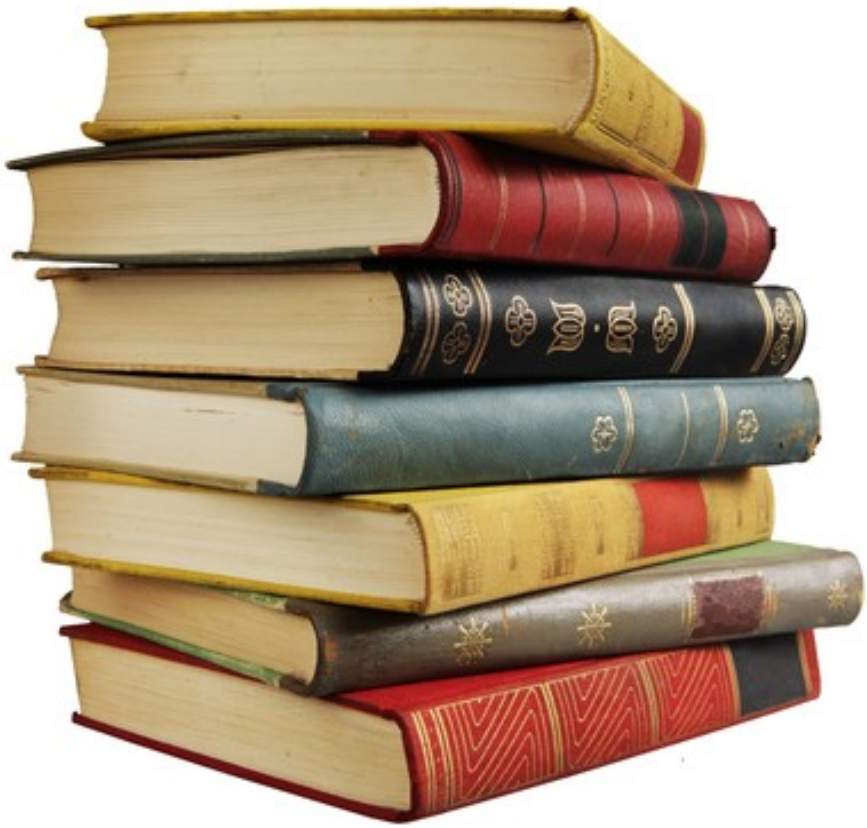
GoodReads Expert System

Corso di Ingegneria della Conoscenza e Sistemi Esperti

Mattia Menna mat. 601515



INTRODUZIONE

GoodReads Expert System (d'ora in avanti chiamato GRES) è un sistema esperto progettato in clips e embeddato in Java con lo scopo di sostituire un esperto libraio per la scelta di un romanzo da leggere. L'utente target di GRES è una persona di qualsiasi età che potrebbe essersi appena affacciata al mondo della lettura, oppure un “veterano” in cerca di nuovi autori.

Il sistema inizia con una piccola classificazione dell'utente, in modo da avere una idea generale sui suoi gusti e magari anche informazioni per il filtraggio di contenuti che potrebbero essere considerati inappropriati per alcuni utenti.

Dopo questa fase inizia una serie di domande di carattere generale, verranno proposti una serie di aggettivi che l'utente vorrebbe il suo romanzo avesse e caratteristiche quali lunghezza e genere.

Alla fine del processo all'utente verranno proposti alcuni titoli e un menù che darà la possibilità all'utente di ritrattare qualche risposta oppure di riprovare.

La parte del sistema scritta in Java offre la possibilità all'utende di memorizzare alcune sue scelte quali autore e genere preferito e la data di nascita che servirà al sistema per il filtraggio di contenuti inappropriati nel caso di utenti con età inferiore ad una certa soglia.

DESCRIZIONE DEL SISTEMA IN DETTAGLIO

CLASSIFICAZIONE UTENTE (No Login)

Il sistema utilizza solamente tre regole per la classificazione utente, la prima serve a classificare un utente in base all'età, cioè a generare alcuni fatti che impediranno al sistema di proporre testi con contenuti inappropriati in caso di età inferiore ad una certa soglia:

(defrule età

(presentazione)

(not (login))

=>

(bind ?eta (DomandaB "Età del lettore? (Inserire un valore numerico)"))

(if (< ?eta 16) then (assert (esplicito no)))

(if (< ?eta 12) then (assert (violento no)))

(assert (età))

)

La seconda serve per raccogliere informazioni riguardo il grado di conoscenza dell'utente nell'ambito del sistema, ciò viene realizzato tramite la conoscenza di quante ore in media l'utente passa a leggere per settimana:

```
(defrule lettore-accanito
(presentazione)
(not (login))
=>
(printout t "Quante ore dedichi alla lettura settimanalmente?" crlf)
(printout t "1)0" crlf)
(printout t "2)Tra 1 e 10" crlf)
(printout t "3)Più di 10" crlf)
(bind ?accanito (Domanda "" 1 2 3))
(switch ?accanito
        (case 1 then
                (assert (lungo no))
                (assert (autore no)))

        (case 2 then
                (assert (lungo indifferente))
                (assert (autore no)))

        (case 3 then
                (assert (lungo indifferente))
                (assert (autore si)))
))
```

La terza viene attivata solo per utenti che nell'ambito del sistema sono definiti “lettori accaniti”, questi sono coloro che hanno risposto con l'opzione numero tre alla domanda precedente. Questa regola propone all'utente una serie di autori, la scelta di uno di questi porterà il sistema ad inferire alcune caratteristiche per la ricerca dei romanzi da consigliare:

(defrule domanda-autori

(declare (salience 10000))

?x <- (autore si)

(età)

(not (login))

(not (esplicito ?))

(not (violento ?))

=>

(printout t "Secondo te, quale di questi autori rappresenta meglio un romanzo che potrebbe piacerti?" crlf)

(printout t "1) J.K. Rowling" crlf)

(printout t "2) J.R.R. Tolkien" crlf)

(printout t "3) Charles Dickens" crlf)

(printout t "4) Sir Arthur Conan Doyle" crlf)

(printout t "5) Isaac Asimov" crlf)

(printout t "6) George Orwell" crlf)

(printout t "7) Ken Follet" crlf)

(printout t "8) Stephen King" crlf)

(printout t "9) Nessuno di questi" crlf)

(printout t "10)Aiuto" crlf)

(bind ?scelta (Domanda " " 1 2 3 4 5 6 7 8 9 10))

(switch ?scelta

(case 1 then

(assert (imprevedibile si))

(assert (violento no))

(assert (leggero si))

(assert (serie si)))

(case 2 then

(assert (lungo si))

(assert (divertente no))

(assert (esplicito no))

(assert (leggero si)))

(case 3 then

(assert (lungo si))

(assert (divertente no))

(assert (esplicito si))

(assert (leggero si)))

(case 4 then

(assert (divertente si))

(assert (lungo no))

(assert (imprevedibile si))

(assert (leggero si)))

(case 5 then

(assert (lungo no))

(assert (serie si))

(assert (imprevedibile si))

(assert (leggero si)))

(case 6 then

(assert (lungo no))

(assert (divertente no))

(assert (leggero no))

```
        (assert (serie no)))
    (case 7 then
        (assert (lungo si))
        (assert (esplicito si))
        (assert (violento si))
        (assert (serie si)))
    (case 8 then
        (assert (violento si))
        (assert (esplicito si))
        (assert (leggero si))
        (assert (serie si)))
    (case 10 then
        (assert (aiuto autore))
        (retract ?x))
))
```

CLASSIFICAZIONE UTENTE (LOGIN)

Nel caso in cui l'applicazione venga lanciata tramite il file Jar, e quindi con l'implementazione della funzione di login, la classificazione utente avviene al momento della registrazione ed è più o meno simile alla classificazione effettuata tramite domande in clips che abbiamo descritto prima:

Ovviamente il sistema continuerà nell'esecuzione anche nel caso in cui si scelga di avviarlo dal Jar senza però usufruire dell'opzione di login, in questo caso l'esecuzione sarà perfettamente uguale all'esecuzione del file clp da console clips.

DOMANDE GENERALI

Dopo la fase di classificazione, abbiamo alcune domande generali che serviranno a raccogliere informazioni sui gusti dell'utente, queste domande varieranno in base alla precedente fase di classificazione e alle scelte che l'utente eseguirà volta per volta.

Propongo due di queste regole come esempio:

```
(defrule serie
(autore ?)
(not (serie ?))
=>
(printout t "Cerchi un romanzo che faccia parte di una serie? (Trilogie, Cicli...)" crlf)
(printout t "1)Sì" crlf)
(printout t "2)No" crlf)
(printout t "3)Non so" crlf)
(bind ?scelta (Domanda " " 1 2 3))
(switch ?scelta
        (case 1 then
                (assert (serie si)))
        (case 2 then
                (assert (serie no)))
        (case 3 then
                (assert (serie indifferente))))


)
```

```
(defrule leggero
(autore ?)
(not (leggero ?))
=>
(printout t "Secondo te un buon romanzo è:" crlf)
(printout t "1)Impegnativo" crlf)
(printout t "2)Leggero" crlf)
(printout t "3)Non so" crlf)
(printout t "4)Aiuto" crlf)
(bind ?scelta (Domanda " " 1 2 3 4))
(switch ?scelta
        (case 1 then
                (assert (leggero no)))
        (case 2 then
                (assert (leggero si)))
        (case 3 then
                (assert (leggero indifferente)))
        (case 4 then
                (assert (aiuto leggero))))
)
```

INDIVIDUAZIONE GENERE

In base ad alcune delle risposte precedenti (non in modo sequenziale) vengono attivate alcune regole che il sistema utilizza per cercare di indovinare il genere letterario che più potrebbe soddisfare le richieste dell'utente. I generi proposti sono:

Storico, Sociale, Avventuroso, Psicologico, Fantastico, Giallo, Gotico, Rosa, Fantascienza.

Propongo due di queste regole come esempio:

```
(defrule storico
(intenso no|indifferente)
(leggero no|indifferente)
(not (ritratta genere))
(not (genere storico))
=>
(printout t crlf "Stai per caso cercando un romanzo storico? (Ricorda che puoi scegliere più di un genere)" crlf
        "1)Si" crlf
        "2)No" crlf
        "3)Non so" crlf
        "4)Aiuto" crlf)
(bind ?scelta (Domanda " " 1 2 3 4))
(switch ?scelta
    (case 1 then
        (assert (genere storico)))
    (case 4 then
        (assert (aiuto storico))))
)
```

(defrule fantastico

(leggero si|indifferente)

(imprevedibile si|indifferente)

(intenso si|indifferente)

(not (ritratta genere))

(not (genere fantastico))

=>

(printout t crlf "Stai per caso cercando un romanzo fantastico? (Ricorda che puoi scegliere più di un genere)" crlf

"1)Si" crlf

"2)No" crlf

"3)Non so" crlf

"4)Aiuto" crlf)

(bind ?scelta (Domanda " " 1 2 3 4))

(switch ?scelta

(case 1 then

(assert (genere fantastico)))

(case 4 then

(assert (aiuto fantastico))))

)

SCELTA DEI ROMANZI

Una volta raccolte abbastanza informazioni, alcune regole verranno attivate e i libri da consigliare verranno asseriti. Il template che definisce un libro è il seguente:

```
(deftemplate libro
    (slot nome (type STRING))
    (slot punteggio (type STRING))
    (slot autore (type STRING))
    (slot serie
        (type STRING)
        (default "-")))
```

Riporto due regole che asseriscono i libri da consigliare come esempio:

```
(defrule oliver-twist
(genere sociale|psicologico|avventura|nil)
(imprevedibile si|indifferente)
(violento no|si)
(leggero si|indifferente)
(serie no|indifferente)
(esplicito no|si)
(lungo no|indifferente)
(divertente no|indifferente)
(intenso si|indifferente)
(ebook si|no|nil)
=>
(assert (libro (nome "Oliver Twist") (punteggio "3.83") (autore "Charles Dickens")))
(assert (trovato)))
```

(defrule racconto-di-due-città

(genere storico|sociale|nil)

(imprevedibile no|indifferente)

(violento si)

(leggero no|indifferente)

(serie no|indifferente)

(esplicito si)

(lungo si|indifferente)

(divertente no|indifferente)

(intenso si|indifferente)

(ebook si|nil)

=>

(assert (libro (nome "Racconto di Due Città") (punteggio "3.78") (autore "Charles Dickens")))

(assert (trovato))

)

REGOLE PER IL SUCCESSO

Se il sistema è riuscito a trovare almeno un libro, allora l'esecuzione è considerata un successo e viene attivata la regola per il successo che insieme alla regola print (che stampa i libri trovati) informa l'utente sull'esito dell'esecuzione:

(defrule successo

(declare (salience -1000))

(trovato)

(not (interfaccia))

=>

(printout t crlf "I seguenti libri probabilmente fanno per te: " crlf)

)

(defrule print

(declare (salience -1001))

(trovato)

(libro (nome ?nome) (punteggio ?punteggio) (autore ?autore) (serie ?serie))

(not (interfaccia))

=>

(printout t "- TITOLO: " ?nome " RATING: " ?punteggio "/5" " AUTORE: " ? autore " SERIE: " ?serie crlf)

)

REGOLA PER IL FALLIMENTO

Se il sistema non è riuscito a trovare nessun libro confacente le caratteristiche ricercate, allora l'esecuzione è considerata un fallimento e viene attivata la regola di fallimento che informa l'utente riguardo l'insuccesso dell'esecuzione:

```
(defrule fallimento
(declare (salience -10000))
(not (trovato))
(not (ritratta))
(not (interfaccia))
=>
(printout t crlf "Mi dispiace, non sono riuscito a trovare nessun libro" crlf)
(printout t "con le caratteristiche richieste, ora puoi procedere nei seguenti modi:" crlf crlf)
(printout t "1)Provare a cercare un altro libro" crlf)
(printout t "2)Modificare qualche risposta" crlf)
(printout t "3)Uscire" crlf)
(bind ?scelta (Domanda " " 1 2 3))
(switch ?scelta
        (case 1 then
                (reset)
                (assert (presentazione))
                (run))
        (case 2 then
                (assert (ritratta))))
)
```

REGOLA PER RITRATTARE

Che l'esecuzione sia un successo o un fallimento, viene comunque presentata all'utente la possibilità di cambiare le risposte date ed effettuare un'altra ricerca senza dover ripetere l'intero processo di esecuzione:

```
(defrule ritratta
(declare (salience -1))
?rit <- (ritratta)
?imp <- (imprevedibile ?attr1)
?viol <- (violento ?attr2)
?legg <- (leggero ?attr3)
?ser <- (serie ?attr4)
?espl <- (esplicito ?attr5)
?lun <- (lungo ?attr6)
?div <- (divertente ?attr7)
?int <- (intenso ?attr8)
?gut <- (ebook  ?attr9)
?gen <- (genere ?attr10)
=>

(printout t "Si sono scelte le seguenti caratteristiche per il libro da trovare: " crlf)
(switch ?attr1
        (case si then
                (printout t "1)Imprevedibile" crlf))
        (case no then
                (printout t "1)Tranquillo" crlf))
        (case indifferente then
                (printout t "1)Imprevedibile/Tranquillo" crlf)))

(if (eq ?attr2 si) then (printout t "2)Anche con contenuto violento" crlf) else (printout t "2)Senza contenuto violento" crlf))
```

```
(switch ?attr3
        (case si then
                (printout t "3)Leggero" crlf))
        (case no then
                (printout t "3)Impegnativo" crlf))
        (case indifferente then
                (printout t "3)Leggero/Impegnativo" crlf)))

(switch ?attr4
        (case si then
                (printout t "4)Facente parte di una serie" crlf))
        (case no then
                (printout t "4)Autoconclusivo" crlf))
        (case indifferente then
                (printout t "4)Anche facente parte di una serie" crlf)))

(if (eq ?attr5 si) then (printout t "5)Anche con contenuto esplicito" crlf) else (printout t "5)Senza contenuto esplicito" crlf))

(switch ?attr6
        (case si then
                (printout t "6)Voluminoso" crlf))
        (case no then
                (printout t "6)Non molto lungo" crlf))
        (case indifferente then
                (printout t "6)Lungo/Breve" crlf)))
(switch ?attr7
        (case si then
                (printout t "7)Divertente" crlf))
        (case no then
```

```
            (printout t "7)Serio" crlf))
        (case indifferente then
            (printout t "7)Divertente/Serio" crlf)))
(switch ?attr8
        (case si then
            (printout t "8)Intenso" crlf))
        (case no then
            (printout t "8)Riflessivo" crlf))
        (case indifferente then
            (printout t "8)Intenso/Riflessivo" crlf)))
(switch ?attr9
        (case si then
            (printout t "9)Facente parte del progetto Gutemberg" crlf))
        (case  no  then
            (printout t "9)Anche facente parte del progetto Gutemberg" crlf))
        (case  nil  then
            (printout t "9)Anche facente parte del progetto Gutemberg" crlf)))

(if (eq ?attr10 nil) then (printout t "10)Nessun genere in particolare" crlf)
        else (printout t "10)Generi:" crlf)
            (do-for-all-facts ((?p genere)) TRUE (printout t "           " (str-cat (nth$ 1 (fact-slot-value
?p implied))) crlf)))

(printout t "11)Riprova" crlf)

(printout t "Scegliere la caratteristica da ritrattare: ")
(bind ?scelta (Domanda " " 1 2 3 4 5 6 7 8 9 10 11))
(switch ?scelta
        (case 1 then (retract ?imp) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
        (case 2 then (retract ?viol) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
        (case 3 then (retract ?legg) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
```

```
(case 4 then (retract ?ser) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 5 then (retract ?espl) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 6 then (retract ?lun) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 7 then (retract ?div) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 8 then (retract ?int) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 9 then (retract ?gut) (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 10 then (do-for-all-facts ((?g genere)) TRUE (retract ?g))
        (retract (assert (ritratta genere)))
        (do-for-all-facts ((?p libro)) TRUE (retract ?p)))
(case 11 then (retract ?rit)
        ))
)
```

REGOLE DI AIUTO

Infine abbiamo un insieme di regole di aiuto che servono a dare assistenza all'utente durante l'intero processo di esecuzione spieganto il perché viene posta e come rispondere ad una certa domanda:

Riporto due regole di questo tipo come esempio:

```
(defrule aiuto-leggero
?x <- (aiuto leggero)
=>
(printout t crlf "Un romanzo leggero è più facile da leggere, il linguaggio è più semplice e la trama " crlf "non troppo complicata, un romanzo impegnativo al contrario necessita una discreta quantità " crlf  "di concentrazione per essere letto e apprezzato.Premere INVIO per continuare" crlf)
(get-char)
(retract ?x)
(printout t "Secondo te un buon romanzo è:" crlf)
(printout t "1)Impegnativo" crlf)
(printout t "2)Leggero" crlf)
(printout t "3)Entrambi" crlf)
(printout t "4)Aiuto" crlf)
(bind ?scelta (Domanda " " 1 2 3 4))
(switch ?scelta
        (case 1 then (assert (leggero no)))
        (case 2 then (assert (leggero si)))
        (case 3 then (assert (leggero indifferente)))
        (case 4 then (assert (aiuto leggero))))
)
```

```
(defrule aiuto-storico
?x <- (aiuto storico)
=>
(printout t crlf "Il romanzo storico è un'opera narrativa ambientata in un'epoca passata, della quale ricostruisce le atmosfere, gli usi, i costumi, la mentalità e la vita in genere, così da farli rivivere al lettore. Secondo l'Enciclopedia Britannica, un romanzo si definisce storico quando 'è ambientato in un'epoca storica e intende trasmetterne lo spirito, i comportamenti e le condizioni sociali attraverso dettagli realistici e con un'aderenza (in molti casi solo apparente) ai fatti documentati.' Premere INVIO per continuare." crlf)
(get-char)
(retract ?x)
(printout t crlf "Stai per caso cercando un romanzo storico?" crlf
        "1)Si" crlf
        "2)No" crlf
        "3)Non so" crlf
        "4)Aiuto" crlf)
(bind ?scelta (Domanda " " 1 2 3 4))
(switch ?scelta
    (case 1 then
        (assert (genere storico)))
    (case 4 then
        (assert (aiuto storico))))
)
```