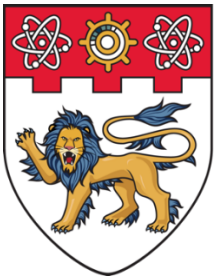


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ3005 ARTIFICIAL INTELLIGENCE

Lab 2 Logbook

Q4 Sympathetic doctor

Lab Group TS5

Soh Jun Jie, U1521123B

CZ3005 ARTIFICIAL INTELLIGENCE LAB 2 LOGBOOK

INTRODUCTION

For this assignment, I have developed an interactive node.js-prolog sympathetic doctor chatbot as well as a pure prolog-based sympathetic doctor chatbot.

The interactive chatbot is less intensive on the prolog engine since only rules and facts about the mood, pain, gestures, patient illness and symptoms and their relations with another are defined. Predicates for illness counter list were also implemented. The prolog engine in this case act as a database for the interactive program to query.

On the other hand, the pure prolog-based sympathetic doctor chatbot I created is very prolog intensive. If a prolog intensive chatbot is what you are looking, please skip the explanation on the interactive chatbot and go directly to the pure prolog base chatbot section.

However, I believe the predicates used in my interactive chatbot project contains sufficient complexity as most predicates mentioned in the assignment hints were implemented.

DOCTOR CHATBOT BACKGROUND

The doctor is diagnosing a speech-impaired patient that can only response yes and no. The patient is also very sensitive to doctor's gesture and will cry if the doctor is insensitive. Worst still, the patient condition might worsen.

To diagnose the patient, the doctor will first ask the patient his mood and pain level. Base on that, the doctor will perform the relevant gestures in his diagnosis to be sensitive to the patient.

During the diagnosis, the doctor will ask the patient about a random symptom. Having that symptom means having a probability the patient is struck with the illness associated. After all the symptoms are examined, the doctor will declare the patient affected by 1 illness base on an illness counter that the doctor was keeping track of.

The illness declared by the doctor is the one with the most times its symptoms was observed. If 2 illnesses result in a tie, only the first one in the illness counter will be declared.

INTERACTIVE DOCTOR CHATBOT

Folder location: interactive_prolog_doctor/

1. High level architecture

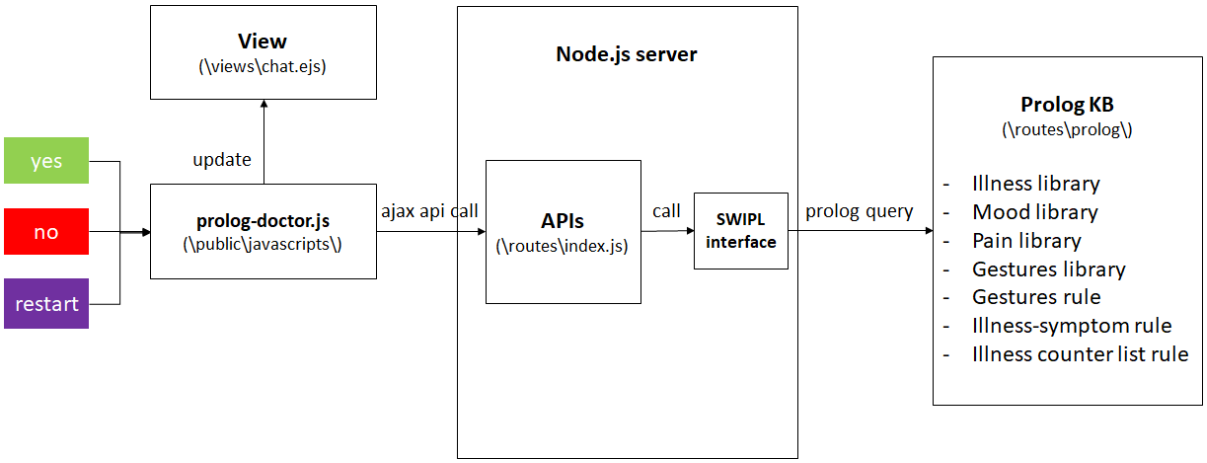


Figure 1: High level overview of the interactive doctor chatbot

SWI-Prolog was integrated with Node.js for the development of this interactive doctor chatbot. The key components of this high level architecture out of all the components are the prolog knowledge based and prolog-doctor.js component.

In this architecture, the prolog KB act as a database for which the node.js server REST APIs can interface and query with. As RESTful APIs are stateless, the prolog KB use as little predicates as possible that requires keeping of states. As a result, the following are areas which can be queried from the prolog KB through the node.js RESTful APIs.

- List of patient mood
- List of patient pain level
- List of illnesses the doctor knows
- List of gestures instances given patient mood and patient pain level
- Random illness and symptom pairing generated by prolog KB
- An illness counter list constructed by Prolog
- Incremented illness counter list given input illness counter list and the illness
- Diagnosed illness given the illness counter list

The yes, no and restart button are also available. When the button is pressed, the prolog-doctor.js will be triggered.

The prolog-doctor.js controls the state of the conversation and maintain an illness counter. It determines the chatbot subsequent action when the patient reply yes or no. At the same time, this component also updates the view which the user is looking at.

We can visualise prolog-doctor.js as a finite state machine diagram in the following page.

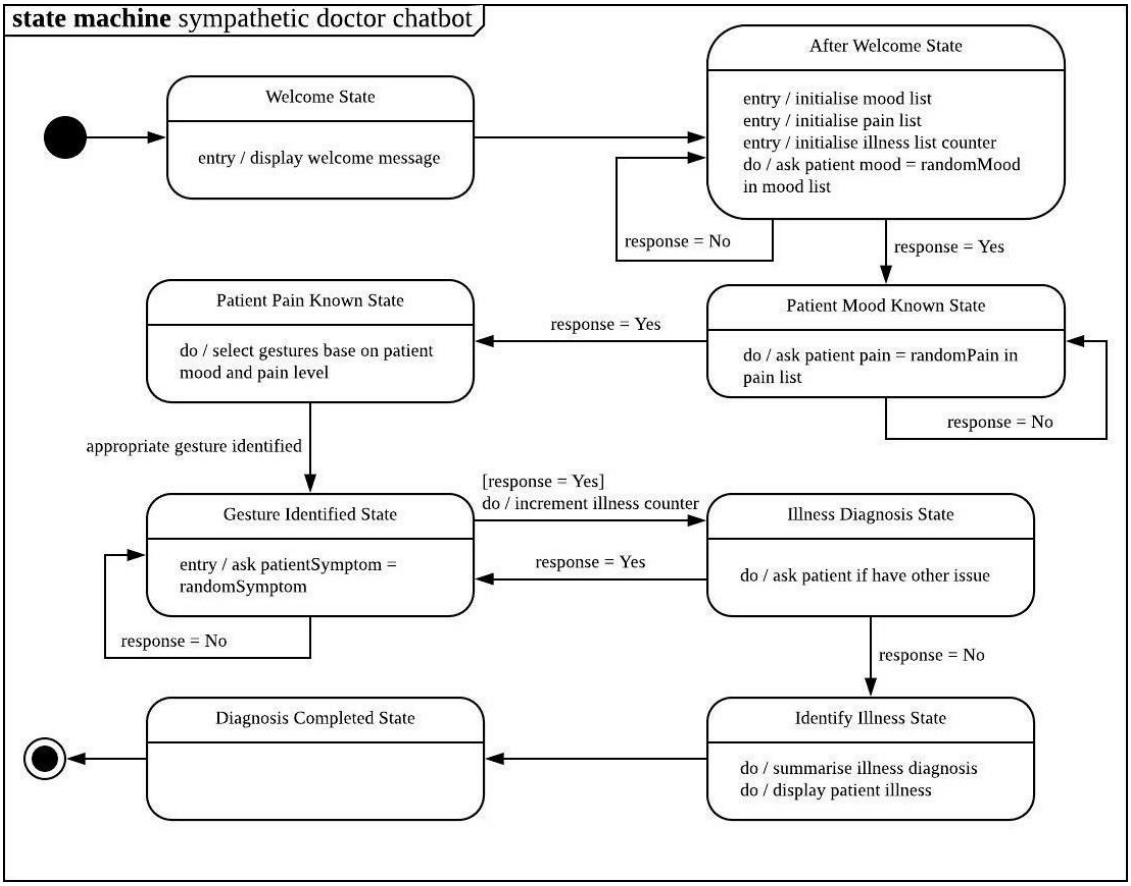


Figure 2: State machine diagram of interactive doctor chatbot

Other than the yes and no button which is provided for the patient to reply yes and no, there is a restart button as well. The restart button can be pressed at any point to reset the entire state machine. This will restart the patient diagnosis.

2. Analysis of predicates used

While the prolog script are properly documented, I still wanted to provide a summary of the analysis of predicates used in the interactive doctor chatbox program.

1. appropriate_gesture(Pain, Mood, X)

   Given a pain level and mood level, the returned X will be appropriate gesture instances

2. get_random_illness_symptom(Illness, Symptom)

   Calling the predicate with Illness and Symptom as variable will generate a random Illness and Symptom pairing

3. get_illness_index(Illness, Index)

   Given a particular illness registered in the illness library, Index variable will return as the list index as recorded in the illness library list

4. build_illness_list_counter(List)

List variable will return as a list with zeros as all its elements and length equal to the number of illnesses in the illness library.

5. increment_illness_counter(List, Illness, Res)

   Given List as the illness counter list and Illness as the illness, Res variable will return the illness counter list that was incremented. It works by retrieving the index of the illness given and then increment the element in the illness counter list having that index.

6. give_illness_diagnosis(IllnessCounterList, Illness)

   Given the illness counter list, the predicate identifies the list index with highest value and return the Illness that is of the same list index in the illness library.

7. max_list (L, M, I)

   Given L as any list with only its elements as integer numbers, M variable will return the highest element in that list. I variable will return the index of that element.

8. increment_list_at_position(L, I, Res)

   Given L as any list with only its elements as integer number, and an integer I, Res variable will return the updated list L where the list at index position I is incremented by 1.

9. indexOf(List, Element, Index)

   Given any List and any of that list Element as input, Index variable will return the index position of Element in that List.

10. list_length(List, L)

    Given any List, L variable will return the length of that list.

3. Executing the interactive chatbot program yourself

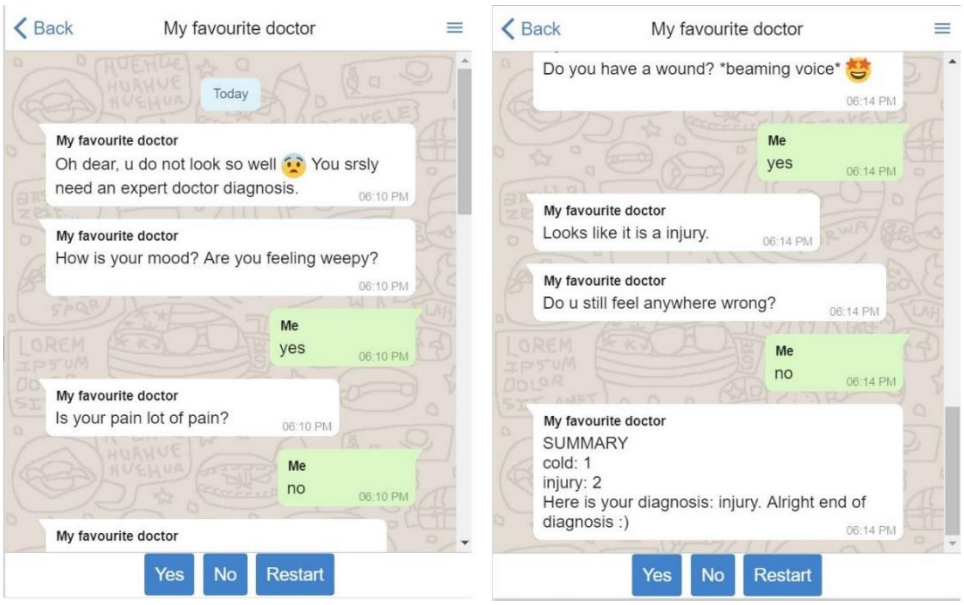
The interactive chatbot program is in the `interactive_prolog_doctor` folder. Please read the README.md file and ensure the pre-requisites for the program are met.

The following are the setup instruction for the interactive doctor chatbot program. This is also mentioned in the README.md file.

1. Ensure that node.js is installed in your machine
2. Node module window build tools must be installed in global using `npm install --global --production windows-build-tools`
3. Ensure SWI-prolog binary is added to environment variable path
4. Install all relevant node.js packages listed in package.json using `npm install` command. Note that this may take some time.
5. Once all packages are installed, execute the command `npm start`
6. Ensure that you have an internet connection, go to web browser and go to the link http://localhost:3000/ to start the chatbot program

4. Sample screens of interactive doctor chatbot

The following image can be found in the `interactive_prolog_doctor/_sample_image/` folder.



5. Video execution of interactive doctor chatbot

Please refer to the `interactive_prolog_doctor/_video/` folder for the execution video.

PURE PROLOG-BASED DOCTOR CHATBOT

Folder location: pure_prolog_doctor/

1. Overview

The pure prolog-base doctor chatbot is composed of the following 4 prolog script, (1) script.pl, (2) gesture.pl, (3) illness.pl and (4) lib.pl.

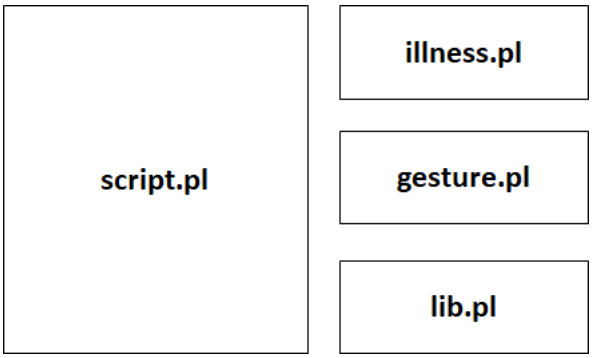


Figure 3: Building block of pure prolog-based doctor chatbot

script.pl governs the core logic of the chatbot program. It contains logic to identify patient pain level, mood level, and can also question the patient about a symptom which will identify the patient illness.

illness.pl contains facts about illnesses, symptoms, mood and pain. It also contains predicates which helps to generate random illnesses and symptoms, construction and manipulation of illness list counter, and identifying an illness from the illness list counter.

gesture.pl contains facts about gestures. It provide the rule mapping between patient mood and pain to appropriate gesturing.

lib.pl on the other hand contains low level library codes on list manipulation which most prolog script written depend on. For example this prolog script contain predicates to identify element of any list given an index. It also contains predicates which can identify the length of a list.

2. Analysis of predicates used

Compared to the interactive chatbot doctor, the prolog predicates used here are more complicated and include the predicates identified previously. Therefore I will not repeat my analysis of predicates that were highlighted previously. Please refer to source code for the implementation as only the brief summary of the predicates are highlighted here.

1. write_list(L)

   Write to prolog console all the element of the List.

2. ask_pain_level(Pain_level)

The predicate will keep asking user a randomly generated pain level recursively until the user reply yes.

3. ask_patient_mood(Mood)

   The predicate will keep asking user a randomly generated mood recursively until the user reply yes.

4. ask_patient_where_wrong(Q, Illness, Gestures, ListCounter)

   The predicate accepts Q as the question string to ask the patient, Gestures as the appropriate gesture instances the doctor must adopt, and ListCounter as the illness counter list.

   It will first write to console question Q e.g. “do you feel anywhere wrong?” and proceed to ask the patient about a random symptom. Having the symptom would increment the illness counter list and would then perform a recursion to ask the patient another question Q e.g. “do you still feel anywhere wrong?”.

   If the reply to question Q is no, then the appropriate Illness diagnosis is returned based on the updated illness counter list.

5. ask_patient_question(Q)

   Write to console the question Q and determine if user reply equals ‘yes’. Predicate fails if not yes.

6. ask_illness_symptom(Illness, Gestures)

   Predicate will generate a random illness and symptom pairing and write to console asking if user has the random symptom with a random gesture instances given in the Gesture list. If the reply is ‘yes’, the associated illness will be returned. If not, a recursion will be performed to ask about another random symptom.

7. start_diagnosis(Illness)

   Predicate will first construct a illness counter list. After which it will identify the pain level and the mood to determine the appropriate gestures list. Illness will be diagnosed with the appropriate gestures. The resultant Illness instance will be written to console for user. The illness variable will return the Illness instance diagnosed.

3. Executing the pure prolog-based doctor chatbot yourself

The pure prolog-based doctor chatbot is located in the pure_prolog_doctor folder. To start the program, follow these steps:

1. In prolog, change working directory to pure_prolog_doctor folder
2. Loading the script.pl file using `['script.pl'].` command
3. Execute the predicate `start_diagnosis(Illness).`
4. Follow the prompt of the chatbot and reply `'yes'` or `'no'` as necessary
5. When responding to chatbot, `'yes'` or `'no'` replies are all in small letters

4. Sample execution of pure prolog-based doctor chatbot

```
?- ['script.pl'].
true.

?- start_diagnosis(Illness).
Is your pain unbearable_pain? (yes/no)
|: no.
Is your pain lot_of_pain? (yes/no)
|: yes.
How is your mood? Are you calm? (yes/no)
|: no.
How is your mood? Are you angry? (yes/no)
|: yes.
Do you feel anywhere wrong? (yes/no)
|: yes.
Do you have a sudden_blindness? *greet* (yes/no)
|: yes.
Looks like it is a myopia.
Do you still feel anywhere wrong? (yes/no)
|: yes.
Do you have a ache? *look_composed* (yes/no)
|: yes.
Looks like it is a fever.
Do you still feel anywhere wrong? (yes/no)
|: yes.
Do you have a sweat? *look_attentive* (yes/no)
|: yes.
Looks like it is a fever.
Do you still feel anywhere wrong? (yes/no)
|: yes.
Do you have a bad_vision? *greet* (yes/no)
|: yes.
Looks like it is a myopia.
Do you still feel anywhere wrong? (yes/no)
|: yes.
Do you have a blurry_vision? *look_attentive* (yes/no)
|: yes.
Looks like it is a myopia.
Do you still feel anywhere wrong? (yes/no)
|: yes.
Do you have a muscle_pain? *greet* (yes/no)
|: yes.
Looks like it is a terminal_illness.
Do you still feel anywhere wrong? (yes/no)
|: no.
Here is your diagnosis: myopia.
Illness = myopia .
```