**CSCI-599 Natural Language Dialogue Systems**

**Final Project Report**

# Developing and Evaluating a Spoken dialogue system based on the popular game 'Who wants to be a Millionaire'.

**Objective :**

To develop and evaluate a spoken dialogue system on the lines of the popular television show 'Who wants to be a Millionaire' by using an existing speech recognition system (CMU's Sphinx 4) and a text-to-speech synthesizer (FreeTTS).

**Introduction :**

**Why a dialogue system is suitable for games** :

By definition, a video game is an electronic game that involves human interaction with a user interface to generate visual feedback on a video device. Since their evolution, developers have tried to make games more and more interactive to enhance the user experience.  Ever advancing technology and production values related to video game development have fostered more lifelike and complex games which have in turn introduced or enhanced genre possibilities.

A spoken dialogue system for a game is one such genre, which aims at making a game more realistic for its audience. Virtual interaction in natural language is bound to improve the user experience in, if not all, certain kinds of games. One such example is the popular game show - Who wants to be a Millionaire.

**The Game Concept:**

'Who wants to be a Millionaire' involves two users - the Host (our system) and the Player (the user). The player has to answer 15 consecutive questions correctly if he wants to become a millionaire. A question is asked by the Host and the Player has to answer it correctly (making possible use of three lifelines discussed below). The Player can Quit the game whenever he wants or ask for Help to see the game's guidelines. As each question is correctly answered, the amount of money won increases according to the question's worth.

The following lifelines are available for a Player and each can be used only once :

50-50 (fifty-fifty) : cross out two wrong choices and continue.

Audience Poll : provide a hint to the question (retrieved from the question bank).

Phone a Friend : replace with another question.

**What I'm trying to achieve** :

Through this project, I intend to develop a spoken dialogue system for the above mentioned game, such that the user gets an experience as realistic as the show one sees on television. The purpose of this project is to develop an interactive game which integrates the existing speech recognition and manipulation capabilities, and evaluate the efficiency of the system as a whole.

**How does it relate to the topics discussed in the course** :

This project integrates an existing speech recognition system and a TTS system into a Java application. The Sphinx 4 system is a trusted and reliable speech recognition system written in Java, while the FreeTTS system, also written in Java, is a popular choice for converting text to speech. This project integrates the concepts of 3 important components in a dialogue system - automatic speech recognition, natural language understanding and text-to-speech synthesis, into an exciting, interactive game.

**Project Description**

The system would consist of two users (not shown on screen) - the game host and the player. It uses a system-initiated approach in which the host carries the conversation forward. At any point of time, the typical interaction between the host and the player would consist of the following steps :

A question with 4 options is displayed on the screen. The host reads out a random question retrieved from the question bank, and the options to the player. He waits for an answer, books an option once the user's answer is comprehended, and declares the result.

**How the project works**

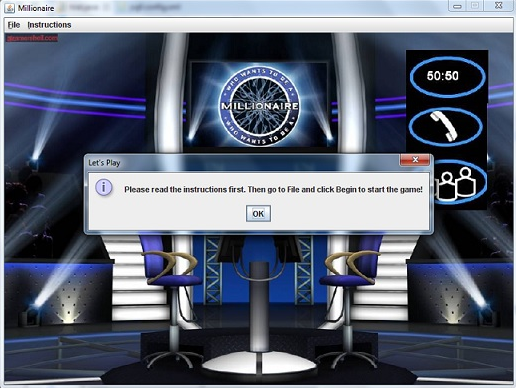
1. **Loading questions and answer options** : A question can be appended to the questions.txt file, and corresponding answer options (four in number) must be appended in the answers.txt file. For simplicity, please put the correct answer option first.

2. **Creating a grammar file :** The program loads the questions and answers from the files, and uses the content to create a grammar file (millionaire.gram) for the speech recognizer. This is a very important step because the recognizer will try to match your recorded speech with the content in the grammar file.

3. **Evaluation report :** After the game has ended, the system generates and displays an evaluation report informing the player of the session's accuracy in terms of number of responses correctly and wrongly recognized.

**Example Scenario**

Step 1:



Step 2: Click OK, read the instructions, then go to File and click on Begin. The first question pops up.

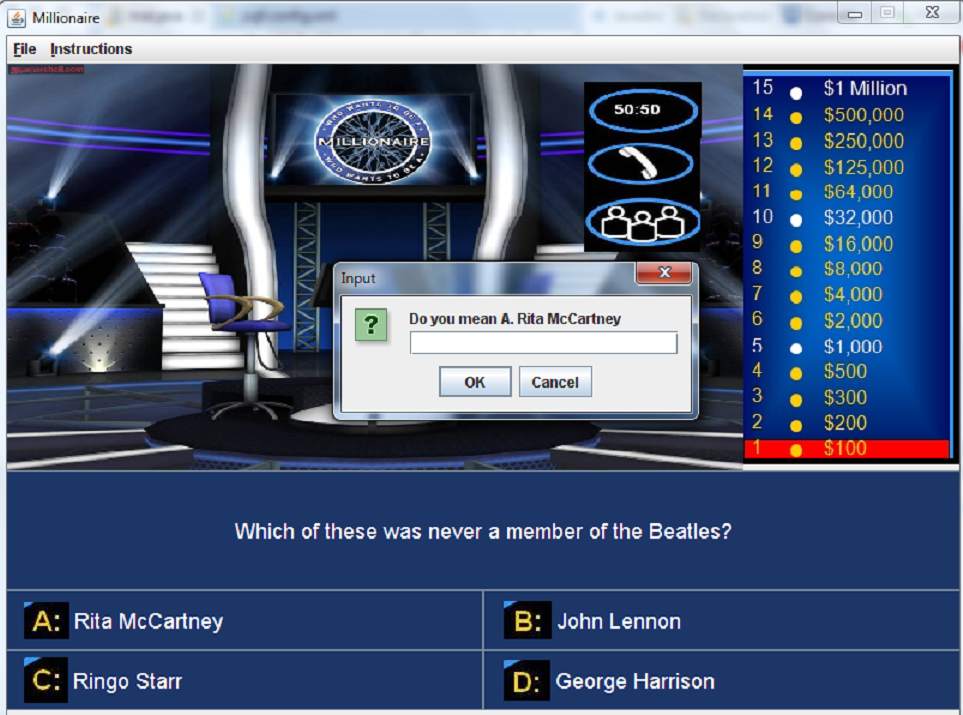


Step 3: Wait for the system to read out the question and answer options for you. If you think the answer is option A for instance, do the following steps :

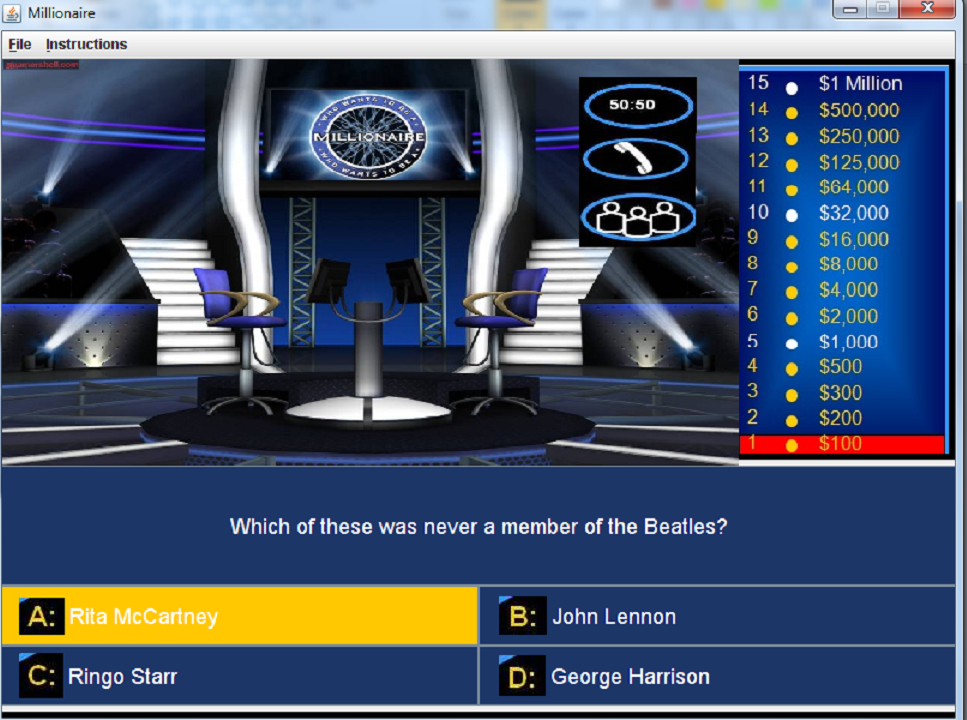
a) Move the cursor on the area above the question panel.

b) Hold down the left mouse button, wait for 2-3 seconds, and record you answer. Once done recording, release the mouse button.

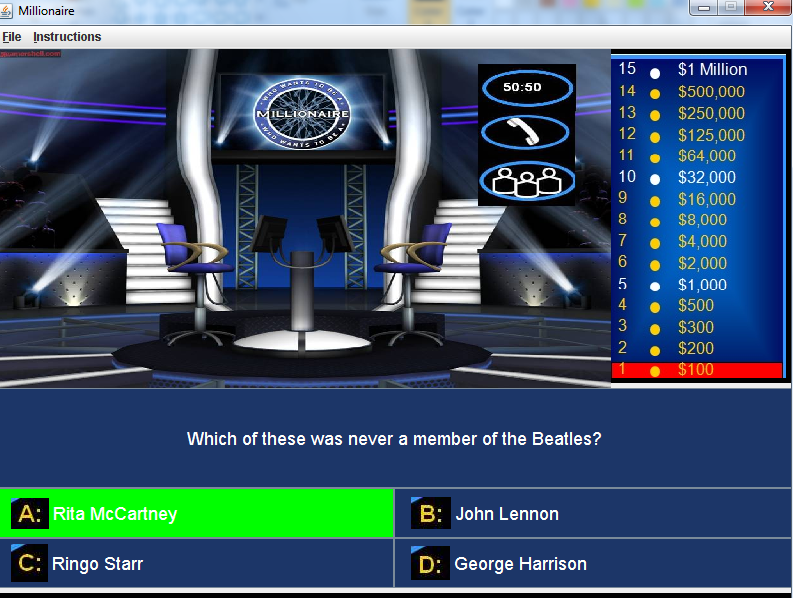
Step 4: If you said "A" and it was correctly recognized, then the following screen will pop up. To lock your answer, click OK, else click Cancel to record another answer.



Step 5: If you clicked on OK, then the system will lock your answer :



Step 6: After about 2 seconds, the system will let you know whether you were right or not.



Step 7: The host will tell you the amount of money won. And then it will move on to the next question.

**Resources:**

The system will use the following existing tools :

1. CMU Sphinx 4 speech recognizer.

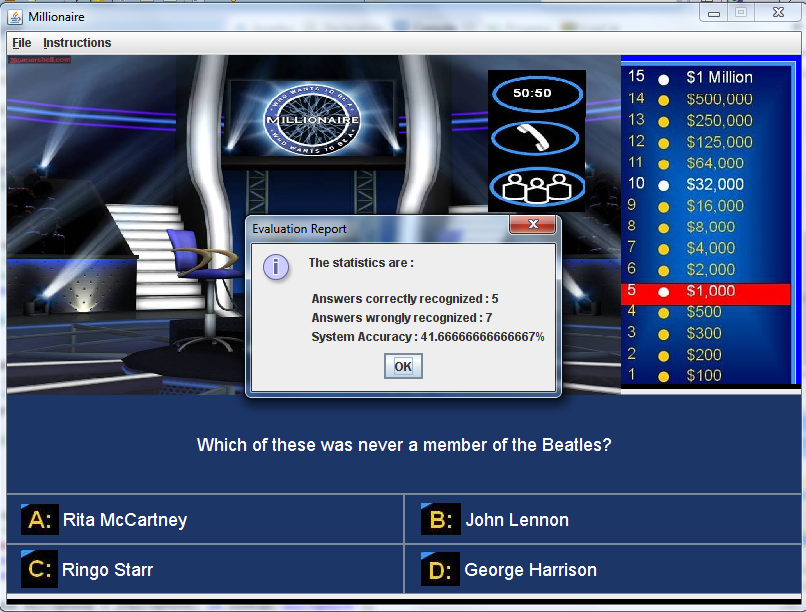
2. FreeTTS (text to speech synthesizer).

3. Eclipse Juno IDE

The system will make use of CMU's Sphinx 4 speech recognizer to listen to the player's speech which will then be manipulated by the java program. Once the player output is confirmed, the system will see if it is the correct answer. For converting the text to speech, a text to speech synthesizer (FreeTTS) will be used.

**Evaluation :**

Generating a report about the efficiency of the speech recognizer (and therefore the program as a whole) is handled by the program itself. After the game ends i.e. either the player quits/exits or loses, the program will automatically generate and display a report like the following :



**System Performance**

The system gave a mixed performance. Sometimes, it scored an accuracy rate of over 60%, and on other occasions it gave an accuracy of as low as 20%. Several factors determine the accuracy rate :

1. The project's efficiency is limited by the efficiency of the Sphinx4 speech recognizer.

2. The player's response is matched with what is there in the grammar file. Therefore, contents of the grammar file limit the scope of correctly recognizing user speech. For example, the user may utter something like : "This is a difficult question!" which our program will not find in the grammar file it has. Thus it will match the user input to a wrong sentence in the grammar file.

3. The user's choice of words play an important factor in determining the system's accuracy. There are a whole lot of ways a user can give his response. Not handling one such response will result in a wrong output.

4. Sometimes, the accent of the user may also affect the performance.

The following table shows the system accuracy during several runs of the program :

|  |  |  |  |
| --- | --- | --- | --- |
| Attempt Number | Number of responses correctly recognized | Number of responses wrongly recognized | System accuracy |
| 1 | 5 | 7 | 41.67% |
| 2 | 7 | 13 | 35.0% |
| 3 | 11 | 7 | 61.12% |
| 4 | 8 | 16 | 33.33% |
| 5 | 2 | 7 | 22.22% |

**Lessons Learned from the Course**

As I mentioned earlier, the topics discussed in the course were organized such that every week I was able to build upon the knowledge I had gained from the past week. Several lectures gave me the motivation to develop this project :

1. Week 3 Speech recognition in dialogue systems : I came to know about the Sphinx4 system, the use of ASR in dialogue systems and the resources that are needed (grammar, language model, dictionary and an acoustic model).

2. Week 6 lecture on evaluation of dialogue systems.

3. Several papers contributed to my learning experience. I came across several hitherto unknown terms which I later read about and understood. The paper "CMU's robust spoken language understanding system" by Sunil Issar and Wayne Ward gave me further insight into this subject.

**Issues Known with the Project**

While testing my project, I came across the following issues :

1. Sometimes the speech recognizer recognizes your input incorrectly for several times in a row. In such a case, please click the left mouse button on the screen, and click OK. You may have to do this several times. The system will generate outputs even though you haven't said anything. After a certain number of trials, you'll see that it has started to recognize your inputs correctly.

2. It is preferable to use single words as inputs to the host. For example : "A" or "elephant" or "quit" or "poll".

**Future Work**

1. The grammar file developed for this project is intended to incorporate only certain kinds of user responses. This can be built upon in the future to develop a more comprehensive grammar.

2. Using actual human speech instead of the robotic voice used in the project will make the project more interesting.

3. Introducing a virtual human as the "host" asking the questions to the player.