Prolog: Gigel

- Responsabili:
  - Teodor Szente [mailto:teodor98sz@gmail.com]
  - Andrei Olaru [mailto:cs@andreiolaru.ro]
  - George Muraru [mailto:george murarugeorgec@gmail.com]
- Deadline soft: **20.05.2020**
- Deadline hard: 21.05.2020
- Data publicării: 07.05.2020
- Data ultimei modificări: 12.05.2020 changelog
- Data tester-ului: 12.05.2020
- Tema se va încărca pe **vmechecker [https://vmchecker.cs.pub.ro/ui/#PP]**
- Forum temă [https://acs.curs.pub.ro/2019/mod/forum/view.php?f=2676]

Introducere

Tema are ca scop implementarea unui simplu agent conversațional bazat pe sistemul ELIZA [https://en.wikipedia.org/wiki/ELIZA]. Dezvoltat în anii ‘60 s-a bucurat de multă popularitate fiind primul candidat cu șanse reale de a trece testul Turing la doar 14 ani după ce acesta a fost formulat.
Deși acum are doar relevanța istorică, reprezintă un exercițiu util pentru a ne familiariza cu conceptele din Prolog, în care a fost implementat de multe ori de-a lungul anilor. Vom numi robotul *Gigel*.

Exemplu de conversație:

```
Tu:salut
Gigel:salutare .
Tu:ce faci ?
Gigel:bine tu ?
Tu:ma uit la un film
Gigel:ce film ?
Tu:ma uit la Trumbo
Gigel:nu am auzit de trumbo , care este filmul tau favorit ?
Tu:filmul meu favorit este Papillon
Gigel:papillon ? nu am auzit de el , despre ce e vorba ?
```

Pentru a ușura parsarea caracterelor Gigel **nu va folosi diacritice** sau **cratime**.

Observați că pentru ușurință toate semnele de punctuație, atât la replică, cât și la răspuns, sunt **despărțite de spatiu** de cuvintele de lângă ele.

Vom reprezenta replicile ca liste de cuvinte și semne de punctuație. Vom numi cuvintele și semnele de punctuație *tokens*.

În temă, bazat pe o serie de **reguli simple**, va trebui ca pentru fiecare replică a utilizatorului să produceți un răspuns; acest răspuns va fi bazat pe *replica utilizatorului*, pe *emoția* detectată în conversație, și (pentru bonus) pe *subiectul* detectat în conversație.

Veți lucra **numai în fișierul `gigel.pl`**. În fișierul `chat.pl` se găsesc o serie de funcții ajutătoare.

Reprezentarea cunoștințelor

Alegerea unei replici se bazează pe o structură pe **două niveluri**. Inițial, bazat pe cuvintele găsite în replica utilizatorului (le vom numi **cuvinte cheie**), se selectează un set de reguli, iar apoi bazat pe forma replicii utilizatorului și eventual pe alte criterii, se selectează / formează un răspuns.

Baza de cunoștințe a lui Gigel este formată din predicate `rules`, fiecare reprezentând un set de reguli pentru anumite cuvinte cheie:

```
rules(cuvinte_cheie, lista_de_reguli).
```

Cuvintele cheie sunt date sub forma unei liste de cuvinte. Atunci când mesajul dat de utilizator conține **în ordine** toate cuvintele cheie se va alege o regulă din lista de reguli, fiecare regulă fiind reprezentată ca:

```
rule(expresie, [replică_1, replică_2, …], acțiuni, emoții, tags),
```

unde:

- **expresie**: un șablon cu care mesajul dat de utilizator trebuie să unifice.
- **[replică_1, replică_2, …]**: o lista de replici din care se va alege un răspuns.
- **acțiuni**: o lista de acțiuni care vor fi tratate după ce robotul a răspuns; pentru scopul acestei teme singura acțiune tratată va fi cea de `exit`, care indică încheierea conversației.
- **Emoții**: vezi Emoții.
- **Tags**: vezi Tags (pentru bonus).

De exemplu dacă vrem ca robotul sa răspundă la `salut` cu `salut.` vom defini regula:

```
rules([salut], [
        rule([_], [
                [salut, .]
        ], [], [], [])
]).
```

Adică, dacă replica utilizatorului conține cuvântul *salut*, indiferent de ce formă are replica, se va răspunde cu *salut*.

Dacă ne dorim ca Gigel să răspundă în funcție de context putem defini:

```
rules([destept], [
        rule([esti, destept], [
                [ai, dreptate, .]
        ], [], [], []),
        rule([nu, esti, destept], [
                [poate, fi, si, asta, adevarat, .]
        ], [], [], []),
]).
```

Sau să se folosească de cuvintele din mesaj:

```
rules([am, fost], [
        rule([am, fost, la, X], [
                [a, fost, frumos, la, X, '?']
        ], [], [], [])
]).
```

Dacă vrem ca un set de reguli să se aplice pentru mai multe cuvinte cheie:

```
rules(X, [
        rule(_, [
                ['hello]
        ], [], [], [])
]) :- member(X, [hello, hey]).
```

Setul de reguli se găsește în fișierul `chat.pl`.

Pentru selecția replicii, se va implementa predicatul `select_answer`, care decide răspunsul și acțiunile de realizat, în funcție de replica utilizatorului și de conținutul memoriei.

Memoria este reprezentată ca un dicționar care reține numărul utilizărilor pentru fiecare replică, iar fiecare replică este reprezentată prin reuniunea cuvintelor din replică. Mai multe despre memorie citiți în secțiunea Memorie.

Acțiuni

Dacă dorim ca Gigel să închidă conversația după o replică:

```
rules([pa], [
        rule([_], [
                [pa],
                [la, revedere],
        ], [exit], [], [])
]).
```

Apoi, în predicatul `handle_actions(+Actions)` vom completa codul astfel încât dacă `exit` se afla printre acțiuni programul va eșua și se va opri.

Emoții

Ne dorim ca Gigel să reacționeze la starea interlocutorului. Pentru această temă ne dorim să "detectăm" 3 emoții: fericit, trist, neutru. Predicatele `happy/sad` conțin termeni asociați fiecărei emoții:

```
?- sad(X).
X = rau ;
X = trist ;
X = regret ;
X = suferinta ;
X = sumbru ;
X = intristare ;
...
?- happy(X).
X = fericit ;
X = bucuros ;
X = vesel ;
X = voios ;
X = incantat ;
X = recunoscator ;
...
```

Gigel va calcula care din aceste cuvinte au fost folosite mai des și va folosi replici doar pentru care emoția găsită se află în lista de emoții din regulă.
Dacă ambele tipuri de cuvinte sunt folosite de un număr egal de ori atunci emoția va fi `neutru`.
Dacă o regula are lista de emoții goală atunci ea poate fi folosită pentru orice tip de emoție, dar o regulă care are o listă de emoții specificată trebuie să aibă prioritate față de o regulă care nu are nimic în lista de emoții, astfel încât să aibă prioritate un comportament mai specific.

Tags (Bonus)

Pentru o conversație mai bună extindeți funcționalitatea de la emoții pentru tag-uri. Acestea reprezintă subiectul conversației. Spre deosebire de emoții, tag-urile pot fi oricâte și sunt definite folosind predicatul `tag` care leagă numele tag-ului de lista de cuvinte legate de tag-ul respectiv:

```
?- tag(Tag, Words).
Tag = sport,
Words = [tenis, volei, fotbal, fotbalist, box, gimnastica, judo, oina, schi...] ;
Tag = film,
Words = [film, camera, actor, scena, cameraman, scenariu, colorizare, regizor].
```

Similar cu comportamentul de la emoții, o listă care are tag-uri specificate trebuie să aibă prioritate față de o regulă „generală", care nu are taguri specificate, astfel încât să aibă prioritate un comportament mai specific.

Cerințe

Alegerea replicii (70p)

Pentru a răspunde la o replică a utilizatorului, Gigel va folosi următorul algoritm:

- va căuta în baza de cunoștințe setul de reguli pentru care toate cuvintele cheie se regăsesc în replica utilizatorului, în ordine.
- în setul de reguli găsit, va găsi prima regulă care descrie corect forma replicii utilizatorului,
- dintre răspunsurile posibile, dacă sunt mai multe, se va alege acel răspuns care a fost utilizat de mai puține ori de către Gigel în conversație, până acum. În acest scop se va folosi o memorie a replicilor de până acum ale lui Gigel.

Alegerea replicii pe baza detecției emoției (30p)

Gigel va selecta dintr-un set de reguli doar regulile care se potrivesc pentru emoția detectată a utilizatorului, dacă o astfel de indicație există în elementul de *emoții* din regulă.

Emoția utilizatorului va fi detectată numărând aparițiile cuvintelor caracteristice fiecărei emoții (vezi Emoții).

Alegerea replicii pe baza subiectului conversației (20p)

Gigel va selecta dintr-un set de reguli doar regulile care se potrivesc cu subiectul conversației, dacă o astfel de indicație există în elementul *tags* al regulii.

Subiectul va fi detectat pe baza aparițiilor cuvintelor caracteristice fiecărui subiect (vezi Tags (bonus)).

Mențiuni

Se vor scădea 10 puncte pentru warning-uri (Singleton variables).

Există foarte multe implementări de ELIZA pe internet pe care le puteți consulta, dar implementarea finală trebuie să vă aparțină.

Detalii de implementare

Lucrați numai în fișierul `gigel.pl`.

Lucrul cu dicționare

Puteți apela manual get și put pe memorie sau alte dicționare ajutătoare dar va trebui să tratați cazurile în care cheia nu există, altfel programul va eșua:

```
Val = Dict.get(Key). % might fail if Key not in Memory
NewMemory = Dict.put(Key, NewVal).
```

Pentru a obține istoricul replicilor puteți accesa cheile memoriei:

```
?- Memory = memory{'joc tenis': 3, 'ma uit la box': 2, 'ma uit la un film': 4},
dict_keys(Memory, Keys).
Memory = memory{'joc tenis':3, 'ma uit la box':2, 'ma uit la un film':4},
Keys = ['joc tenis', 'ma uit la box', 'ma uit la un film'].
```

Funcții ajutătoare

Deoarece cheia unui dicționar nu poate fi o lista de cuvinte, aceasta este convertită mereu în string înainte să fie folosită drept cheie. Pentru a găsi numărul de apariții a cuvintelor va fi nevoie să convertiți replicile din stringuri în lista de atomi sau invers:

```
?-  words('ma uit la un film', Tokens).
Tokens = [ma, uit, la, un, film].

?-  unwords([ma, uit, la, un, film], String).
String = 'ma uit la un film'.
```

Tot pentru lucrul cu memoria sunt definite predicatele ajutătoare `add_answer` și `get_answer`, ca și `min_element` și `max_element` (vedeți documentația lor în fișierul `chat.pl`).

Documentare

Pentru ușurința documentării, variabilele din schelet sunt denumite sub forma `_Variabilă`. Dacă numele începe cu underscore, nu vor apărea warnings legate de variabile *singleton* pentru variabila respectivă (ceea ce s-ar întâmpla în cazul scheletului).

Vă recomandăm să folosiți în rezolvare numai nume de variabile care **nu** încep cu underscore.

Mod interactiv

Puteți conversa cu Gigel efectuând în prolog interogarea `gigel`.

Arhiva de pornire

- arhiva de pornire

Changelog

- 14.05.2020: clarificare comentarii schelet
- 14.05.2020: clarificare privind prioritatea regulilor în contextul unei emoții sau al unui subiect de conversație.
- 12.05.2020: corecție arhitectură de testare pentru necesitatea calculului soluției de către temă.
- 12.05.2020: corecție tester pentru o mai bună testare a emoțiilor și tag-urilor.
- 07.05.2020: publicare tema.