CS4706 - Spring 2010
Project 3 - "Building a Spoken dialog system (SDS)" (200 points)

For this project you will build a simple spoken dialog system using (and modifying as necessary) the TTS and ASR components you implemented for your domain in projects 1 and 2. The SDS should allow the user to ask questions related to your domain. You can structure the interactions you support however you want. You should design your system by first deciding what types of user interactions you will support and then creating a Call Flow Diagram, a directed graph showing the System States (e.g. Greeting, Help, Info-request, Exit) as nodes with arcs showing which state can follow which other state (E.g. Greeting can be followed by Info-request or Help or Exit, so there should be arrows connecting the Greeting node to each of these 3). You will identify these states in your SDS by considering the current state, the legal states this state is connected to in the graph, and the user input.

Your SDS will be evaluated on a) the complexity of the domain; b) how well the system handles that complexity; c) the interactions you support; and d) how well the system performs. For complex domains, you may wish to control the interaction, so that the user can only ask about a limited number of things at a time. For example, for an airline reservations domain in which users can ask about cities, dates, times, airlines, and flight numbers, you might want to restrict users to specifying only one or two pieces of information at a time, e.g. "I'm interested in flights from Chicago to New York." Then "I want to leave on May 15 at 10 a.m." Use your judgment based on what makes the most sense for your domain and on the performance of your ASR system. The system prompts should make it clear what type of queries or information the user is expected to provide on any given turn.

Your SDS should support the following:

- An initial Greeting and brief statement of the purpose and capabilities of your system.
- Processing of user input related to your domain, creation of a semantic representation of the input, identification of SDS state, lookup of information requested in a table or on line, creation of text to be output, and output TTS of that text.
- Vocal confirmation for ending the interaction.
- Generation of a System log file myUniSDSlog<sessionID> for each dialog, including:
  - Date/time stamp for each system and user turn
  - System state recognized before each System prompt
  - Text of System prompts
  - Recognized user input
  - System's semantic representation of user input

  - SDS state.
  - System response.
  - Filenames for .wav files of each system output and user input., associated with the appropriate System text entry and User recognized text entry.
  - You may include other relevant information in your log files as you see fit. Document the information your log provides and its format in a README.myUni.log file.

Here is an example of a minimal System log file:

```
4/29/10 14:40:03
User: Hello.
Recognized string [hello]
Semantics [greeting]

State [Greeting]

System: Hello.  I amMathTutor,  here to help you with math. I can help you with
algebra or geometry or calculus. Please tell me what you would like help on.
<myUniSessionNumSysOuputNum.wav>

User:  I need help with triangles.
Recognized string [Need help a triangle]
<myUniSessionNumUserInputNum.wav>
Semantics [requesttype=help, topic=triangle]

State [New-Info-Request]

System: What help with triangles do you need?
<myUniSessionNumSysOuputNum.wav>

User: I would like to know how many sides are on a triangle.
<myUniSessionNumUserInputNum.wav>
Recognized string [like know many sides triangle]
Semantics [requesttype=know-about,  topic=triangle, property=sides]

State [Current-Info-Request]

System: A triangle has three sides. Do you have any more questions about
triangles?
<myUniSessionNumSysOuputNum.wav>

User: No.
<myUniSessionNumUserInputNum.wav>
Recognized string: [no]
```

Semantics [no]

State [New-Info-Request]

System: *What other objects do you need help with? Or say Goodbye to exit.*
<myUniSessionNumSysOuputNum.wav>

User: *Tell me how to find the circumference of a circle ?*
<myUniSessionNumUserInputNum.wav>
Recognized string [find the circle]
Input: [Requesttype=find, Topic=circle]

State: [Current-Info-Request]

System: *You want to know about circles?*
<myUniSessionNumSysOuputNum.wav>

User: *Yes.*
<myUniSessionNumUserInputNum.wav>
Recognized string [yes]
Semantics [yes]

State [Confirmation]

[. . .]

User: Goodbye and thanks very much.
<myUniSessionNumUserInputNum.wav>
Recognized string [goodbye]
Semantics[exit]

State [Exit]

System: *Goodbye and thank your for using MathTutor.*
<myUniSessionNumUserInputNum.wav>

Optional Features

You will notice in the example above that this system also has the capability of asking for ***confirmation*** if it is not sure it has understood the user input. You may also want to include ***help information***, in case the user becomes confused about what information the system can provide or other things. You may also want your system to be able to ***backtrack to a previous state*** when you realize that system and user hold different beliefs about the course of the dialogue. You may also think of additional features that would improve your system. If you include any of these optional features, you should describe them fully in your pdf file describing your system, as discussed below.

Note

Your script will need to include commands to play and record wav files. You can do this with following commands (the same ones used by the script to record your prompts for the TTS) in Festival:

- $ESTDIR/bin/na_play $outputFile.wav
- $ESTDIR/bin/na_record -f 16000 -time $duration -o $inputFile.wav

Submission and Presentation

You should submit the following:

- Your executable script along with any additional software source code you've written that it calls
- The grammar file for your (modified) ASR
- The textual log file and related wav files for a sample dialog with the system you recorded that shows the features of the system. The dialog should include at least 5 user turns. The input and ouput log filenames should be in numeric order and labeled as in the example above.
- A pdf document named myUni.SysDesc.pdf describing:
    a. The functionality of your domain. I.e. what the domain is about, what sorts of questions can be asked, what (if any) modes it has and a state diagram showing transitions.
    b. Changes you had to make to your ASR and TTS system to implement the SDS.
    c. What tradeoffs you made and why.
- The submitted script should run from any location and save the log files into the current directory. The script can call into your project directory to run the TTS and ASR components and any SDS-specific software. Please make sure files have execute permissions to run from any user.

Upload these files in one zip file proj3-myUni.zip (e.g., proj3-rc2447.zip) to courseworks.

In addition to the online submission of your system, everybody will demo their system to the class in the timeslot reserved for the class final (Wednesday May 12, 1:10-4:00pm). Each person or team will be given 7 minutes to briefly describe their domain and demonstrate their system in use. We will allow an additional minute or two for questions. The class final presentations will be held in the Speech Lab.