprocesamiento. Éste toma todo lo escuchado, lo normaliza y busca los sinónimos de cada palabra. Por último, filtra las palabras que no son claves y le pasa las restantes palabras claves al sistema de producciones que agrega hechos con estas palabras a su memoria de trabajo.

**Simulador basado en sistemas de producción.**

El sistema de producciones constituyó el grueso del trabajo realizado. Tanto éste como la red RETE que el mismo utiliza son de desarrollo propio.

Sus partes son:

* Máquina de inferencia (MI)

En implementaciones sin RETE, este módulo se encarga de verificar si las condiciones de las reglas son satisfechas por los hechos de la memoria de trabajo. En la implementación que se hizo para la cátedra, este trabajo es delegado a la memoria de producción.

La MI debe, por otro lado, seleccionar una regla a ejecutar dentro de todas las reglas cuyas condiciones se cumplen. Esto se implementó mediante la aplicación sucesiva de diversos criterios de resolución de conflicto, los cuales se definieron en el agente. Estos fueron: *no duplicación*, *novedad*, *prioridad*, *especificidad* y *aleatorio*, en ese orden.

Por último, debe ejecutar la regla seleccionada. En la implementación que se hizo para el presente trabajo, el motor de inferencia devuelve una acción, la cual es ejecutada por el agente en el ambiente.

* Memoria de trabajo (MT)

Contiene los hechos. El estado del agente toma este rol. Se implementó mediante una conexión a una base de conocimientos en Prolog, que al principio contiene hechos de inicialización, como los que listan las palabras clave y sus niveles de relación con incidentes, y luego va agregando y eliminando hechos a medida que se ejecutan las acciones. Cada vez que cambian los hechos que contiene la memoria de trabajo se les avisa a los nodos predicados correspondientes de la red RETE de la memoria de producción y estos los propagan por la misma, activando y desactivando reglas.

* Memoria de producción (MP)

La memoria de producción se encarga del almacenamiento de reglas y es en este módulo donde la red RETE juega un papel principal.

**La red RETE**

La implementación del algoritmo RETE que se hace para este trabajo es una versión simplificada del diseño original. En esta, existen cinco tipos de nodo: *raíz*, *predicado*, *filtro*, *unión*, y *regla*.

El nodo *raíz* actúa simplemente como raíz de la red. No tiene otro propósito más que el de representar a la memoria de producción.

Los nodos *predicado* buscan de la memoria de trabajo los hechos que representan.

Los nodos *filtro* toman hechos de otros nodos y los filtran según lo requerido para satisfacer las condiciones. Por ejemplo, en la Figura 1 la condición solicita que el valor de *Nivel* sea mayor o igual al de *Limite* y ese requerimiento se corresponde con un filtro específico que se encarga de propagar solo aquellos hechos lo cumplan. Existen otros nodos *filtro* que actúan cuando una condición contiene dos predicados con la misma variable. En ese caso, el filtro propaga solo aquellos grupos de hechos que satisfacen los predicados de la condición y se refieren al mismo sujeto.

Los nodos *Unión* reciben hechos de otros nodos y les aplica un *producto cartesiano*, generando todas las combinaciones posibles de esos hechos, que corresponderán a distintas formas de satisfacer la condición.

Y los nodos *regla* son el final de la red. Los hechos que atravesaron los filtros y se sometieron a las uniones desembocan en estos nodos, los cuales almacenan grupos de hechos que, en conjunto, satisfacen la condición de una regla en su totalidad. Hay uno por cada regla. La máquina de inferencia solo necesita chequear los nodos *regla* para saber qué reglas pueden disparar y, por lo tanto, qué acciones se pueden aplicar. Representan la agenda de una red RETE tradicional.

La red RETE se inicializa inyectando los hechos presentes en la MT en los nodos *predicado*. Luego, se vincula los nodos *predicado* a la MT para que, al escuchar un cambio (adición o eliminación) en los hechos, puedan tomar y distribuir los mismos a través de la red.

La eficiencia del algoritmo RETE radica en que elimina la necesidad de hacer nuevos cálculos de emparejamiento si los hechos no se modifican entre ciclo y ciclo, y si se agrega o quita un hecho, el cambio se propagará desde el nodo *predicado* que le corresponde hasta los nodos *regla*, habilitando o deshabilitando acciones con un mínimo de cálculos. El tiempo de procesamiento es independiente de la cantidad de reglas.

Por supuesto, las redes RETE tienen sus desventajas. La cantidad de memoria que ocupan es muy grande debido a la replicación de datos en los nodos unión a lo largo de la red, es decir, los datos no solo se encuentran en la memoria de trabajo.

**Integración con el trabajo práctico anterior**

El trabajo que es sujeto de este paper fue el segundo trabajo práctico realizado para la cátedra. El primero involucraba crear una inteligencia artificial que guiara a un patrullero por una ciudad para llegar a la escena donde ocurría un incidente, teniendo que sortear varios obstáculos.

El segundo trabajo práctico solicitó que la acción para alguna de las reglas fuera comunicarse con ese patrullero para enviarlo a la zona del incidente. Durante tal acción debía también generar el ambiente para el patrullero. La posición del incidente dentro de la ciudad se configuró para ser aleatoria en cada ejecución, mientras que la ciudad y los obstáculos serían siempre los mismos.

**Trabajo subsiguiente**

Posterior a la entrega del trabajo práctico a la cátedra se le adicionaron funcionalidades nuevas al sistema.

En primer lugar, se programó una aplicación móvil en Android que es capaz de grabar sonido e interpretar lo que escucha usando un servicio externo para el reconocimiento del habla. La aplicación se conecta a un servidor que se ejecuta en una pc que toma las conexiones móviles y ejecuta una instancia de GuardIAn por cada una. Éstas se encargan de procesar las palabras que se le envían al servidor. Además, se creó una interfaz de usuario para facilitar el agregado o remoción de hechos iniciales de las memorias de trabajo de las nuevas instancias.

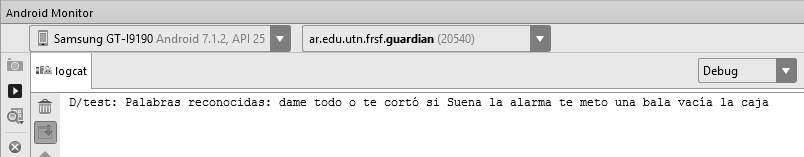


Figura 3: Captura de pantalla de la herramienta de debug de Android, mostrando el texto que la aplicación interpretó a partir de la captura de audio.

**Resultados**

Las pruebas del sistema demostraron que éste es capaz de elegir acciones correctas cuando se lo alimenta con conversaciones cuyo contenido indica claramente que un incidente está ocurriendo. La figura 3 muestra lo que la aplicación Android interpretó de lo que escuchó durante una prueba del agente, mientras que en la figura 4 se pueden ver las acciones que el agente eligió en consecuencia.

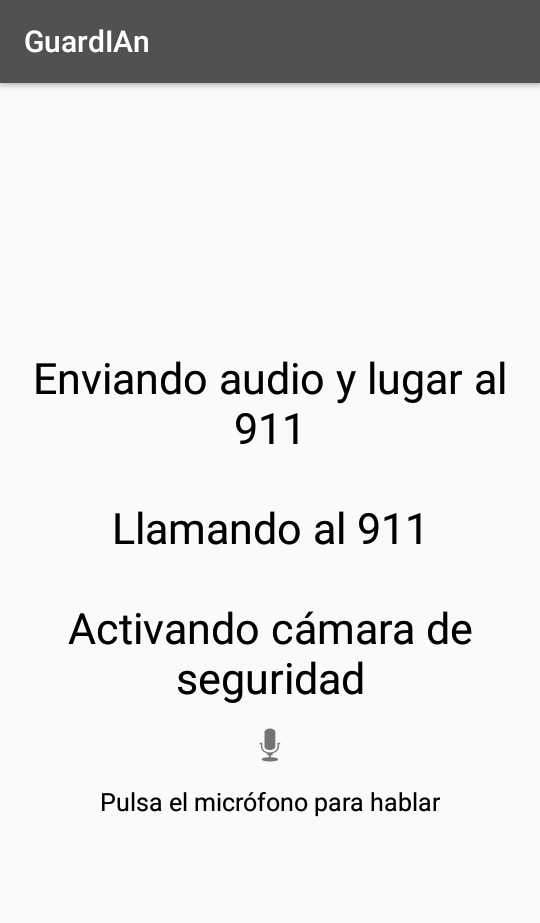


Figura 4: Captura de pantalla de la aplicación Android mostrando las acciones que el agente eligió.

También pudo comunicarse con la inteligencia artificial del patrullero, el primer trabajo práctico para la cátedra de Inteligencia Artificial.

La performance del sistema de producciones es aceptable, gracias a la utilización de la red RETE.

**Discusión**

Un obstáculo para la realización de este trabajo es que todos los servicios de speech-to-text de escucha continua son pagos. Las capacidades del agente se encuentran actualmente limitadas por la aplicación de captura de audio que utiliza, la cual sólo graba por períodos cortos de tiempo. A pesar de esto, el agente está listo para funcionar con aplicaciones de escucha continua.

En cuanto al trabajo pendiente, el módulo responsable de normalizar las palabras de la entrada puede ser mejorado. Durante la realización del trabajo, la atención se focalizó en el sistema de producciones, por lo que se relegó la eficiencia del proceso de normalización que puede ser ampliamente mejorada mediante una mejor utilización de herramientas de procesamiento del lenguaje natural.

**Conclusión**

El agente desarrollado cumple con las funciones solicitadas por la cátedra de IA. La red RETE permitió al agente ser eficiente y procesar un torrente continuo de palabras. La aplicación Android que se programó dotó al agente con funciones de captura de audio y speech-to-text.

En general, el trabajo fue una experiencia rica para sus participantes.

**Referencias**

1. Russell, S. & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Prentice-Hall, 3nd edition.
2. Diapositiva ‘Sistemas de producciones’ de la cátedra Inteligencia Artificial. UTN FRSF.
3. Sistemas de producción. https://www.infor.uva.es/~calonso/IAI/Tema9-Sistemas%20de%20Produccion/Rete.pdf (accedido: 28/08/2017)
4. Klahr, D., Langley, P. and Neches, R. (1987). *Production System Models of Learning and Development*. Cambridge, Mass.
5. Nilsson, N. J. (1998) Artificial Intelligence: A New Synthesis. Morgan Kaufmann ed.
6. Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem. Artificial Intelligence, 19, 17-37.
7. Roa, J., Gutiérrez, M. y Stegmayer, G. (2009) FAIA: Framework para la enseñanza de agentes en IA. IE Comunicaciones: Revista Iberoamericana de informática educativa. No 7/8, ISSN 1699-4574, Pp. 43-56.

**Datos de contacto**

*Gioria, Emiliano.*

*Institución: Universidad Tecnológica Nacional, Facultad Regional Santa Fe.*

*Cód. postal: 3000.*

*E-mail: emigioria@hotmail.com*