# CZ3005: Artificial Intelligent
# Patient with a Sympathetic Doctor

Zhang Xinye U1622118K

February 28, 2018

## Introduction

In this lab we need to develop a Dialogue AI with Prolog. The Prolog system is regarded as a sympathetic doctor, conversing with a patient who can answer only yes or no. The doctor is able to diagnose the patient's condition while asking question sensitively depending upon patient's pain level and mood level.

## Overview

Under above requirements, I have developed a set of 5 illnesses, 9 symptoms, 5 pain levels, as well as 5 mood levels as the knowledge base. The doctor core has the following interfaces being called by GUI or CLI: nextQuestion/1, answer/2, and diagnos/1. The doctor firstly ask for patients pain level and mood level. Then, the system will decide on proper gestures to make response with. Afterwards, the system will ask patients whether or not they have each of 9 predefined symptoms.

As for GUI of the Dialogue AI, I have implemented a simple HTTP server in Prolog, which made use of build-in http server library of SWI-Prolog. Comparing to those C/Java Prolog Lnterface libraries, this ensures the system 100% cross-platform.

## Structure

The whole project is carried out with the following structure.

- doctor_core.pl: the core knowledge and logic, including implementation of nextQuestion/1, answer/2, and diagnose/1, sets of pains, moods, and illnesses.
- human_print.pl: humanize strings for output.
- server.pl: web server program, front-end
- util.pl: utility functions used by different part, including list_empty/2

# Implementation and explanation

## General Idea of Doctor Core Implementation

To solve problem, we need to break down to some major problems.

1. ***Storing user's previous choice.*** Similar to the implementation in TalkingBox Demo, every time user give an answer to a question, assert(answered) will be called. Furthermore, is the answer to the question is positive, assert(have_symptom) will be called to.

2. ***Reading user's previous choice.*** As most of the logic is based on set operation, we need to collect the previous separated asserts to a list. So, I have used findall/3 for answer collection. Instead of findnsols/3 as in the Demo, findall/3 is more generic.

3. ***Determine the phase of the program.*** There are four phase, namely, asking pain, asking mood, asking symptoms, and giving diagnose result. Three flags *pain, mood*, and *ready_to_diagnos* is set upon the completion of the first three phases. To determine whether to set the flag, we firstly collect previous result into *History*, then subtract *History* from each phase's library list, namely, *pain_library, mood_library*, and *symptom_library*, if the result is empty corresponding flag is set.

## doctor_core.pl

```
:-['utl.pl'].
pain.  % store selected pain, only one from library will be selected
mood.  % store selected mood, only one from library will be selected
ready_to_diagnos. % flag setted when all questions are asked and answered
have_symptom(nothing). % symptoms with positive answer, including symptoms, pain,
    and mood
answered(nothing).      % answered items, including symptoms, pain, and mood

%% Set of pains and moods
pain_library([unbearable_pain, lot_of_pain, manageable_pain, mild_pain, no_pain]).
mood_library([calm, angry, weepy, stressed, sad]).
%% Set of symptoms
symptom_library([temperature, sweat, ache, sneeze, cough, blood, chill, rash,
    headache]).
%% Set of gestures
gesture(polite_gesture, [look_concerned,mellow_voice,light_touch,faint_smile]).
gesture(calming_gesture, [greet, look_composed, look_attentive]).
gesture(normal_gesture, [broad_smile, joke, beaming_voice]).
%% Set of illness and its list of symptoms
illness(fever, [temperature, sweat, ache, weepy, headache]).
illness(cold, [sneeze, cough, temperature, chills, mild_pain]).
illness(cancer, [mild_pain, temperature, sweat, sneeze]).
illness(injury, [blood, lot_of_pain, weepy, angry, sweat]).
illness(food_poisoning, [temperature, rash, stressed, ache, sneeze]).
```

```
23
24 %% Select a random item from selected set of gesture
25 gesture(G):-
26     (
27         %% condition 1: havn't choose pain/mood, or no_pain or calm is selected
28         %% Set GL to normal_gesture
29         (not(current_predicate(pain/1)); not(current_predicate(mood/1));
30             pain(no_pain); mood(calm)),
31          gesture(normal_gesture, GL);
32         %% condition 2: unbearable_pain, lot_of_pain or angry is selected
33         %% Set GL to polite_gesture
34         (pain(unbearable_pain); pain(lot_of_pain); mood(angry)),
35         gesture(polite_gesture, GL);
36         %% condition 3: manageable_pain, mild_pain, weep or stressed is selected
37         %% Set GL to calming_gesture
38         (pain(manageable_pain); pain(mild_pain); mood(weepy); mood(stressed)),
39         gesture(calming_gesture, GL)
40     ),
41     random_member(G, GL). % get a random item from GL
42
43 %% determin thether all items form a library L have been answered. can be applied
       to pain_library, mood_library or symptom_library
44 list_finished(L, ValidChoices, If_finished):-
45     findall(X, answered(X), History),
46     list_to_set(L, P),
47     list_to_set(History, S),
48     subtract(P, S, ValidChoices),
49     list_empty(ValidChoices, If_finished).
50
51 %% Following 3 rules are interface nextQuestion/1, query nextQuestion(Next) in
       prolog will return the next item to be asked.
52 %% Cut operator has been used.
53
54 %% ask symptom
55 nextQuestion(Next):-
56     % pain and mood has finished, should ask Symptom
57     pain_library(Pain_library),
58     mood_library(Mood_library),
59     symptom_library(Symptom_library),
60     list_finished(Pain_library, _, If_pain_finished), % determine Pain_library
           has all been answered
61     list_finished(Mood_library, _, If_mood_finished), % determine Mood_library
           has all been answered
62     list_finished(Symptom_library, ValidChoices, _),
63     (
64         (current_predicate(pain/1);If_pain_finished), % if one of pain is selected
               or Pain_library is answered through
```

```
65          (current_predicate(mood/1);If_mood_finished) % if one of mood is selected
                or Mood_library is answered through
66      ),!,
67      random_member(Next, ValidChoices).
68
69  %% ask mood
70  nextQuestion(Next):-
71      % pain has finished, should ask mood
72      pain_library(Pain_library),
73      mood_library(Mood_library),
74      list_finished(Pain_library, _, If_pain_finished), % determine Pain_library
            has all been answered
75      list_finished(Mood_library, ValidChoices, _),
76      (
77          current_predicate(pain/1); If_pain_finished % if one of pain is selected
                or Pain_library is answered through
78      ),!,
79      random_member(Next, ValidChoices).
80
81  %% ask pain
82  nextQuestion(Next):-
83      % pain have not been selected
84      pain_library(Pain_library),
85      list_finished(Pain_library, ValidChoices, _),!,
86      random_member(Next, ValidChoices).
87
88
89  %% helper function for positive answer of answer/2, depending of answering a
        pain, mood or symptom, make different action
90  answer_h(Q):-
91      pain_library(Pain_library),
92      mood_library(Mood_library),
93      symptom_library(Symptom_library),
94      (
95          member(Q, Pain_library) -> assert(pain(Q)); % if Q is a pain
96          member(Q, Mood_library) -> assert(mood(Q)); % if Q is a mood
97          member(Q, Symptom_library)->true           % otherwise Q is a symptom
98      ),
99      assert(have_symptom(Q)).                       % add Q to have_symptom
100
101 % Interface for answering question, eg. query answer(headache, yes) in prolog
        will tell the system you have headache
102 answer(Q, Answer):-
103     assert(answered(Q)),
104     (
105         Answer == yes -> answer_h(Q); true
106     ),
```

```
107     % check whether wvery thing has been answered, in other words whether ready
            to make diagnos
108     symptom_library(Symptom_library),
109     (
110         list_finished(Symptom_library, _, If_symptom_finished),   % whether
                symptom finished
111         (If_symptom_finished -> assert(ready_to_diagnos(true));true)
112     ).
113
114 %% helper function for making diagnos, determin whether one illness is satisfied
115 diagnos_h(X):-
116     findall(A, have_symptom(A), Symptom_list),
117     illness(X, L),
118     is_subset(L,Symptom_list).
119
120 % Interface for making diagnos, collect all satisfied illnesses.
121 diagnos(L):-
122     findall(A, diagnos_h(A), L).
```

### General Idea of Utl Implementation

Two predicates are defined in utl.pl

*list_empty/2* uses list split, is the List can be split-ed there are at least one element inside, otherwise it is empty.

*is_subset/2* check whether each element in L1 is inside L2, if so return true, otherwise fails.

### utl.pl

```
1  %% determine whether a list is empty
2  list_empty([], true).
3  list_empty([_|_], false).
4
5  %% determine whether a list L1 is a subset of L2
6  % eg. is_subset(L1, L2).
7  is_subset([], _).
8  is_subset([H|T], L):-
9      member(H,L), is_subset(T,L).
```

## CLI Demo

User invoke the program by querying *start.* in Prolog prompt, and follow the instruction of the displayed message. Every time, user makes response with *response(Q,A).*, answer is

stored, next question will be displayed. If the whole process of asking finished, after user making a response, the system will automatically print the diagnose result.

```
-> % swipl CLI.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
%% invoke the program================================================================
%% ask for pain level================================================================
?- start.
<   broad smile  >
I will help you, but do you feel like manageable pain ?
(response with response(manageable_pain,yes). or response(manageable_pain,no). )
true.

?- response(manageable_pain,no).
<   broad smile  >
My friend, are you feeling unbearable pain ?
(response with response(unbearable_pain,yes). or response(unbearable_pain,no). )
true.

%% ask for mood level================================================================

?- response(manageable_pain,yes).
<   joke   >
No worries, but do you feel like angry ?
(response with response(angry,yes). or response(angry,no). )
true.

?- response(angry,no).
<   beaming voice  >
My friend, do you feel like weepy ?
(response with response(weepy,yes). or response(weepy,no). )
true.

?- response(weepy,yes).
<   look composed  >
I will help you, but do you feel like chill ?
(response with response(chill,yes). or response(chill,no). )
true.

%% ask for symptom================================================================

?- response(chill,no).
<   greet   >
```

```
I will help you, but are you feeling blood ?
(response with response(blood,yes). or response(blood,no). )
true.

?- response(blood,no).
<   look composed  >
Well, are you feeling sweat ?
(response with response(sweat,yes). or response(sweat,no). )
true.

%% .......

?- response(cough,no).
<   look composed  >
My friend, do you feel like ache ?
(response with response(ache,yes). or response(ache,no). )
true.

%% giving diagnose result============================================================

?- response(ache,yes).
<   greet   >
I will help you, but you might have fever.
true.
```

## Conclusion

With well defined rules and predicates, we could easily add new illnesses, symptoms, or editing existing data, with minimal modification to the logic. With the use of http library, user could access the program from any clients, which also largely reduced the difficulty and complexity of dependency problems of compiling and executing the project.

# Appendix

## Screen Shots of GUI

User should start the server by running *swipl CLI.pl* in bash, or running *['server.pl'].* in Prolog prompt. Then, access GUI with http://localhost:5000 in any local web browser.

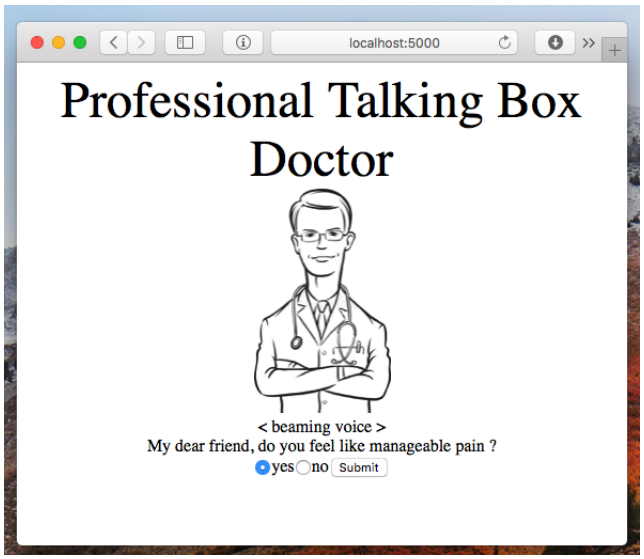


Figure 1: Question 1

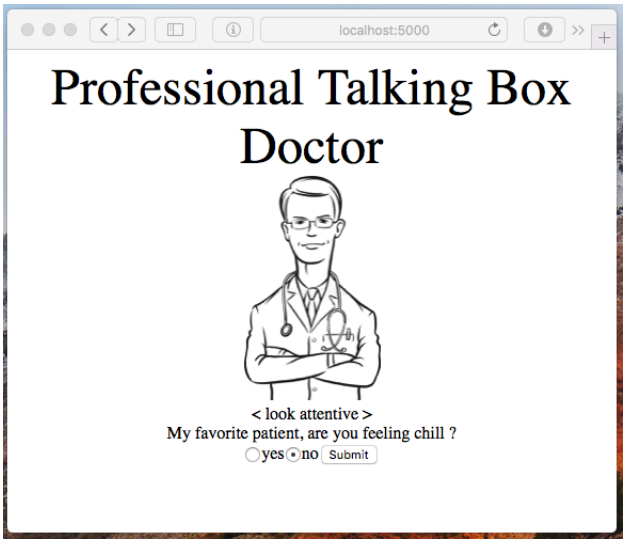


Figure 2: Question 2

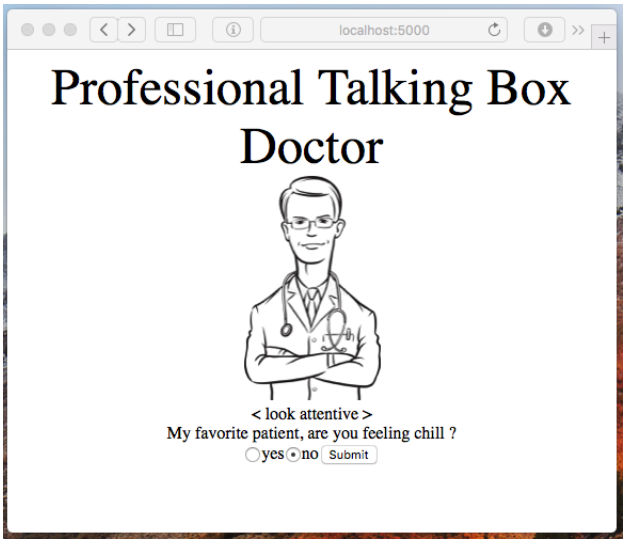


Figure 3: Result