Ingeniería del Conocimiento 2021-2022
Grado en Ingeniería Informática
Universidad de Granada

Sistema Basado en el Conocimiento

Autor:
Brian Sena Simons.

DNI: Y3683926F

Grupo:
3ºA Subgrupo A2

1. Introducción al sistema.

En esta práctica se desarrolla un sistema basado en el conocimiento en CLIPS dónde se pide implementar al menos dos funcionalidades. Estas son: aconsejar un alumno de informática sobre que rama de la carrera a elegir y dado dos asignaturas de tercero saber indicar cuál le va a gustar más.

El procedimiento seguido para el desarrollo de la base de conocimiento es el modelo de espiral de boehm donde vamos fabricando y testeando nuevas funcionalidades, sacando prototipos y realizando pruebas escaladas hasta la obtención del modelo final con el cumplimiendo de todos los requisitos.

Ambos subsistemas se diseñaron a partir de la recopilación de información del experto sobre ambos temas y con resolución de ejemplos determinado para ayudar en la codificación del conocimiento.

1.1. Aconsejar ramas

Para realizar la primera de ellas he optado por un esquema de "lista de compras" dónde para ello tenemos una instancia del usuario en nuestro sistema con todas las ramas aconsejadas y a continuación a medida que vamos haciendo preguntas vamos tachando aquellas ramas que discrepan con el resultado obtenido. Esa instancia usuario también dispone de estructura de datos para ir almacenando las justificaciones con cada proceso de eliminación de rama.

En el caso de que se intente eliminar una rama que no este no pasa nada, apenas se añade la justificación oportuna que se iba hacer. A continuación, el criterio de parada consisten en dos factores: el fin de la lista, este debe ser únicamente cuando haya 1 rama restante entre todos y nunca debe alcanzar el estado vacio, y cuando hemos realizado todas las preguntas. Es por ello que puede aconsejar más de una rama.

1.2. Aconsejar entre asignaturas.

Para realizar el segundo subsistema, tenemos que tener en cuenta que vamos a manejar mayor incertidumbre. Como para el desarrollo de este subsistema se vieron diversas técnicas para el manejo de este tipo de información he optado por utilizar las probabilidades bayesianas. La razón de esta elección recae sobre el hecho de que el experto suele aconsejar de la misma forma, con un cierto intervalo de confianza o probabilidad de que se de el caso.

Además este sistema nos permite dar un nivel de certeza al usuario en el otro lado de modo que le podría ayudar a inferir si realmente vale la pena seguir el consejo o seguir indagando. Para adaptar nuestro sistema simplemente se ha definido preguntas similares al otro subsistema.

En este subsistema la ejecución es totalmente secuencial, primero vienen las preguntas, luego las deducción simples a priori y a continuación las deducciones a posteriori y la búsqueda por la de mayor probabilidad. Además, las preguntas ayudan a inferir casos no preguntados, como el hecho de si es una persona estudiosa o no según lo que se haya respondido a preguntas anteriores, y también posee preguntas dependientes de las asignaturas elegidas.

El proceso es muy sencillo, una vez adaptado las redes bayesianas en la práctica anterior sabemos que funciones tocar. Primeramente forzamos que se ejecuten primero las preguntas. Luego con las respuestas obtenidas sabemos exactamente la cantidad de información que disponemos y que tipo/nivel de complejidad de inferencia podemos realizar. Es por ello que hay una jerárquia en las deducciones dónde inferencias de probabilidades conjuntas de menor complejidad cuando existen alguna de mayor complejidad son obviadas dado que las últimas abarcan un mayor espectro con mayor nivel de fiabilidad.

Una vez determinado hasta que grado de condicional o probabilidad conjunta podemos calcular, se realizan las respectivas inferencias, se busca el máximo y se le indica al usuario la razón del cambio de probabilidad y la decisión final.

2. Verificación y Validación.

Para el primer subsistema, una vez definido las preguntas se podría ponderar correctamente la influencia de cada una de ellas sobre el resultado. De este modo podía el experto predecir de ante mano los posibles resultados y verificar que el resultado obtenido correspondia con el esperado.

Para el proceso de validación se ha utilizado diversos usuarios ajenos a la producción del sistema para que realizen pruebas y dicten sus opiniones sobre la facilidad y aparencia general del sistema. Además, se ha intentado cometer fallos y producir la caída del sistema mediante entradas erróneas, fuera de rango, combinaciones sin sentido, y entre otras.

Con lo último se demonstró la robustez y accesibilidad del sistema, requisitos expuestos a priori sobre el sistema. Para el segundo subsistema se realizó el mismo esquema anteriormente definido. Sin embargo, la prueba por combinación y "fuerza-bruta" sería demasiado exhaustiva.

Es por ello que el proceso consistió en un escalado iterativo de complejidad, verificando las restricciones para el correcto funcionamiento de un sistema probabilístico como es la rango de valores de las probabilidades y observando las reglas que se disparaban y que calculos hacian. Una vez determinado que para una probabilidad condicional de tamaño 2 funcionaba, se pasaba a definir lo necesario para una de tamaño 3 y así.

Una vez terminado el sistema se realizó otra reunión con usuarios ajenos al sistema para obtener retro-alimentación. Los resultados fueron positivos, el tamaño del grupo usuario fueron de 10 usuarios ajenos al sistema, 8 sin conocimientos de informática.

3. Descripción del sistema

3.1. Aconsejar entre ramas

Clips 1: Estructura de datos utilizada

```
; Definimos una lista de ramas iniciales a recomendar
; Lo que haremos es ir quitando seg n responda a las preguntas.
; Una vez respondida las preguntas y/o solamente queda 1.
; Accionamos entonces la funcion decidir_y_explicar para el resultado
(defclass Ramas (is-a USER)
    (multislot lista
        (type STRING)
        (allowed-strings "IC" "SI" "IS" "TI" "CSI")
        (create-accessor read-write)
    )
    (multislot conclusion
        (type STRING)
        (create-accessor read-write)
    )
    (multislot justificacion
        (type STRING)
        (create-accessor read-write)
    )
)
```

Ahi vemos que tenemos varios “multislot” que sirve como un vector para que podamos ir insertando valores dentro de los modificables por adicción como lo son “conclusión” y “justificación” y los por suspresión como “lista”. Para ello tenemos que hacer uso de funciones de clips como “slot-insert$” y “slot-delete$”. A continuación se define una función diseñada para encontrar un valor en un vector y entonces borrar-lo:

Clips 2: Función de borrar de lista

```
(deffunction encontrar_index (?valor $?array)
    (bind ?respuesta nil)
    (bind ?i 1)
    (while (<= ?i (length$ ?array) )
    do
        (bind ?rama (nth$ ?i ?array) )
        (if (eq ?rama ?valor) then
            (bind ?respuesta ?i)
            (bind ?i (length$ ?array) )
        )
        (bind ?i (+ ?i 1) )
    )
    ?respuesta
)
;; Esta funci n utiliza la funcion auxiliar anterior para borrar
;; del array; Si este no existe no hace nada;
(deffunction borrar_valor (?user ?valor)
    (bind ?respuesta (send ?user get-lista))
    (bind ?i1 (encontrar_index ?valor ?respuesta) )
    (if (not (eq ?i1 nil) ) then
        (slot-delete$ ?user lista ?i1 ?i1  )
    )
    (bind ?respuesta (send ?user get-lista))
    ?respuesta
)
```

Estas funciones nos sirven para ir modificando nuestra supuesta “lista de compras” para ir reducciendo el número de ramas recomendadas por el sistema.

3.2. Aconsejar entre asignaturas

Clips 3: Estructura de datos

```
;; Definimos una instancia de usuario solamente para disponer de un
;; Vector de asignaturas para asegurarnos de que el usuario no intenta
;; Introducir una asignatura no disponible en nuestro sistema.
(defclass Datos (is-a USER)
    (multislot lista
        (type STRING)
        (create-accessor read-write)
    )
)
```

A partir de una lista de asignturas podemos manejar y controlar la correcta elección de una pareja de asignturas por el usuario evitando valroes erróneos. A continuación necesitamos definir el conocimiento y probabilidades conocidas. Una vez definida estas probabilidades tendremos valores con condicionales de mayor tamaño que los ejemplos definidos por el código del profesorado, es por ello que se definen funciones de complejidad creciente.

Clips 4: Búsqueda de máxima probabilidad

```
(defrule getMax
    (declare (salience -100))
    (red causal efectos)
    ?f <- (max_posteriori ?X ?p)
    (prob_posteriori ?Y ?q)
    (not (maxE ?Y ?a)) (test (>= ?q ?p) )
    =>
    (retract ?f)
    (assert (maxE ?Y ?q) )
    (assert (max_posteriori ?Y ?q) ) )
```

Clips 5: Probabilidad conjunta para condicional triple

```
(defrule probconj4
(deducciones simples)
(probcond3 ?X SI ?c1 ?v1 ?c2 ?v2 ?c3 ?v3 ?pc)
(prob ?X SI ?)
(prob ?c1 ?v1 ?p1)
(prob ?c2 ?v2 ?p2)
(prob ?c3 ?v3 ?p3)
(not (and
        (probcond4 ?X SI ?ca ?va ?cb ?vb ?cc ?vc ?cd ?vd ?pr)
        (valor ?ca ?va)
        (valor ?cb ?vb)
        (valor ?cc ?vc)
        (valor ?cd ?vd)
     )
)
(not (sumado ?X SI ?c1 ?v1 ?c2 ?v2 ?c3 ?v3) )
(todo)
(not (valor ?c1 Desconocido))
(not (valor ?c2 Desconocido))
(not (valor ?c3 Desconocido))
=>
(bind ?p (* (* (* ?pc ?p1) ?p2) ?p3) )
(assert (probconj4 ?X SI ?c1 ?v1 ?c2 ?v2 ?c3 ?v3 ?p))
(assert (sumar probconj2 ?X SI ?c1 ?v1 ?p))
(assert (sumar probconj2 ?X SI ?c2 ?v2 ?p))
(assert (sumar probconj2 ?X SI ?c3 ?v3 ?p))
(assert (sumar prob ?X SI ?p))
(assert (sumado ?X SI ?c1 ?v1 ?c2 ?v2 ?c3 ?v3) ) )
```

Clips 6: Inferencia para 3 causas

```
(defrule inferencia3causas
(red causal causas)
(calculado)
(influye ?c1 ?X)
(influye ?c2 ?X)
(influye ?c3 ?X)
(test (neq ?c1 ?c2))
(test (neq ?c2 ?c3))
(test (neq ?c1 ?c3))
(valor ?c1 ?v1) (valor ?c2 ?v2) (valor ?c3 ?v3)
(not (test (eq ?v1 Desconocido) ) )
(not (test (eq ?v2 Desconocido) ) )
(not (test (eq ?v3 Desconocido) ) )
(probcond3 ?X SI ?c1 ?v1 ?c2 ?v2 ?c3 ?v3 ?p+x/c1c2c3)
(prob ?c1 ?v1 ?p1) (prob ?c2 ?v2 ?p2) (prob ?c3 ?v3 ?p3)
(not (and
        (probcond3 ?X SI ?ca ?va ?cb ?vb ?cc ?vc ?cd ?vd ?pr)
        (valor ?ca ?va) (valor ?cb ?vb)
        (valor ?cc ?vc) (valor ?cd ?vd)
     )
)
(not (inferido ?X) )
(prob ?X SI ?)
=>
(assert (prob_posteriori_causas ?X  ?p+x/c1c2c3))
(assert (prob_conjunta ?X (* ?p3 (* ?p2 (* ?p1 ?p+x/c1c2c3)))))
(assert (prob_conjunta_negativo ?X (* ?p3 (* ?p2 (* ?p1 (- 1 ?p+x/c1c2c3))
   ))))
(assert (inferido ?X) ) )
```

4. Manual de uso del sistema

El usuario una vez cargado el sistema con:

Clips 7: Carga del sistema

```
CLIPS>(load SBC.clp)
CLIPS>(reset)
CLIPS>(run)
;; Elegir una opcion del menu de control del modulo A
```

Lo demás es auto-descriptivo y además posee un flujo de control de entradas evitando equivocaciones.