Project Documentation

Khoa Lai

April 2019

1 Personal Information

Project topic: Home Psychiatrist
Student's name: Khoa Lai
Student number: 732255
Degree program: Data Science (Bachelor level).
Year of studies: 1st year
Date: 21/04/2019

2 General Description

Create a simple program that converses with the user via text. The program parses and modifies the sentences provided by the user, connecting them to form its own responses intelligently. The program may be considered as a sort of "home psychiatrist" that listens to the worries of the user, asks the user questions about the topics that come up and gives "advice". The program may also have some other kind of "personality".

The required level of intelligence of the AI is not particularly high, but it should be able to chat at least somewhat coherently. It should be able to

modify the input from the user enough to know how to give comments or ask questions on the topics talked about. The AI must not be based solely on recognizing specific predefined phrases, and it may not follow any prewritten script. It must try to interpret and analyze the phrases given by the user and respond based on that analysis. When the program does not understand the structure of a sentence it has been given, it may ask the user to try to express it differently or change the subject. The program starts by giving text-based questions on the screen to the user (in terms of graphical user interface). Firstly, it asks the user for the feeling of the user. At this stage, it records the information into a text file, strongly emphasizing on the feeling which it has received. Secondly, based on the feeling input, it goes through a process of evaluation to see which reasonable responses it should give. The program examines the input (which is presented in terms of a sentence) to give response/advice to the user. The user are encouraged to elaborate on their feeling.

The requirement of the program are as following:

- The psychiatrist's AI must me modifiable. The data for this should be loaded from a text file (you are free to choose the format)

- The user's sentences must be parsed and cut into smaller parts - Graphical user interface
- The option to save the conversation into a file (e.g. "the psychiatrist's archive") by giving the desired filename in the command prompt. If the file is not empty, the conversation is added on to the end of the file. The start of each conversation should include a date.
- The psychiatrist should base its speech on the entire conversation, not just the previous sentence.

Level of difficulty implemented: Moderate-hard.
***Interesting feature:** While the user elaborates on their feeling, the psychiatrist gives an inspirational quote for as an encouragement to the user.

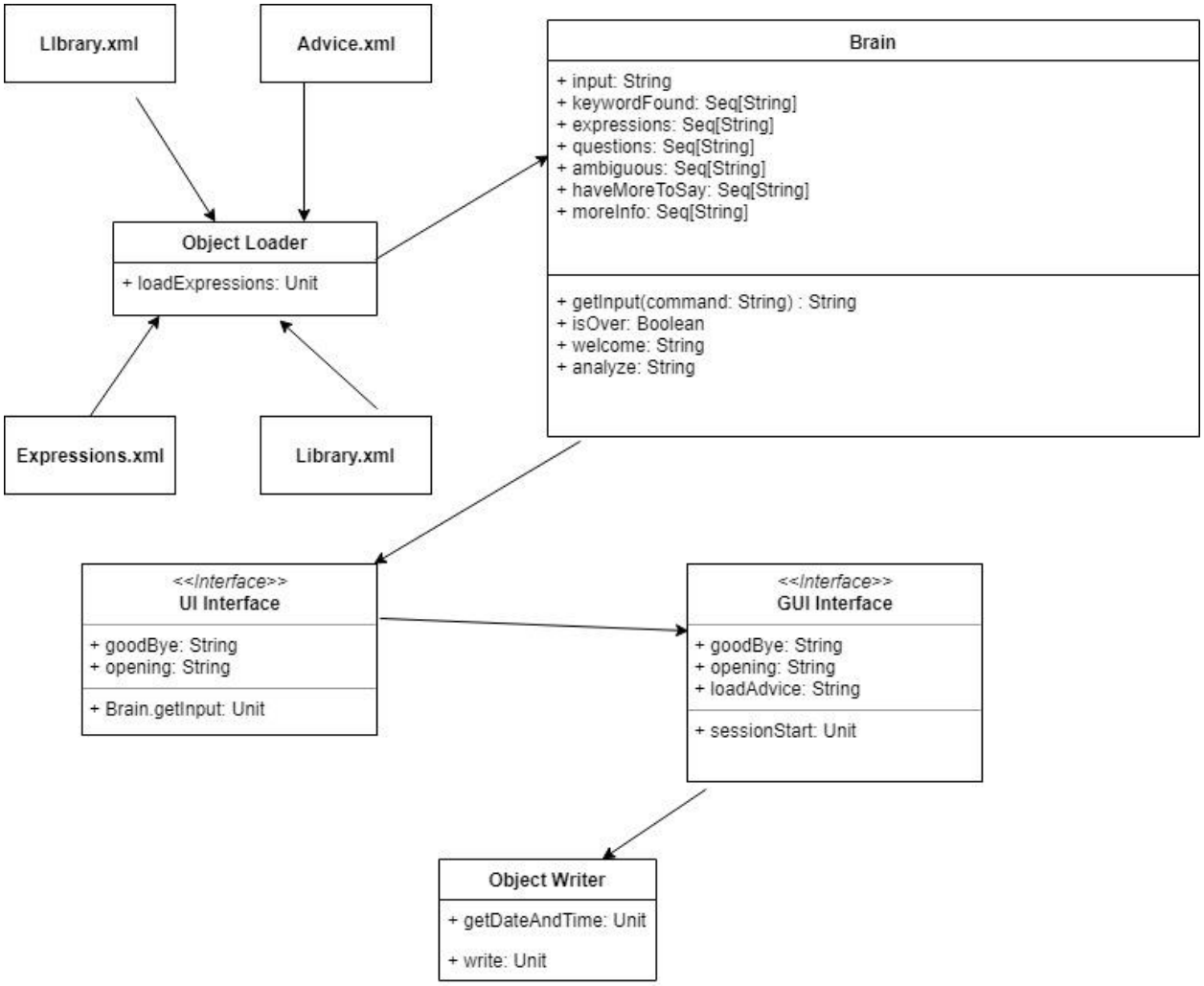
3 User interface

The program first sends a welcoming message to the user and instructs the user to elaborate about their feelings, until the user might want to end the session by saying 'good bye'. The form in which the user can give is in free form, meaning that the psychiatrist can analyze the input while it is still remaining valid. The scope of valid inputs are defined by the 'brain' of the psychiatrist (including blank input, the input is in suspicion of being grammatically incorrect (subject, verb, etc missing), which it might know to ask the user if the

input seems to be invalid/not understandable. If it receives such invalid input, the program repeatedly asks the user to retype the input until valid input is found.

4 Program structure

- Descriptive UML class structure:



- The program is split into sub-parts for scalability and extensibility, also end up being easy to fix if error occurs.

Below is the describe of the final class structure and its use in modeling individual part of the problem:

4.1 Describe how individual problem is solved in each class

- **Object Loader**: helps load the collection of expression into the brain of the psychiatrist. (The collection is modifiable).
- **Key methods: loadExpressions:** used to load the collection of expression from external files to the brain of the psychiatrist
- **Object Brain**: analyzes the sentence by parsing the sentence into subcomponents. The psychiatrist is now able to chat coherently without grammatical error. The brain stores the collection of expression used, so after analysis, the psychiatrist knows what kind of response it should give that are related and reasonable. It defines the rule to know when to end the program and terminate the session. Essential tasks are mainly taken care by this object.

- **Key method:** functions that are used to receive the expression from the loader object. (including: **updateExpressions, updateQuestion, updateAmbiguity, encouragement, talkMore, uncertainty, blank).** **analyze**: the core method of the object, to parse the sentence into several components and analyze to give response based on the input.
- **Object Writer**: writes the conversation into external files. The user then be able to see what have been recorded so far.

-Key method: - **External files (expressions.xml, library.xml, Signaling.xml)**: Used to store the expression needed to answer the user's input. The files are modifiable. The collection of expression also represents the intelligence of the psychiatrist by being able to respond to a wide variety of input.

5 Algorithm

When the program receives the input from the user, it parses the input into several components (subject, verb, object). To know the subject of the input so that the psychiatrist knows how to respond using appropriate pronoun. In addition, the input is parsed into being analyzed, and based on each responses, the program try to interpret

and analyze the phrases given by the user and respond based on that analysis.

The intelligence of the psychiatrist is defined below:

- **Case 1:** If the user says some phrases that are not totally uninformative but more information is needed (for example: I don't know, yes, no, I suppose, I guess), the program indicates that the user should specify in more details about the problem. It is in normal sense that the program should be encouraging the user to specify more details to avoid having insufficient information for analysis. **- Case 2:** When it finds out the user has entered blank input, which it deems to be invalid, it repeatedly asks the user to retype again until valid input is found.
- **Case 3:** When there is not a keyword, the program responds with a remark that lacked content, such as "I see" or "Please go on" to encourage the user to specify more details.
- **Case 4:** When the program has found an important keyword, the psychiatrist analyzes and gives response based on the keyword it receives.

6 Data Structure

-Load from external files: The program loads the data to its brain from XML files. XML files used are

mutable, meaning the files are modifiable so that the user is able to configure the script at their own favor. The syntax is not difficult to write the data and read it from the object Brain. The alternative choice can be text file, depending on the developer's own favor. **-Store the expressions:**After loading the data from the external file, the data is stored in Seq[String]. These are well-defined rules for input analysis of the psychiatrist. Every rule has its own Seq[String] so that it might become convenient when locating the bug and how the logic has gone wrong.

7 Testing

- **Test case 1:** The program should be able to switch the pronoun as it analyzes and then produces correct output with correct pronoun.
- **Test case 2:** The program should end when the user says 'good bye' to the command.
- **Test case 3:** The program should be able to load data from external files. **- Test case 4:** The program should ask the user to retype when it receives a blank input.

All the tests implemented above are passed successfully. The program works correctly.

8 Files and network access

The program works with XML file, which is used to store the data (rules, scripts, expression) and load it into the ”Brain” for analysis. Different rules are written in single nodes, and to be used to load by the Brain’s method that take care of that specific rule. It is comfortable to access to the attribute of the collection, and the content of each node is modifiable, easy to use.

9 Known bugs and missing features

Although the collection of expression that is already defined is sufficient to chat coherently with the user, it is always nicer to continually contribute to the collection of expression. The larger the collection, the more human-like the program will be. Other than that, I don’t think there is any another bugs that might crash the program.

10 3 best sides and 3 weaknesses

3 best sides:

- The program is able to chat coherently with the user, making the user feel like he/she is encouraged well enough to elaborate on the feeling.

- The external file is used good enough for modifiability and extensibility. Changes can be made directly into the external XML file so that the user can modify the intelligence of the psychiatrist at their own favor.
- The program reacts correctly to the special case where it might potentially crash the program or make the program function incorrectly. **For example:** blank input: ask the user to retype until input that can be analysed is received, 'quit': the session is terminated when the user types 'quit'.

3 weaknesses:

I think the weaknesses of the program are well-covered. Some potential bugs that might crash the program have been solved, including invalid input, the user types 'good bye' but the program still takes place.

11 Deviations from the plan, realized process and schedule

There are several tasks that are matched to the expectation listed in the plan. Most of the time on the project is spent on building the databases for the psychiatrist. The proposed plan was different from what is happening when embarking on implementation. Firstly, the core logic behind the

psychiatrist's analysis ability is not consuming as much time as expected, as well-prepared plan has reduced the workload. The Unit Testing did not consume as much time as defined in the plan, I realized it might come from well-commented code and well-split sub-tasks. Secondly, the Unit Testing has been helpful a lot.

12 Final evaluation

In general, the project works greatly matched to the requirement (moderatehard difficulty). However, in order to make the psychiatrist act more human-like, the collection of expression should be updated gradually if I have more time. The good aspect is that the program is suitable for modifiability and scalability, in terms of the external file that the user can modify at their own way. The user can make the psychiatrist respond in their own favor, which is good for a general project. Moreover, the choice of loaded data is XML, which I find to be easy to load and store data, the format is also clear and concise, one spends not much time familiarizing with. Other data structure choice can be made more effective by having cases for the feeling of the user. When the user enters the input, the psychiatrist will try to find keyword, which is similar to what I have done, but to narrow down

the topic and using cases would be more effective than storing keywords in the List. If I start the project again, I maybe should contribute more to the collection of expression and I will get feedback from others testing my program, and configure the psychiatrist to be more human-like.

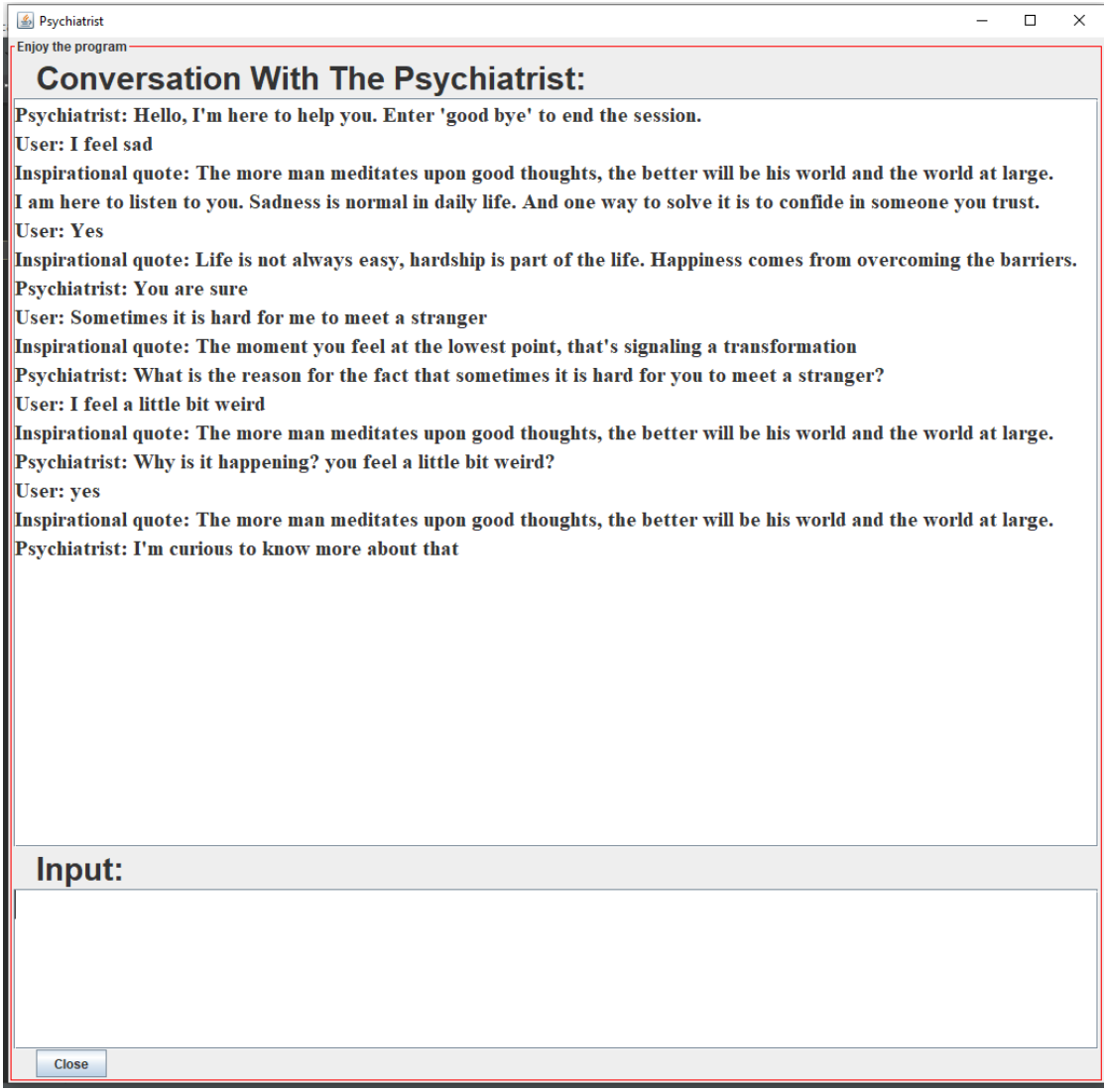
13 References

- Suggested approach to the problem: https://en.wikipedia.org/wiki/ELIZA
- Consulted source of real psychiatrist: https://www.youtube.com/watch?v=Ce3kPwfBfEct=39s

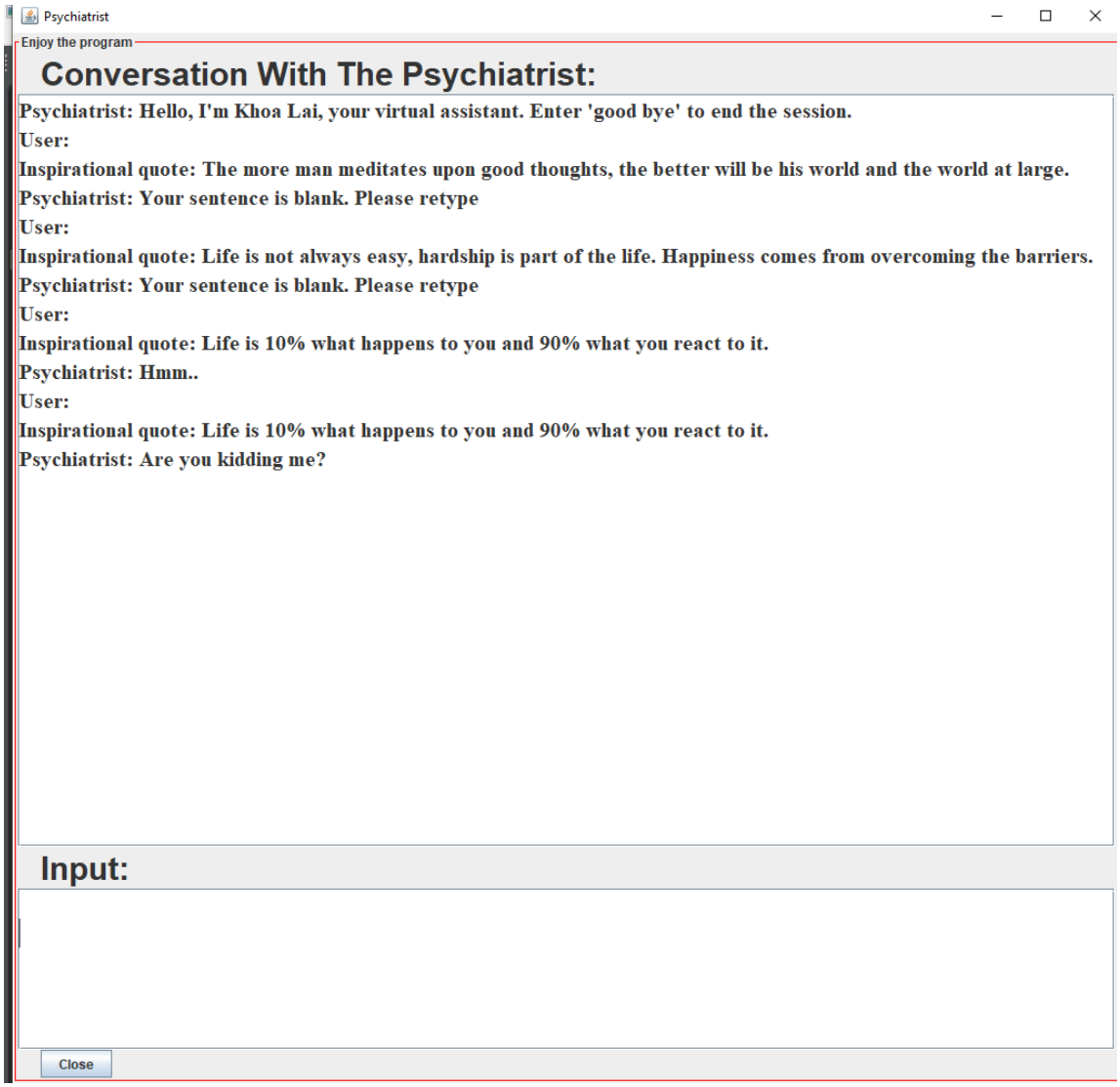
14 Appendixes

Some illustrative examples of the operation of the session:

- **Case 1:** The input is valid. Hence the session operates correctly.



- **<u>Case 2:</u>** The user has entered an blank input. Hence it detects and asks the user to retype until valid input has been found.



Enjoy my project :D