CSC 200(H) URCS Chatbot Advisor

Updated 2/21

0 Change Log

- 2/21. Changes to Final Reporting and Presentation.
- 2/10. Initial version.

1 Project Description

Restricted Domain Dialogue Agents: Pawlicki-in-a-Box[1] UR's new president has a problem. Again! She has proposed expansions that total the better part of a billion dollars. And she had careful financial plans as to how to achieve this. But a new audit shows that things will fall far short, money-wise.

After some energetic checking, the President has found out the key cause of the predicted shortfall. Shortly after a CS faculty member was used to help improve the payroll software (there is no record of which one helped, although the four comment lines among the 150,000 lines of new programming are each signed "TFP," surely some deep code and one that the school's best minds have so far been unable to crack), CS faculty member Thaddeus ("Ted") F. Pawlicki's salary suddenly jumped by a multiplicative factor of well over one thousand. By a complete coincidence, Ted has just announced his departure for a recently endowed chair at the very prestigious University of We-Don't-Have-an-Extradition-Treaty-with-the-USA. Unfortunately, due to remaining deficiencies in the payroll software, if the President refills Ted's position, the new hire must start at no less than Ted's most recent salary.

So the President has reluctantly decided to leave the position open. Through shifting the teaching of some of the department's less rigorous courses to the Interpretive Dance Program and some course-assignment shuffling regarding the remaining courses, the President has managed to handle the course-coverage loss. However, she is deeply worried about the loss of Ted's famed advising skills. So who does she call? Yes, the students of CSC200(H), who so very recently rose so well to the task of "safe programming." In particular, President Mangelsdorf asks you to write a "Pawlicki-in-a-Box" program. This program must handle the advising of all freshman/freshwoman and sophomore CS premajors. Its goal is to provide a strong level of advice, support, and insight to such students, by fulfilling the conversational role that Ted fulfilled when students walked into his office.

Here is the framework. You do not have to tackle speech recognition or speech synthesis at all. We'll use a totally text-oriented interface. Also, you may assume that this version of Ted doesn't have access to students' advising records (although if you want, your programs can keep files that stay around between runs of the program, and thus allow it to become better and better as it gets more and more experience), and so during the conversation with "Ted," "Ted" will have to get from the student (whom you may assume to be a fresh- man/freshwoman or sophomore) who has come to see him whatever information "Ted" needs to identify what help the student wants and to provide advice to the student on that issue. In particular, when the program starts up, we'll assume that a student has just walked into "Ted"'s office. Your program should then (always) greet the visitor by printing out the formulaic two lines:

[1]Professor Lane Hemaspaandra created this project and last used it in Spring 2018. His project description is so vivid and fun that it is copied here verbatim with minimal/minor changes/updates.

```
> Hello.  What can I do for you today?
> (over)
```

(Note: Every line "Ted" outputs must start with the `>` character and then a space. This will make it easier to visually scan the conversation logs.) More generally, both "Ted" and the student will signal that they are done with what they are currently saying by having a line of input that simply says "(over)" in the student's case and "`> (over)`" in "Ted"'s case (and the student is allowed to choose to interchangeably use "(over)" and "(o)" and "."—the student might often choose to type "." as it is faster and easier). Note that the student in reply might not give "Ted" all the information "Ted" needs, and might not even pose any coherent issue, and so "Ted" may have to enter into a dialogue with the student. As to what that dialogue might look like and how long it might be, well, as students, you'll have a general idea of that. (Among the types of issues students might naturally come to "Ted" for are issues of what courses to take, prerequisites, getting ready to declare the major, summer jobs, courses, and much more.) And in the 2/21 recitation session you'll have a chance to show off your program's strengths—and to have your program put energetically to the test by other groups.

There will be a second set of special phrases that when on a line by themselves by convention will indicate that the student is leaving "Ted"'s office; "(over and out)" and "(oo)" and "bye" may be used interchangeably for that. "Ted" will never end a conversation on its own; only the student can invoke those session-ending key-phrases. (All of these key-phrases are case-sensitive, by the way. "(OO)" on a line by itself is not a valid sign-off but rather is just a strange parenthetical exclamation to "Ted.")

Of course, writing a perfect such program is (currently) pretty much impossible. In fact, even doing it nondisastrously is a wildly daunting task that is probably also pretty near impossible and would fill any senior AI researcher with terror. (So be very thankful that you are not senior AI researchers; no one likes being filled with terror. In fact, naïveté can be an asset: Róbert Szelepcsényi was an undergraduate, resolved—in the affirmative—the huge open issue of whether the context-sensitive languages are closed under complementation probably largely because he didn't know enough to know that the problem was hopelessly hard. . . so he just went right ahead and solved it. . . correctly!) However, you'll want to do your best to have your program work as well as possible.

Good luck, and do well. President thanks you in advance for saving her big-chunk-of-a-billion dollar plan (and the rumor that she is kicking back a certain percentage of the savings to your course's professor is, most unfortunately, quite untrue).

2 Requirements

Scientific Collaboration The project is not competition among groups, rather a collaborative effort routine in scientific research. Each is a research group sharing a similar interest and objective and may take similar or different approaches to solve the same problem. You will practice open collaboration and learn to interact and communicate effectively to pool ideas and examine them from different perspectives through collective work so to avoid detours, dead-ends or re-inventing wheels for yourself and for others.

Programming You must implement your advising tool in a safe programming language that forbids NULL pointer de-references at compile time. Examples choices are Rust and Haskell.

Partnering You must complete the assignment in a group of three or four people.

Code Sharing We encourage groups to collaborate. You are always allowed to discuss and share ideas. In addition, we allow students to share code *after obtaining permission from the course staff.* Note that it is always your responsibility that your chatbot works correctly.

Mini-symposium On Friday February 21 during the recitation time, 4pm to 5pm in WH3201, each group is required to participate and bring a prototype for other groups and the teaching staff to try and to answer their questions. The purpose of the symposium is to share problems and solution ideas and to solicit and provide feedback.

Final Reporting and Presentation Each group is required to submit a report on the system design and evaluation by mid-night Tuesday March 3. We allow an automatic extension to Friday March 6. Specific guidelines and requirements on the report and the presentation will be given in a supplement on Monday February 24.

Finally, each group presents the report in class on Monday March 30. The requirements will be given later.