Práctica 2: Razonamiento difuso hacia atrás

Sistemas Basados en Conocimiento 2024/25 (1C)

Fecha	Descripción de la versión
2024-10-02	Enunciado de la práctica 2

Objetivo

Implementar en Python un sistema basado en reglas capaz de realizar razonamiento hacia atrás (backward chaining), incorporando lógica difusa. El sistema debe utilizar una base de conocimiento, y ejecutarse desde la línea de comandos para permitir al usuario proporcionar consultas a resolver.

Requisitos

Se deberá crear una **aplicación de terminal**, usando **python**. El proyecto deberá estar gestionado con **git**. La aplicación deberá funcionar en un entorno linux equivalente al del laboratorio, por lo que se deberá usar python estándar y entornos virtuales. Se recomienda encarecidamente usar **uv**, siguiendo las indicaciones del campus virtual y que dará el profesor. En caso de no funcionar la práctica en la terminal de linux, se considerará suspensa.

Base de conocimiento

La base de conocimiento será un fichero de texto con reglas en formato:

```
cons :- ante1, ante2, ante3
```

con cualquier número de antecedentes. Las reglas podrán tener además grado de verdad, como:

```
cons :- ante1, ... [0.4]
```

Por simplificar, trataremos los hechos como reglas sin antecedentes:

```
hecho [0.8]
```

Cada regla será una línea. Las líneas que empiezen por #, o que estén vacías, se considerarán comentarios y se deberán ignorar.

Se proporciona una base de conocimiento de ejemplo, pero se deberá extender con más reglas, además de crear al menos otra base de conociento nueva para un dominio distinto.

Motor de inferencia

El motor de inferencia tendrá que ser capaz de recibir una consulta (un término como `pregunta?`) y determinar si se cumple o no, y con qué grado de verdad.

El algoritmo de razonamiento hacia atrás se recomienda implementarlo en una versión recursiva:

```
funcion backward_chain (consulta):
    para cada regla en la base de conocimiento:
        si el consecuente y la consulta encajan y
            se cumplen todos los antecedentes:
                return Sí
        si no:
            probar la siguiente regla
```

Además de traducir el algoritmo de pseudocódigo a python, y corregir las pequeñas cosas que faltan, habrá que añadir el uso de lógica difusa correctamente.

Interfaz

El programa deberá poderse ejecutar desde la línea de comandos, usando la librería `click`. Para leer la entrada del usuario, se recomienda usar la función `input` de python.

1. El programa debe permitir especificar por línea de comandos el fichero de base de conocimiento.
2. El programa debe permitir al usuario introducir una consulta, escribiendo el literal seguido de una interrogación (`desarrollador?`).

Funcionalidad optativa

1. Añadir al programa el comando "print" para poder examinar la base de conocimiento en tiempo real, lo que nos ayudará con el debugging.
2. Permitir al usuario añadir hechos en ejecución. Así se podrían ejecutar distintos casos con distintas premisas para consultar en cada caso concreto el resultado.
3. Permitir configurar el programa con un fichero de configuración (`.toml`) u opciones de línea de comando, por ejemplo para cambiar el idioma, los rangos de respuesta (mucho, poco, etc.) o la lógica difusa en uso (min/max vs. suma/producto).
4. Mostrar no sólo la respuesta sino también las reglas aplicadas, es decir, la derivación conseguida para la consulta. Esto hace nuestro sistema "explicable", además de ayudarnos con el debugging.

Ejemplo de ejecución

```
$ uv run practica2.py base_laboral.txt
> print
desarrollador :- programacion, analisis [1]
ing_frontend :- web, soft_skills [1]
...
psicologia :- profesor [0.7]
> soft_skills?
No
> add profesor [0.8]
> print
desarrollador :- programacion, analisis [1]
ing_frontend :- web, soft_skills [1]
...
psicologia :- profesor [0.7]
profesor [0.8]
> soft_skills?
Sí, mucho (0.7)
```

Entrega

Formato

1. La base de conocimiento creada se entregará como archivo `.txt` en el campus virtual. Si son varias, se entregarán juntas como fichero `.zip`.
2. El código se entregará a través de un repositorio git, que se usará para todas las prácticas del curso. La url del repositorio se subirá en el campus virtual. Se deberá crear una etiqueta `entrega_p2` en el repositorio, que será la versión del código que el profesor corregirá. El commit deberá estar subido al repositorio remoto a tiempo para la fecha límite de entrega.

Criterios de corrección

En la tarea de entrega del campus virtual se puede ver la rúbrica de corrección para la práctica, no sólo de las bases de conocimiento sino también del resto. Entre otras cosas, se valorará:

1. Limpieza del código. Se recomienda usar `black`.
2. Control de errores y buena arquitectura del programa.
3. Documentación y comentarios, explicativos pero sintéticos.

Fecha límite

Fecha de entrega: **11 de octubre de 2024**