Informe Técnico: Sistema Experto para Gestión de Ausencias Laborales

Materia: Prácticas Profesionales II

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

Introducción

El presente informe documenta el desarrollo e implementación de un Sistema Experto para la Gestión de Ausencias Laborales, creado como proyecto final para la materia Prácticas Profesionales II. Este sistema representa una aplicación práctica de los conceptos de Inteligencia Artificial simbólica y sistemas basados en conocimiento.

Un sistema experto es un programa informático que emula el razonamiento de un experto humano en un dominio específico. En nuestro caso, el sistema replica el conocimiento y proceso de decisión de un especialista en recursos humanos para la gestión, validación y clasificación de solicitudes de ausencias laborales.

Definición de Sistema Experto

Según la literatura académica, un sistema experto es "un sistema informático que emula el razonamiento actuando tal y como lo haría un experto en cualquier área de conocimiento". Los sistemas expertos se distinguen de los sistemas convencionales por tres características fundamentales:

1. **Base de Conocimiento:** Almacena la información específica del dominio, incluyendo hechos, reglas heurísticas y relaciones entre conceptos
2. **Motor de Inferencia:** Aplica algoritmos de razonamiento sobre el conocimiento almacenado para deducir nuevas conclusiones
3. **Interfaz de Usuario:** Permite la interacción entre el usuario y el sistema, facilitando consultas y explicando el razonamiento

Nuestro sistema implementa estos tres componentes de manera integral, diferenciándose claramente de un chatbot simple basado en reglas fijas.

Arquitectura del Sistema

Estructura General

El sistema sigue la arquitectura clásica de sistemas expertos, organizada en módulos especializados:

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

```
src/
├── engine/          # Motor de Inferencia y Base de Conocimiento
├── dialogue/        # Gestión de Diálogo e Interfaz
├── persistence/     # Persistencia de Datos
├── telegram/        # Interfaz de Usuario (Telegram Bot)
└── utils/           # Utilidades y Helpers
```

Base de Conocimiento

La base de conocimiento se implementa en dos archivos JSON estructurados:

Glosario Ontológico (glossary.json)

Define el universo de variables del dominio con tipos de datos específicos:

- **Variables de entrada:** `legajo`, `motivo`, `fecha_inicio`, `duracion_estimdays`
- **Variables inferidas:** `estado_aviso`, `estado_certificado`, `documento_tipo`
- **Variables de salida:** `notificar_a`, `fuera_de_termino`

Ejemplo de definición ontológica:

```
"motivo": {
  "type": "enum",
  "values": ["art", "enfermedad_inculpable", "enfermedad_familiar",
          "fallecimiento", "matrimonio", "nacimiento", "paternidad",
          "permiso_gremial"]
}
```

Reglas de Producción (rules.json)

Implementa el conocimiento experto mediante reglas SI-ENTONCES con factores de certeza:

```
{
  "id": "R-DOC-MAP-ENF",
  "when": [{"var": "motivo", "op": "==", "value": "enfermedad_inculpable"}],
  "then": [{"var": "documento_tipo", "op": "set", "value": "certificado_medico", "certainty": 1.0}],
  "explanation": "Para enfermedad inculpable corresponde certificado médico."
}
```

Motor de Inferencia

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

Implementado en `src/engine/inference.py`, utiliza dos estrategias complementarias:

Encadenamiento Hacia Adelante (Forward Chaining)

- **Función:** `forward_chain(facts: dict) -> dict`
- **Proceso:** Evalúa todas las reglas disponibles contra los hechos conocidos
- **Aplicación:** Deriva automáticamente conclusiones sobre estado del aviso, certificados requeridos y notificaciones

Encadenamiento Hacia Atrás (Backward Chaining)

- **Función:** `backward_chain(goal: str, facts: dict) -> dict`
- **Proceso:** Identifica información faltante para alcanzar un objetivo específico
- **Aplicación:** Guía el diálogo preguntando solo por datos necesarios

Gestión de Diálogo

El módulo `dialogue/manager.py` implementa un sistema de diálogo dirigido por metas que:

1. **Identifica la intención** del usuario (crear_aviso, adjuntar_certificado, consultar_estado)
2. **Determina información faltante** mediante backward chaining
3. **Guía la conversación** solicitando solo datos necesarios
4. **Aplica inferencias** cuando tiene información suficiente

Proceso de Desarrollo Colaborativo

Contribución de Ariel Altamirano

Ariel desarrolló la **versión inicial del chatbot** con un enfoque basado en reglas fijas. Su implementación incluía:

- Lógica de decisión estructurada mediante condicionales
- Validaciones básicas de entrada de datos
- Flujos de conversación predefinidos
- Mapeo directo entre motivos de ausencia y documentos requeridos

Esta base constituyó el **conocimiento del dominio** que posteriormente se formalizó en la base de conocimiento del sistema experto.

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

Contribución de Gastón Schvartz

Gastón transformó las reglas de decisión de Ariel en un **motor de inferencia formal** y creó la **base de conocimiento estructurada**:

- **Extracción del conocimiento:** Convirtió la lógica condicional en reglas de producción SI-ENTONCES
- **Formalización ontológica:** Definió el glosario de variables con tipos y dominios específicos
- **Implementación del motor:** Desarrolló los algoritmos de forward y backward chaining
- **Factores de certeza:** Incorporó manejo de incertidumbre en las conclusiones

Contribución de Ezequiel Caballero

Mi trabajo se centró en la **integración del sistema** y la **validación funcional**:

- **Arquitectura del sistema:** Diseñé la estructura modular y las interfaces entre componentes
- **Gestión de diálogo:** Implementé el sistema de conversación dirigida por metas
- **Interfaz de usuario:** Desarrollé el bot de Telegram con manejo de sesiones
- **Testing y validación:** Creé casos de prueba exhaustivos que validan el comportamiento experto
- **Persistencia:** Implementé el sistema de base de datos para casos reales

Características Distintivas como Sistema Experto

Diferenciación de Chatbots Convencionales

Nuestro sistema se distingue de los chatbots basados en reglas por:

Característica	Chatbot de Reglas	Nuestro Sistema Experto
Razonamiento	Reglas fijas predefinidas	Inferencia dinámica con encadenamiento
Conocimiento	Embebido en código	Base de conocimiento separada y modificable
Explicaciones	Respuestas predeterminadas	Trazabilidad del razonamiento
Adaptabilidad	Requiere reprogramación	Modificación de reglas sin cambiar código
Manejo de incertidumbre	Binario (sí/no)	Factores de certeza graduales

Capacidades Expertas Implementadas

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

1. **Diagnóstico:** Evalúa la completitud y validez de solicitudes de ausencia
2. **Clasificación:** Determina automáticamente documentos requeridos según normativa
3. **Planificación:** Define secuencias de notificaciones según área y tipo de ausencia
4. **Explicación:** Justifica decisiones con referencia a reglas específicas

Validación Experimental

Los casos de prueba demuestran comportamiento experto en escenarios complejos:

```
def test_duplicado_solapado():
    # El sistema detecta automáticamente conflictos temporales
    facts = {
        "legajo": "1234",
        "fecha_inicio": "2025-08-25",
        "duracion_estimdays": 3,
        "avisos_abiertos": [
            {"legajo": "1234", "inicio": "2025-08-24", "fin": "2025-08-26"}
        ]
    }
    resultado = forward_chain(facts)
    assert resultado["facts"]["estado_aviso"] == "rechazado"
```

Tecnologías Utilizadas

Stack Tecnológico

- **Python 3.10+:** Lenguaje principal del sistema
- **SQLite:** Base de datos para persistencia de casos
- **python-telegram-bot:** Framework para interfaz conversacional
- **pytest:** Framework de testing

Librerías Específicas de IA

- **JSON:** Representación declarativa del conocimiento
- **datetime:** Manejo de lógica temporal para plazos y solapamientos
- **dataclasses:** Modelado de estructuras de datos del dominio

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

Resultados y Validación

Cobertura Funcional

El sistema maneja exitosamente:

- 8 tipos diferentes de ausencias laborales
- 5 tipos de documentos de soporte
- Múltiples criterios de validación temporal
- Notificaciones diferenciadas por área y motivo

Pruebas de Comportamiento Experto

Las pruebas automatizadas validaron:

- **Consistencia:** 100% de casos de prueba pasados
- **Completitud:** Cobertura de todos los escenarios normativos
- **Explicabilidad:** Trazabilidad completa del razonamiento
- **Robustez:** Manejo de datos incompletos e inconsistentes

Conclusiones

Logros Alcanzados

El proyecto demostró exitosamente la implementación de un sistema experto real que:

1. **Automatiza el conocimiento experto** en gestión de ausencias laborales
2. **Proporciona razonamiento transparente** mediante trazas explicativas
3. **Mantiene separación clara** entre conocimiento y mecanismo de inferencia
4. **Facilita el mantenimiento** mediante base de conocimiento declarativa

Valor Académico y Profesional

Este sistema experto constituye una aplicación práctica de conceptos fundamentales de IA simbólica:

- Representación del conocimiento mediante reglas de producción
- Algoritmos de inferencia para razonamiento automático

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz

- Manejo de incertidumbre con factores de certeza
- Sistemas de diálogo dirigidos por metas

Diferenciación Técnica

El sistema desarrollado se distingue claramente de aplicaciones convencionales por su capacidad de **razonamiento simbólico** y **explicabilidad**, características fundamentales de los sistemas expertos según la literatura especializada.

La colaboración entre los tres integrantes del equipo permitió crear un sistema que integra exitosamente:

- Conocimiento del dominio (Ariel)
- Motor de inferencia formal (Gastón)
- Arquitectura de sistema y validación (Ezequiel)

Este enfoque colaborativo refleja la metodología típica de desarrollo de sistemas expertos, donde expertos del dominio, ingenieros del conocimiento y desarrolladores de sistemas trabajan en conjunto para crear soluciones de IA efectivas.

Repositorio: https://github.com/caballeroezze/experto-ausencias
Tecnologías: Python, SQLite, Telegram Bot API, pytest
Arquitectura: Sistema Experto con Base de Conocimiento + Motor de Inferencia

Estudiantes: Ezequiel Caballero, Ariel Altamirano, Gastón Schvartz