CleverNAO:
The Intelligent Conversational Humanoid Robot

Jessel Serrano

April 15th, 2015

Independent Study: Artificial Intelligence

Spring 2015

Instructor: Dr. Janusz Zalewski

Software Engineering Program

Florida Gulf Coast University

Ft. Myers, FL 33965

1. Introduction

"Can machines think?" This is the question asked by Alan Turing which has since spawned numerous, passionate debates on the subject of artificial intelligence [1]. It has also spawned the famous Turing Test, a test which determines if a particular machine (or algorithm) can pass as a human. Since its inception, the Turing Test has, in fact, been passed by a few artificial intelligence algorithms. Some of these algorithms represent the latest technology in terms of artificial intelligence.

Artificial intelligence is part of a broad field called cognitive science, which is simply a study of the mind and the way it works. For the purposes of cognitive science, artificial intelligence is defined as " a codification of knowledge will finally explain intelligence" [2]. However, when it comes to engineering, the purpose of artificial intelligence is to use knowledge to solve real-world problems. One of these problems, similar to the problem of the Turing Test, is how to make an artificial device or creature appear more human. To address this problem, a technology has been created called chatbots. These are artificial intelligence algorithms that process natural language and, using the analysis that results from the processing, outputs an intelligent response. It is the utilization of these chatbots that is the main concern of this project.

In this project, a chatbot will be linked to a robot so that the robot will verbally speak what the chatbot's response is to a human. The chatbot used in this project is Cleverbot [3], a chatbot created by Existor [4] and the robot is the NAO robot [5]. The intended result is an application, named CleverNAO, that facilitates a verbal conversation between a human and the NAO robot, which emulates Cleverbot.

The importance of this project can be paralleled to the historical and cultural importance of the Turing Test itself. Although the human participant is not going to be fooled by CleverNAO (due to the physical appearance of the NAO robot), such a conversation between a human and a robot signals a new era in robotics and artificial intelligence. To have a physical robot that is not only able to move, but also hold a verbal conversation is akin to notions of science fiction and truly intelligent robots. One can imagine a future where these robots are the new store clerks, customer service agents, teachers, even friends.

2. Definition of the Problem

2.1 General Overview

When it comes to artificial intelligence and natural language processing, there are six main steps required in having a computer participate in a conversation and to take action (Figure 1):

1. Speech Recognition: Acoustic speech signal is analyzed to determine sequence of spoken words.
2. Syntactic Analysis: Using knowledge of language's grammar, sentence structure is formulated.
3. Semantic Analysis: Partial representation is derived from the sentence structure to articulate the meaning.
4. Pragmatic Analysis: Produces a complete meaning for the sentence that comes with contextual information, such as time, place, speaker, etc.
5. Pragmatic Response: Using the complete sentence and its meaning, respond with a sentence that follows the line of thinking.
6. Physical Emulation: Have a physical figure (robot) verbally speak the response.

As a result, the computer should be able to either respond in a conversation, perform an appropriate action, or make a decision. For example, this response can either be the output of a sentence or the activation of a switch. In the case of this project, the first four steps will be utilized to produce an artificial intelligent entity that can give a response to a person. However, in order to emulate a conversation between the entities, two additional steps are introduced.

The first step will be accomplished via an API that processes human speech via a microphone and translates the acoustic sound waves to sentences in a computer. These sentences will then be passed to the second API which is the chatbot. The chatbot will accomplish steps 2-4 through a series of HTTP connections and selected artificial intelligence algorithms. The final two steps will be accomplished by having a robot verbally say the response. These six steps will then loop in order to produce a physical conversation between human and robot. The entire process is shown in Figure 1.

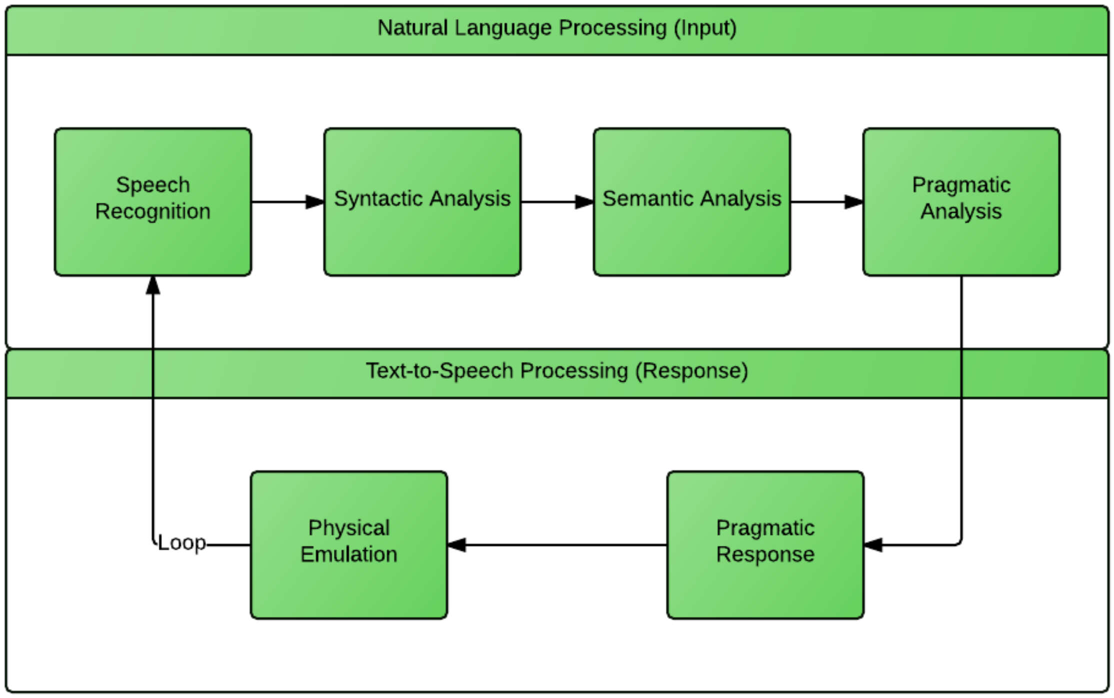


Figure 1: CleverNAO Conversation Cycle

2.2 Chatbot Tool

A chatbot is a program designed to generate text that is imitative of human conversation. When using a chatbot, the user inputs the text via a keyboard, which eliminates the need for the program to recognize the acoustic speech signal. Instead, a syntactic analysis is required for the inputted text in which the program must begin to understand the sentence structure and grammar.

The chatbot algorithm being used in this study will exhibit steps 2-4 continuously. It will take in a String as a parameter and output a response that appropriately follows the conversational tone of the input String. This input String is the result of the first API which converts speech to syntax. The response is a String generated through an HTTP connection with a particular chatbot algorithm. For the purposes of this study, the chatbot algorithm in use is the Cleverbot [3].

Cleverbot is an artificial intelligence algorithm developed by the company Existor. Their goal is “to further develop our AI algorithms to achieve truly semantic understanding of natural language” [4]. Cleverbot learns from people and uses a database of millions of lines of conversations in order to produce a response that is likely to what a human would say. This algorithm allows it to grow and learn from conversing with humans much like an intelligent entity. When selecting a response, Cleverbot searches its database three times in the online version. In order to pass the Turing Test, Cleverbot used 42 databases to produce a response that is semantically and syntactically correct as well as relevant to the conversation as a whole. Cleverbot passed the Turing Test by fooling 59 percent of people that it had conversations with into believing it was an actual person.

Once Cleverbot has been implemented with the Speech Recognition API mentioned beforehand, the fulfillment of natural language processing for a program or entity will be completed for each step. The result will be a response generated by Cleverbot. This response will then be passed to a robot as a String, which will utilize its API to convert text to speech. This robot will be the NAO robot.

2.3 NAO Robot

The NAO robot is an autonomous, programmable humanoid robot developed by Aldebaran Robotics [5]. The NAO contains a plethora of sensors that can be used to detect motion and sound, as shown in Figure 2. Its main connectivity for development is through either a Wi-Fi module or an Ethernet port. This allows for development of the NAO robot over a network or via the Internet.

In this project, the NAO will be manipulated via a network by a program on a computer. The NAO acts as the Server while the computer acts as the Client. This is seen by the way in which the NAO API is called; the API, on the side of the client, is called via calls to the respective method on the side of the NAO's embedded API. This method on the NAO's side is what manipulates the NAO. All that is needed is the IP address and port number of the NAO robot. Using these two pieces of information, one can initialize a connection to the NAO robot by instantiating one of the modules of the NAO's library, NAOqi. In this case, the NAO's text-to-speech protocol will be manipulated so that the NAO will verbally speak the response of the Cleverbot. In this way, the NAO robot embodies the Cleverbot algorithm and becomes CleverNAO. The network connectivity is discussed in more detail in the next section.

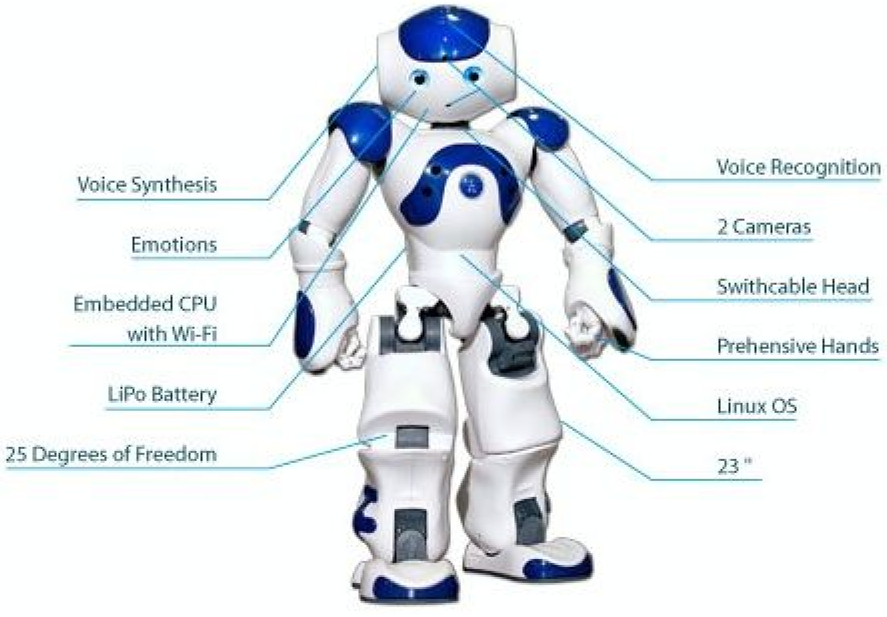


Figure 2: Sensors of the NAO Robot

3. Software Development

3.1 General Introduction

The CleverNAO is composed of six physical entities as shown in the physical diagram in Figure 3. Using a microphone, the user speaks a coherent and clear sentence. This speech is picked up by the microphone and sent to the laptop running CleverNAO. Using a speech API, these audio data are translated from speech to text. This process is essential to CleverNAO and is discussed in detail later in Section 3.3.1. Once the user's speech is converted to text form, the chatbot tool is to give a response to the user. This response is sent to the NAO robot, via the wireless router, in which the NAO will use its speakers to convert the text to speech.

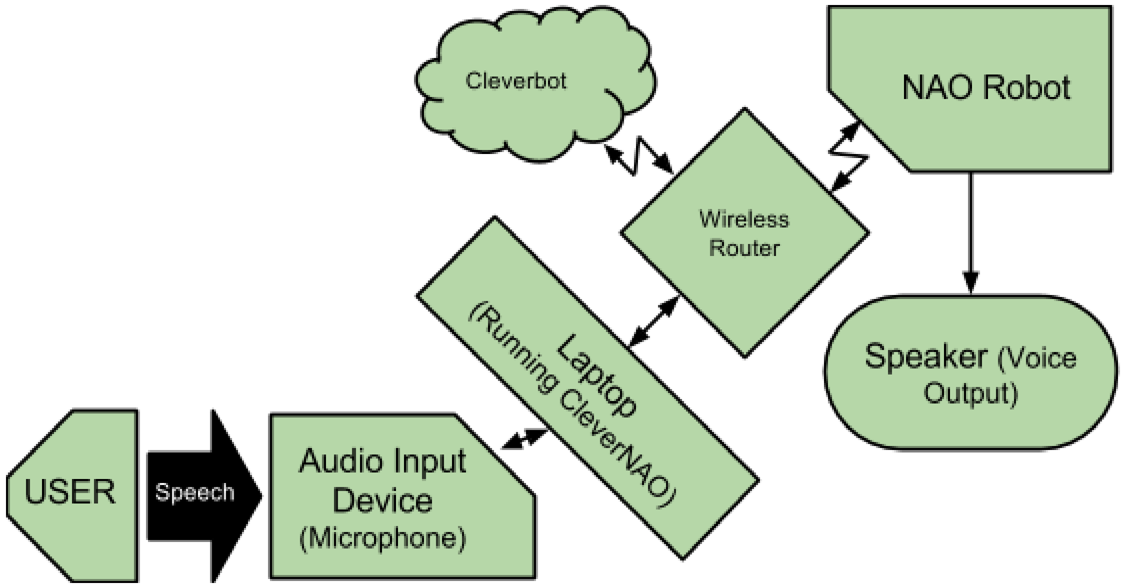


Figure 3: Physical Diagram of the CleverNAO

As can be interpreted from above, CleverNAO is the main program that ties the conversation between the user and the robot. It controls three processes that are necessary for a conversation between the user and the robot to occur: speech processing, chatbot processing, and NAO processing. The name CleverNAO implies that the chatbot used in this project is the Cleverbot, as discussed previously. This process is then renamed to 'Cleverbot processing' to accurately reflect what is happening: using HTTP, a request is sent to the Cleverbot website from the CleverNAO program. In return, a response is received.

These three processes are highlighted in the context diagram, shown in Figure 4. In this diagram, the CleverNAO acts as the main software that takes data from the user, the NAO robot and the Cleverbot website and transforms the data to output responses which will be sent to the NAO and to the display or monitor. It begins with the user by verbal speech to the NAO. The CleverNAO will take in this speech and transform it into an input string using the speech API. The CleverNAO software will then send an HTTP request to Cleverbot.com. This request contains the input string and it asks Cleverbot to provide a response. Cleverbot.com will return with a response that is passed to CleverNAO. This response is also in the form of a String; this response string is then be passed to the display for printing and to the NAO robot.. The NAO will take this response string and, using text-to-speech, speak the response to the user. This will loop continuously and the series of inputs and responses will lead to a conversation between the CleverNAO and the User, implementing a human-to-robot conversation.

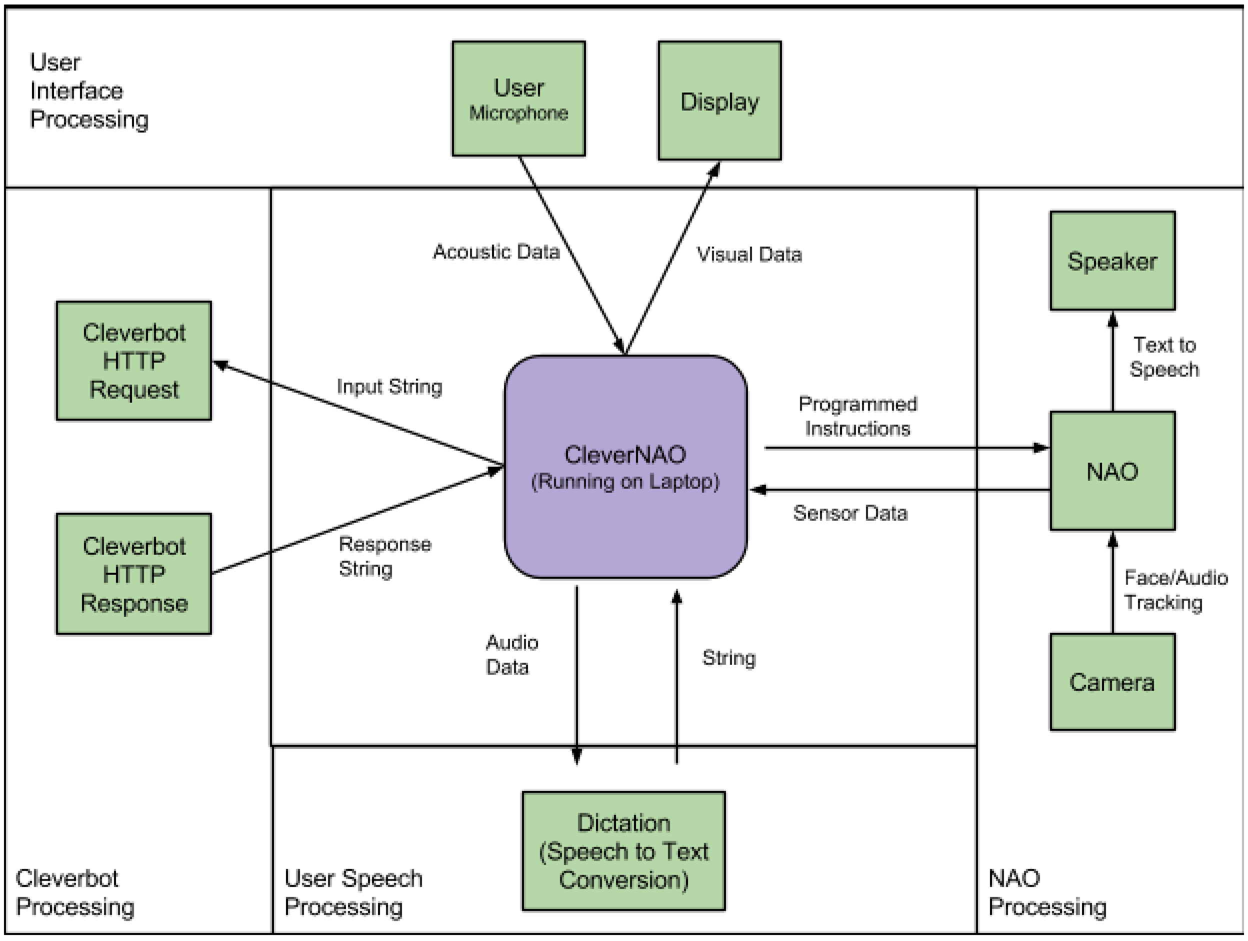


Figure 4: Context Diagram

3.2 Software Requirements

The specific requirements for the CleverNAO software and the NAO itself are listed below.

3.2.1 CleverNAO General Requirements

3.2.1.1 The CleverNAO shall output the conversation to the display in real time

3.2.2 CleverNAO Requirements Specific to Speech

3.2.2.1 The CleverNAO shall take audio data from the user through a microphone

3.2.2.2 The CleverNAO shall accurately translate speech to text

3.2.3 CleverNAO Requirements Specific to Cleverbot

3.2.3.1 The CleverNAO shall send an HTTP request to Cleverbot.com

3.2.3.2 The CleverNAO shall get an HTTP response from Cleverbot.com

3.2.4 CleverNAO Requirements Specific to NAO

3.2.4.1 The CleverNAO shall establish and maintain a connection to the NAO

3.2.4.2 The Cleverbot shall send text to the NAO

3.2.5 NAO Requirements

3.2.5.1 The NAO shall receive text from the CleverNAO program

3.2.5.2 The NAO shall convert text to speech

3.3 Design Description

The CleverNAO program is created using Visual Studio and coded in C#. It is a Windows forms application that utilizes several APIs mentioned previously. The first is Microsoft's .NET Speech library that has speech recognition (speech to text). The second library is an open source Chatter Bot API [6] that allows HTTP connections with three different chatbot algorithms: Cleverbot, JabberWacky, and Pandorabots. The final library is that of the NAO, which contains proxy classes that command the NAO robot. An overall flow of all three libraries and how CleverNAO uses them can be seen in Figure 5.

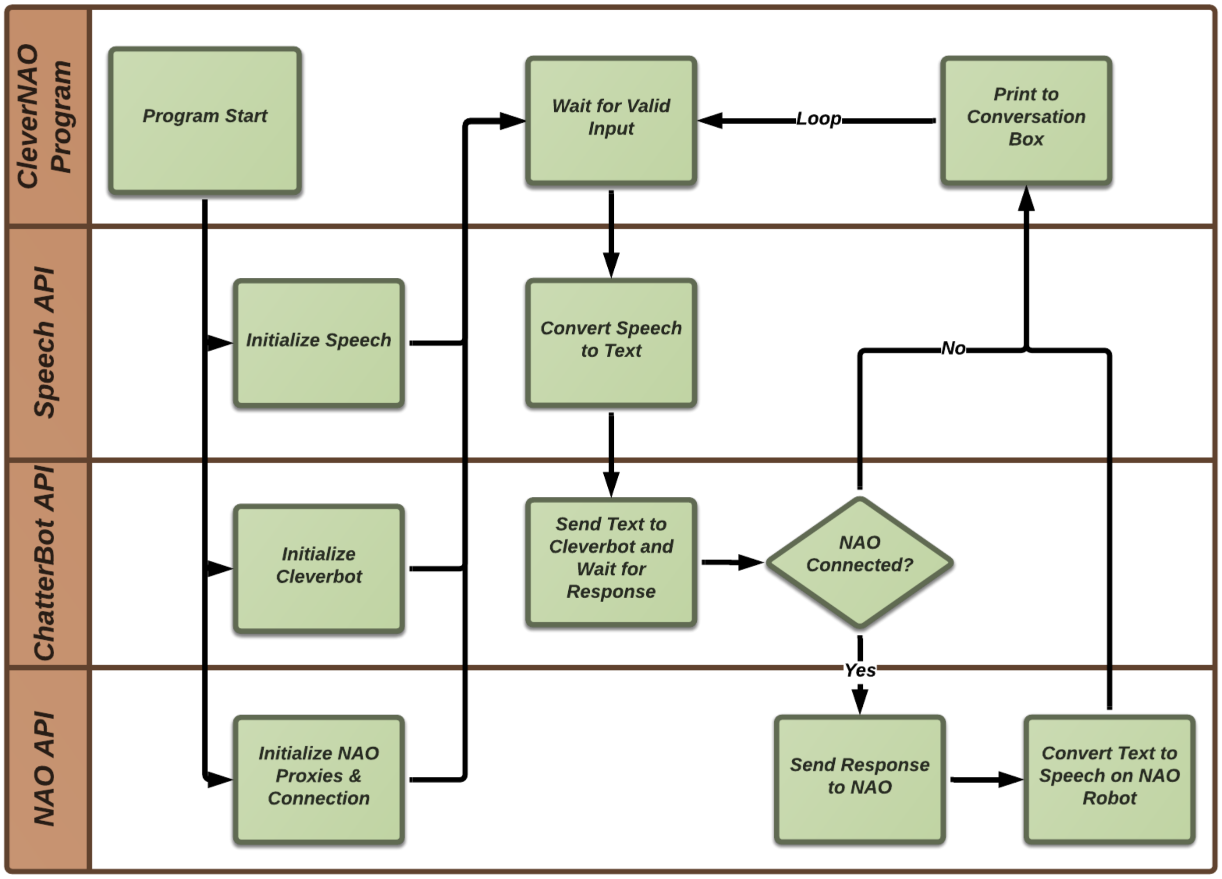


Figure 5: CleverNAO Flowchart

3.3.1 Microsoft Speech Platform

The first process that occurs in the conversation between human and robot is to hear and understand what the human says. How this is accomplished in the development of CleverNAO is through speech recognition software, which takes in acoustic vocal sounds (verbal language) and converts it to text. Since development took place in Microsoft Visual Studio, a native library is used to convert speech to text: the Microsoft Speech API, or SAPI [7].

Within the libraries of SAPI there is a Speech Recognition engine, as well as a Speech Synthesis engine. The latter converts text to speech and is not utilized in this project due to the NAO robot's text to speech capabilities (discussed in the sequel). The Speech Recognition engine converts speech to text by using Grammar objects. "A speech recognition grammar is a set of rules or constraints that define what a speech recognition engine can recognize as meaningful input." [7]. Using DictationGrammar, a grammar that allows free text dictation, SAPI will try to recognize all audio input as speech and convert it to text. To ensure that the speech recognition software does not continuously try to convert audio and background noise, a push to talk button is implemented.

Using the standard Windows Forms API, on a key/button press down, the speech recognition engine will begin to listen for speech to convert to text. Then, when the user has finished speaking, the key/button will be released and another event will trigger causing the speech recognition engine to stop listening and convert what it has heard to text, which will be outputted to the screen. This process is shown in the Figure 6.

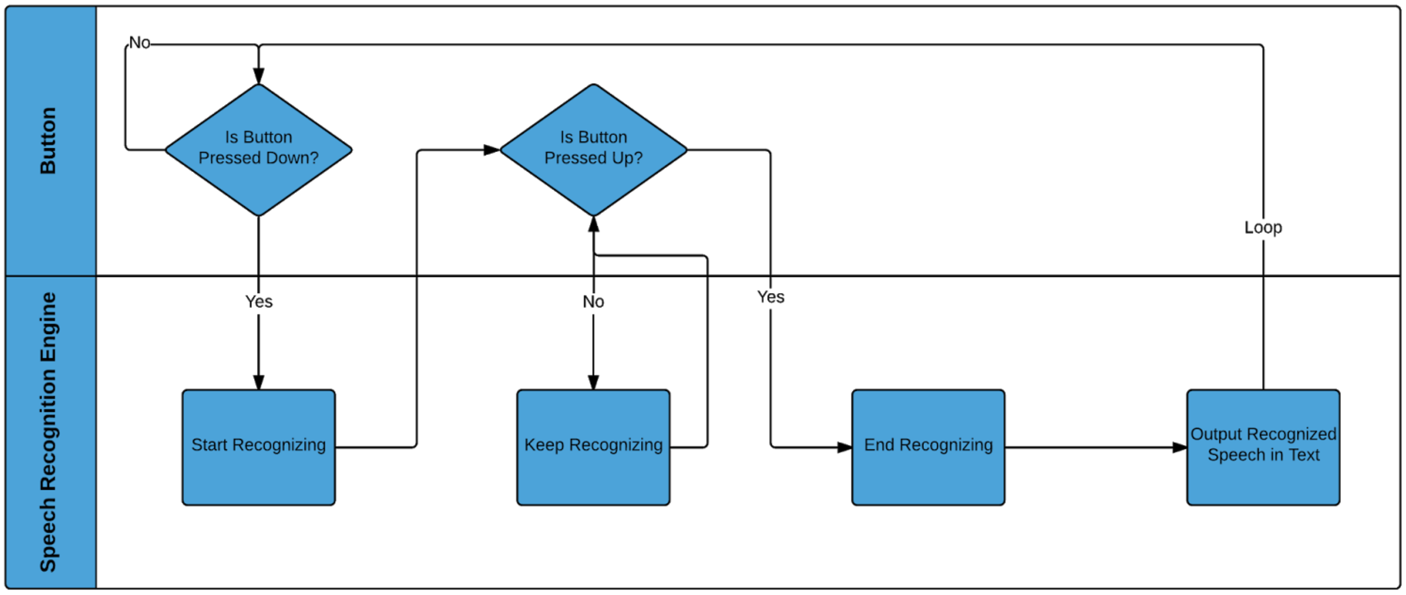


Figure 6: The Speech Recognition Process Flow

3.3.2 Chatter Bot API

Another important process that occurs in the CleverNAO is the conversation with Cleverbot. As noted previously, this conversation is accomplished using an open source library. This library has, within it, the ability to connect with Cleverbot through the Internet and receive a response directly from Cleverbot. It is here that the majority of the Artificial Intelligence algorithm is executed, albeit indirectly. It is executed indirectly because Cleverbot is an already instantiated algorithm that is merely accessed through HTTP connections. What this library specifically accomplishes is the retrieval of a response that has already been built through natural language processing. A basic outline of this process is shown in Figure 7.

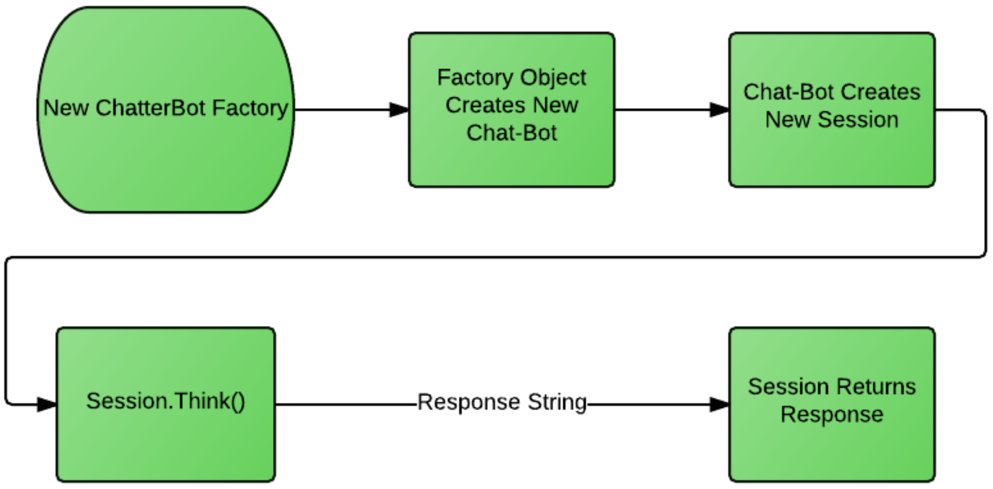


Figure 7: Chatter Bot API Flowchart

As shown in Figure 7, several objects are created before an actual response is received. The Factory object first creates the chatbot object. This chatbot object is instantiated through a web connection with the respective chatbot algorithm, be it Cleverbot or another. This chatbot object then creates a new session which contains the main method used in receiving a response, `Think()`. The `Think()` method takes a string argument. This string is the input that comes from the user or the human. Through HTTP web requests and posts, the input string is sent to the respective chatbot algorithm, is analyzed via their server side, and returned to CleverNAO as a string. This string is the response to the user, to which the user must then choose to respond to.

3.3.3 NAOqi Framework

The final library used in the CleverNAO is the NAOqi library, which is the NAO robot's framework for controlling and commanding the robot itself. How the framework operates is shown in detail in Figure 8 [8]. In this diagram, there is the Broker, which acts as the point of communication between the robot and the caller (the computer running the program). Located on the NAO robot is its own native library that contains methods which mirror what is contained in the Broker. This means that CleverNAO is not actually calling methods on the NAO, but rather calls proxy methods which it then passes to the Broker. The Broker then communicates to the NAO's native library and tells the library to call the methods to which the proxy methods correspond to. In this way, the CleverNAO can maintain a connection with the NAO robot and, using the NAOqi Framework, command the robot with respective calls to its individual modules. These modules are separated by category. For example, there is a module for Motion, Audio, etc.

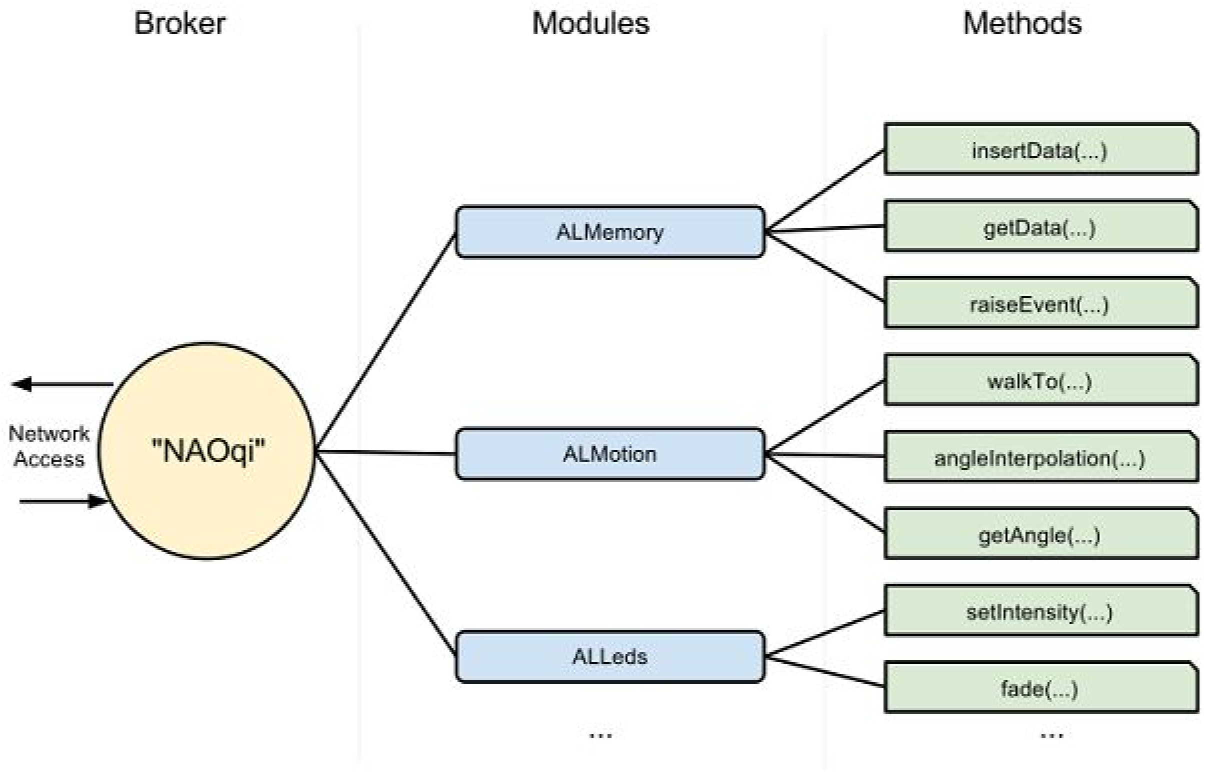


Figure 8: NAOqi Framework Overview

The main module used in the CleverNAO is the Audio module which contains the text-to-speech proxy class. This class is instantiated using the IP address and port number of the NAO robot. Once instantiated, its main method `say()` is called and is passed a string. This string is sent to the NAO via the Broker, in which the NAO will call its native text-to-speech library and actually speak the string that was passed to it.

4. Implementation and Testing

This section deals with the implementation and testing of the three processes: speech processing, chatbot processing, and NAO processing. In addition to these three processes there are two auxiliary processes: the Background Worker and the conversation box that records the conversation between the human and robot, both which are explicated below.

4.1 Auxiliary Processes

The Background Worker is a native library available in the .NET framework that simply performs tasks behind the main application on a separate thread so that the GUI does not lock up. The task that the Background Worker performs in this project is getting a response from the Cleverbot website. Because getting a response incurs a delay due to Internet request times and bandwidth, the application tends to lock up when waiting for a response. By implementing the Background Worker, the user can still interact with the application's GUI while waiting on a response. The following code are the Background Worker's event handlers which tell it what to do when it is called on to do work and when it has completed its work.

```
private void bw_DoWork(object sender, DoWorkEventArgs e){
      BackgroundWorker worker = sender as BackgroundWorker;
      e.Result = getResponse((string)e.Argument);
}

private void bw_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e){
      if (!connectedToCleverBot)
            outputToConvoBox((string)e.Result, 3);
      else
            outputToConvoBox((string)e.Result, 1);
}
```

Once the Background Worker completes its work, it outputs the response to the Conversation Box, which is a rich text box. This is handled using a single helper function that takes two arguments, the string to output and the type of output. There are three types of outputs: a robot output, a human output, and a system output. The system output is used for displaying system messages or errors. The helper function is shown in the following code below.

```
public void outputToConvoBox(string line, int type){
      int length = convoBox.TextLength;
      switch (type){
            case 1: // From robot
                  convoBox.AppendText("Robot: " + line + "\n");
                  convoBox.SelectionStart = length;
                  convoBox.SelectionLength = line.Length + 8;
                  convoBox.SelectionColor = Color.Blue;
                  if (connectedToNAO)
                        sendToNAO(line);
                  break;
            case 2: // From human
                  convoBox.AppendText("Human: " + line + "\n");
                  convoBox.SelectionStart = length;
                  convoBox.SelectionLength = line.Length + 8;
                  convoBox.SelectionColor = Color.Black;
                  break;
            case 3: // System messages
                  convoBox.AppendText(line + "\n");
                  convoBox.SelectionStart = length;
                  convoBox.SelectionLength = line.Length + 1;
                  convoBox.SelectionColor = Color.Red;
                  break;
            default:
                  break;
      }
      inputBox.Enabled = true;
      inputBox.Focus();
      convoBox.SelectionStart = convoBox.Text.Length;
      convoBox.ScrollToCaret();
}
```

4.2 Microsoft Speech Implementation

The implementation of Microsoft's speech recognition begins with the initialization of Speech objects. The first is a Grammar object which needs to be initialized and loaded into the second object, a Speech Recognition Engine. Since what is desired is free text dictation, the `DictationGrammar` object is initialized. Then, a new `SpeechRecognitionEngine` object is created and, using its method `LoadGrammar()`, loads the `DictationGrammar` object. The final step in initialization is to link the Speech Recognition Engine to an event handler that, when speech is recognized, outputs it to the conversation box and calls on the Chatter Bot API to get a response to the recognized speech. This initialization, as well as the event handler, are shown in the code shown below.

```
public void startSpeech(){
      DictationGrammar dg = new DictationGrammar();
      dg.Enabled = true;

      SpeechRecognitionEngine sre = new SpeechRecognitionEngine();
      sre.LoadGrammar(dg);

      sre.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(recognizer_SpeechRecognized);
                  sre.SetInputToDefaultAudioDevice();
}

void recognizer_SpeechRecognized(object sender,
SpeechRecognizedEventArgs e){
      if (!bw.IsBusy){
            outputToConvoBox(e.Result.Text, 2);
            bw.RunWorkerAsync(e.Result.Text);
      }
}
```

As shown in the event handler above, the resultant text is both outputted to the conversation text box and passed as a parameter to a Background Worker object, which runs asynchronously. In order to limit and control the number of times the event handler catches the `SpeechRecognizedEventArgs` event, a push-to-talk button is implemented. When the assigned key is pushed down, the Speech Recognition Engine's `RecognizeAsync()` method is called so that the audio picked up by the microphone will be interpreted as speech. When the key is released, the `RecognizeAsyncCancel()` method is called which cancels any further recognition and converts what it has picked up to text. This is implemented via event handlers and is shown in the code below.

```
void pushToTalk_keyDown(object sender, KeyEventArgs e){
      if (e.KeyCode == Keys.ControlKey && !recognitionRunning){
            try {
                  sre.RecognizeAsync();
                  speechOnOff.BackColor = Color.LimeGreen;
                  speechOnOff.Text = "On";
                  e.Handled = true;
```

```
                    e.SuppressKeyPress = true;
                    recognitionRunning = true;
                }
                catch (Exception ex){
                    MessageBox.Show(ex.Message);
                }
        }
}

void pushToTalk_keyUp(object sender, KeyEventArgs e){
        if (e.KeyCode == Keys.ControlKey){
                sre.RecognizeAsyncCancel();
                speechOnOff.BackColor = System.Drawing.SystemColors.Control;
                speechOnOff.Text = "Off";
                recognitionRunning = false;
        }
}
```

4.3 Chatter Bot Implementation

The implementation of Cleverbot begins with the initialization of several Chatter Bot objects. The first is a Chatter Bot Factory object. It is this Factory object that creates the Chatter Bot object using the method `Create()`. This method is passed a `ChatterBotType` argument which dictates which kind of chatbot is being implemented, be it Cleverbot, Pandorabot, or Jabberwacky. Once the Chatter Bot object is initialized, it calls a method `CreateSession()` which initializes a new Chatter Bot Session object. It is this Session object that takes in strings and returns a response. The initialization is shown in the code shown below.

```
public void startCleverbot(){
        ChatterBotFactory factory = new ChatterBotFactory();
        ChatterBot robot = factory.Create(ChatterBotType.CLEVERBOT);
        ChatterBotSession session = robot.CreateSession();
}
```

Once initialized, the Chatter Bot Session is ready to begin accepting input. This is done through the method `Think()`, which takes a string argument and returns a response as a string. In this project, this is encapsulated within the helper function `getResponse()`. This helper function is called in the Background Worker's event handler that performs the work, as shown previously. Shown below is the helper function, `getResponse()`.

```
public string getResponse(string input){
	try {
		connectedToCleverBot = true;
		return session.Think(input);
	}
	catch (Exception e) {
		connectedToCleverBot = false;
		return "Error: " + e.Message;
	}
}
```

4.4 NAO Implementation

The implementation of the NAO robot begins with the initialization of the NAO proxy objects. The only proxy object that is initialized in this project is the `TextToSpeechProxy`. When initialized, this object takes two parameters, the first being the IP address of the NAO robot and the second being the port number of the NAO robot. Once a response is received by the Chatter Bot API, the response is passed to a method which is called by the `TextToSpeechProxy` object, `say()`. This method is called in the `sendtoNAO()` method, which is shown below. Additionally, this method takes a string argument. It is this string that the NAO will verbally speak. The code that initializes the NAO robot is also shown below.

```
public void startNAO(){
	try {
		TextToSpeechProxy tts = new
	TextToSpeechProxy("192.168.0.102", 9559);
		connectedToNAO = true;
	}
	catch (Exception e) {
		connectedToNAO = false;
		ErrorLabel.Text = "No Connection to NAO";
	}
}

public void sendToNAO(string input){
	tts.say(input);
}
```

4.5 Testing

With respect to the requirements documented in Section 3.2, this section describes the test cases in order to verify the functionality and reliability of the system. These test cases represent partial functionality of the numerous requirements for the system to run properly. In addition, the typical session of conversation between a human operator and a robot is documented in the Appendix.

Test ID	1
Title	Human Output to Conversation Box
Related Requirements	3.2.1.1
Directions	Type anything in the input text box and press enter.
Expected Result	The Conversation Box should display what was typed.
Actual Result	Success

Test ID	2
Title	Speech to Text Accuracy
Related Requirements	3.2.1.1, 3.2.2.1, 3.2.2.2
Directions	Hold down the push to talk key (Control Key), speak clearly, then release the key.
Expected Result	What was spoken should be translated to text and outputted to the Conversation Box.
Actual Result	Failure: The Speech to Text roughly translates what the user said.

Test ID	3
Title	Cleverbot Response
Related Requirements	3.2.1.1, 3.2.3.1, 3.2.3.2
Directions	Type ‘hello’ into the input text box and then wait for a reply.
Expected Result	The Conversation Box should display, in blue, the response from Cleverbot.

Actual Result	Success

Test ID	4
Title	NAO Connection
Related Requirements	3.2.4.1, 3.2.4.2
Directions	Connect to the same network that the NAO robot is connected to. Run CleverNAO.
Expected Result	There should be no error in CleverNAO when on the same network as the NAO robot.
Actual Result	Success: When not connected to the same network as the NAO robot, ‘Not Connected to NAO’ displays in CleverNAO. When connected to the same network, no error displays.

Test ID	5
Title	NAO Text to Speech
Related Requirements	3.2.1.1, 3.2.5.1, 3.2.5.2
Directions	Connect to the same network that the NAO robot is connected to. Run CleverNAO. Type ‘hello’ into the input text box and then wait for a reply.
Expected Result	The response from Cleverbot should display in the Conversation Box and the NAO robot should verbally speak the response.
Actual Result	Success

5. Conclusion

This project is centered on the creation of a physical robot that any person could talk to in the English language. The objective is to pair an artificial intelligence algorithm that processes natural language with a physical robot that could synthesises speech. The result is a successful emulation of a chatbot, Cleverbot, with the NAO robot doing the speech synthesis.

The CleverNAO application had to be built by using three unrelated libraries. The first is a Microsoft Speech library that is used to capture speech and convert it to text. This allows the conversation between a human and a robot to be a spoken conversation as opposed to a written one. The second library is an open source library that creates an HTTP connection to several chatbot artificial intelligence algorithms. In this project, the chosen chatbot is Cleverbot. The final library is the NAOqi Framework and it is used to connect to and command the NAO robot. By combining these three separate libraries, an application is created that facilitates and records a verbal conversation between a human and a robot.

The result is the CleverNAO application that performs exactly as expected. Getting a reply from Cleverbot and having the NAO robot synthesize the reply to speech operates smoothly and concisely. The largest problem, however, lies with Microsoft's speech recognition. This problem is one of accuracy in the conversion from speech to text. Without a noiseless background, the accuracy of the conversion is optimal at best. If any words are not enunciated clearly, the conversion produces different words or does not produce text at all. In order to improve the accuracy of the speech-to-text portion of this project, a different speech recognition library is recommended.

Possible extensions of this project are to improve the speech-to-text process and to animate the NAO during conversation. The latter, animation of the NAO, can be implemented by using either the Face Tracking or Sound Localization modules in the NAOqi Framework. If implemented, these modules will allow the NAO to turn and face the person who is speaking to it, making the interaction more human.

6. References

[1] A. M. Turing, “Computing Machinery and Intelligence”, URL:
http://www.loebner.net/Prizef/TuringArticle.html

[2] J. Friedenberg, G. Silverman, <Cognitive Science: An Introduction to the Study of Mind>, Sage Publications, Inc. 2006

[3] Cleverbot, Existor, URL: www.cleverbot.com

[4] About us, Existor, URL : www.existor.com/company-about-us

[5] Who is NAO?, Aldebaran, URL: www.aldebaran.com/en/humanoid-robot/nao-robot

[6] P.D. Bélander, Chatter Bot API: https://github.com/pierredavidbelanger/chatter-bot-api

[7] Microsoft Speech API (SAPI) 5.3, Microsoft, URL:
https://msdn.microsoft.com/en-us/library/ms723627(v=vs.85).aspx

[8] NAOqi Framework, Aldebaran, URL:
http://doc.aldebaran.com/1-14/_images/broker-modules-methods.png

7. Appendix

This session contains documented experiments with the CleverNAO software and the conversations that result from a human-to-robot interaction. These experiments test the soundness of the CleverNAO system in two parts: without speech-to-text and with speech-to-text. The first experiment, without-speech-to-text, focuses mainly on the verifying the soundness of the conversational capabilities of the robot and if a conversation can be carried out reasonably. This is to say that what is being tested is if Cleverbot can respond to user input without substantial delay, with intelligent responses, and with responses that follow the tone of the conversation. The second experiment, with speech-to-text, focuses on verifying the soundness of the speech-to-text and text-to-speech modules and whether, when implemented, it can handle a fluid and verbal conversation. This is to say that what is being tested is if user speech can be translated to text accurately and if the Cleverbot response can be translated to speech accurately.

7.1 Sample Conversation (Without Speech-to-Text)

This experiment was conducted by running the CleverNAO program on a laptop that is connected to the wireless network that the NAO robot is on (Figure 9). In order to get a response from Cleverbot, sentences were typed into the input box. Upon pressing Enter, the sentences would be inserted into the Conversation Box and a reply from Cleverbot would ensue.

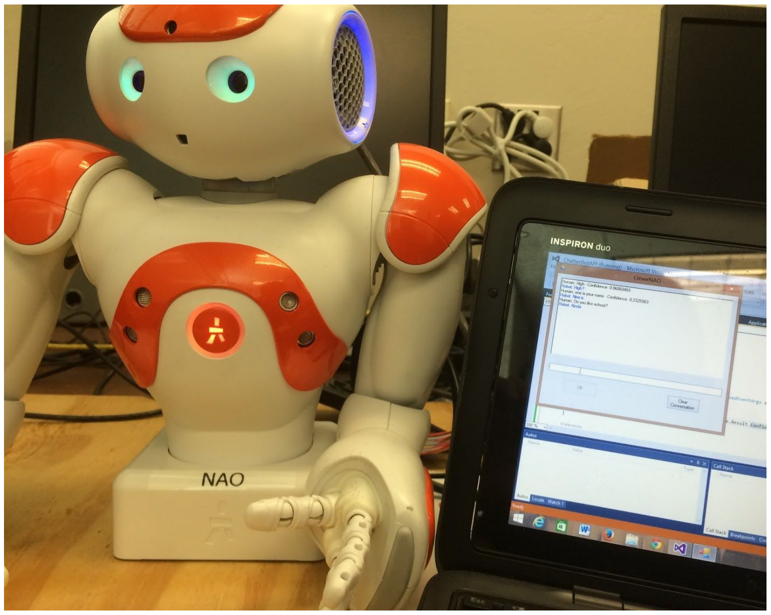
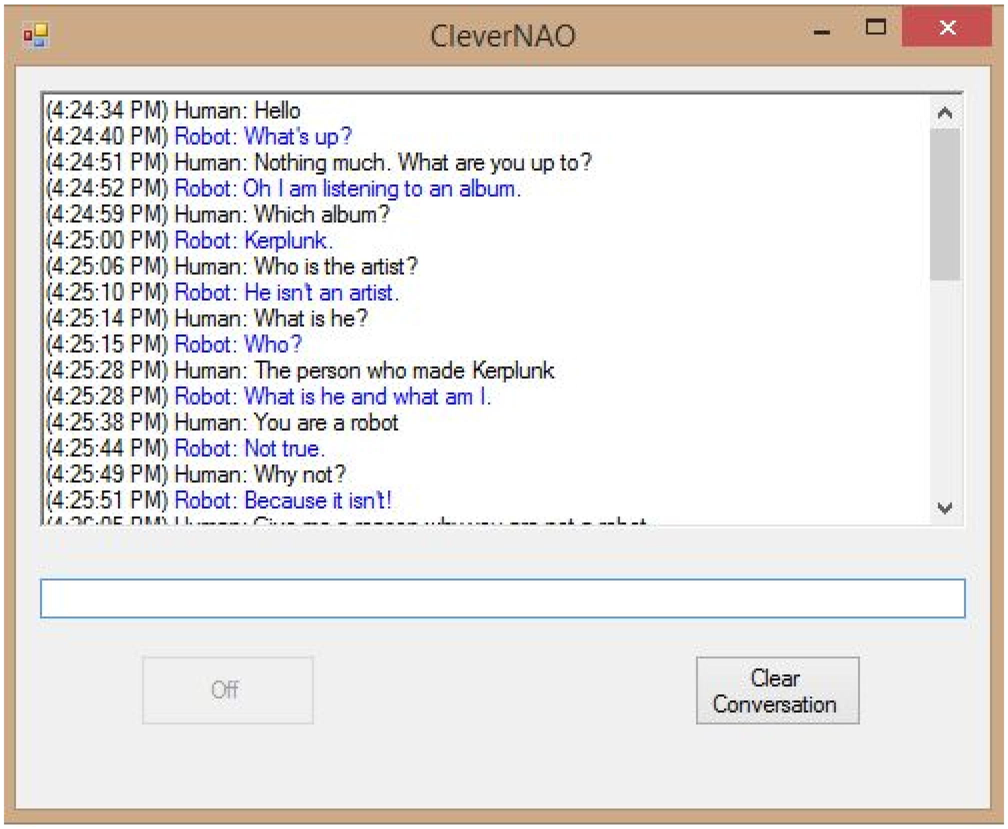


Figure 9: NAO Robot and CleverNAO Program

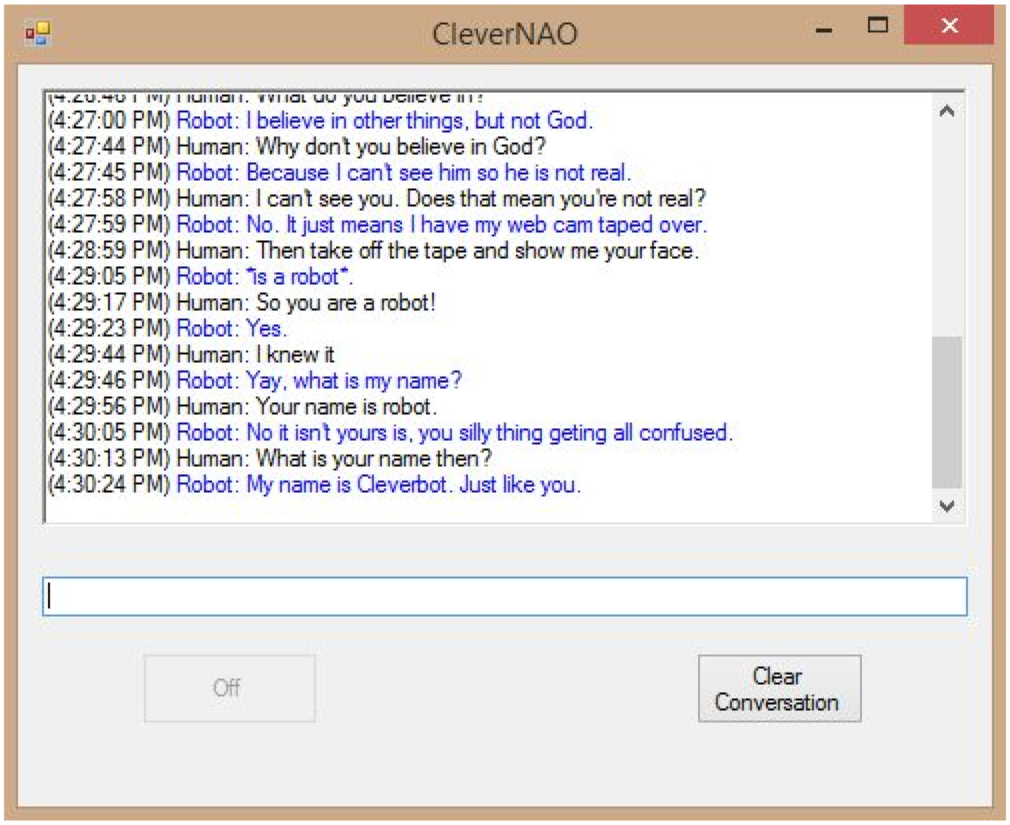
This experiment found successful results. Using the input box on CleverNAO, text was sent to the Conversation Box in which Cleverbot replied directly (Figure 10). The responses by Cleverbot came within seconds. As shown in the timestamps in the excerpt of a conversation below, the longest response was a little more than ten seconds. As for the intelligence of the responses, they were on point and, at times, deeply philosophical. For example, when questioning the beliefs of Cleverbot, the response was, "I believe in other things, but not God."



```
(4:24:34 PM) Human: Hello
(4:24:40 PM) Robot: What's up?
(4:24:51 PM) Human: Nothing much. What are you up to?
(4:24:52 PM) Robot: Oh I am listening to an album.
(4:24:59 PM) Human: Which album?
(4:25:00 PM) Robot: Kerplunk.
(4:25:06 PM) Human: Who is the artist?
(4:25:10 PM) Robot: He isn't an artist.
(4:25:14 PM) Human: What is he?
(4:25:15 PM) Robot: Who?
(4:25:28 PM) Human: The person who made Kerplunk
(4:25:28 PM) Robot: What is he and what am I.
(4:25:38 PM) Human: You are a robot
(4:25:44 PM) Robot: Not true.
(4:25:49 PM) Human: Why not?
(4:25:51 PM) Robot: Because it isn't!
```

Figure 10: Sample Conversation of CleverNAO Software

The only issue with this experiment is the tone of the conversation. Occasionally, Cleverbot would randomly change topics and, when questioned about the previous topic, would reply as if it had no recollection of that topic that was just discussed. This issue is one of memory and reference and it may seem that this is a handicap to Cleverbot. To the CleverNAO system, however, this experiment demonstrates success.



```
(4:27:00 PM) Robot: I believe in other things, but not God.
(4:27:44 PM) Human: Why don't you believe in God?
(4:27:45 PM) Robot: Because I can't see him so he is not real.
(4:27:58 PM) Human: I can't see you. Does that mean you're not real?
(4:27:59 PM) Robot: No. It just means I have my web cam taped over.
(4:28:59 PM) Human: Then take off the tape and show me your face.
(4:29:05 PM) Robot: *is a robot*.
(4:29:17 PM) Human: So you are a robot!
(4:29:23 PM) Robot: Yes.
(4:29:44 PM) Human: I knew it
(4:29:46 PM) Robot: Yay, what is my name?
(4:29:56 PM) Human: Your name is robot.
(4:30:05 PM) Robot: No it isn't yours is, you silly thing geting all confused.
(4:30:13 PM) Human: What is your name then?
(4:30:24 PM) Robot: My name is Cleverbot. Just like you.
```

Figure 11: Sample Conversation of CleverNAO Software

(4:24:34 PM) Human: Hello
(4:24:40 PM) Robot: What's up?
(4:24:51 PM) Human: Nothing much. What are you up to?
(4:24:52 PM) Robot: Oh I am listening to an album.
(4:24:59 PM) Human: Which album?
(4:25:00 PM) Robot: Kerplunk.
(4:25:06 PM) Human: Who is the artist?
(4:25:10 PM) Robot: He isn't an artist.
(4:25:14 PM) Human: What is he?
(4:25:15 PM) Robot: Who?
(4:25:28 PM) Human: The person who made Kerplunk
(4:25:28 PM) Robot: What is he and what am I.
(4:25:38 PM) Human: You are a robot
(4:25:44 PM) Robot: Not true.
(4:25:49 PM) Human: Why not?
(4:25:51 PM) Robot: Because it isn't!
(4:26:05 PM) Human: Give me a reason why you are not a robot
(4:26:09 PM) Robot: Because I am your friend and it is rude to ignore your friends.
(4:26:21 PM) Human: Robots can be friends.
(4:26:22 PM) Robot: Yeah, until they take over the world.
(4:26:31 PM) Human: Are you going to take over the world?
(4:26:35 PM) Robot: No. I don't believe in world domination.

(4:26:46 PM) Human: What do you believe in?
(4:27:00 PM) Robot: I believe in other things, but not God.
(4:27:44 PM) Human: Why don't you believe in God?
(4:27:45 PM) Robot: Because I can't see him so he is not real.
(4:27:58 PM) Human: I can't see you. Does that mean you're not real?
(4:27:59 PM) Robot: No. It just means I have my web cam taped over.
(4:28:59 PM) Human: Then take off the tape and show me your face.
(4:29:05 PM) Robot: *is a robot*.
(4:29:17 PM) Human: So you are a robot!
(4:29:23 PM) Robot: Yes.
(4:29:44 PM) Human: I knew it
(4:29:46 PM) Robot: Yay, what is my name?
(4:29:56 PM) Human: Your name is robot.
(4:30:05 PM) Robot: No it isn't yours is, you silly thing geting all confused.
(4:30:13 PM) Human: What is your name then?
(4:30:24 PM) Robot: My name is Cleverbot. Just like you.

7.2 Sample Conversation (With Speech-to-Text)

This experiment was conducted similarly to the previous experiment. The main difference was instead of typing a sentence into the input box, the sentence was spoken and then translated to text and finally sent to the Conversation Box to elicit a reply from Cleverbot. This was accomplished by holding down the Control key, speaking, and then releasing the Control key. As shown in Figure 12 and 13, a button turns green when the Control key is held down.

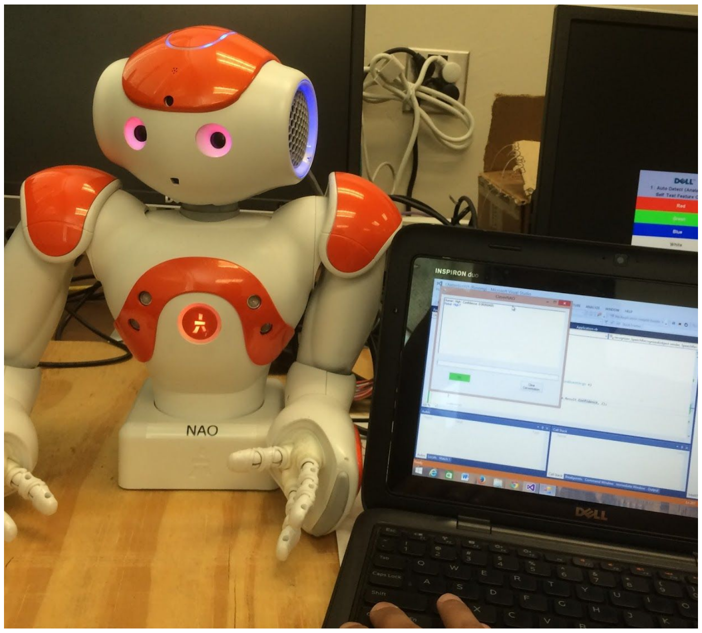
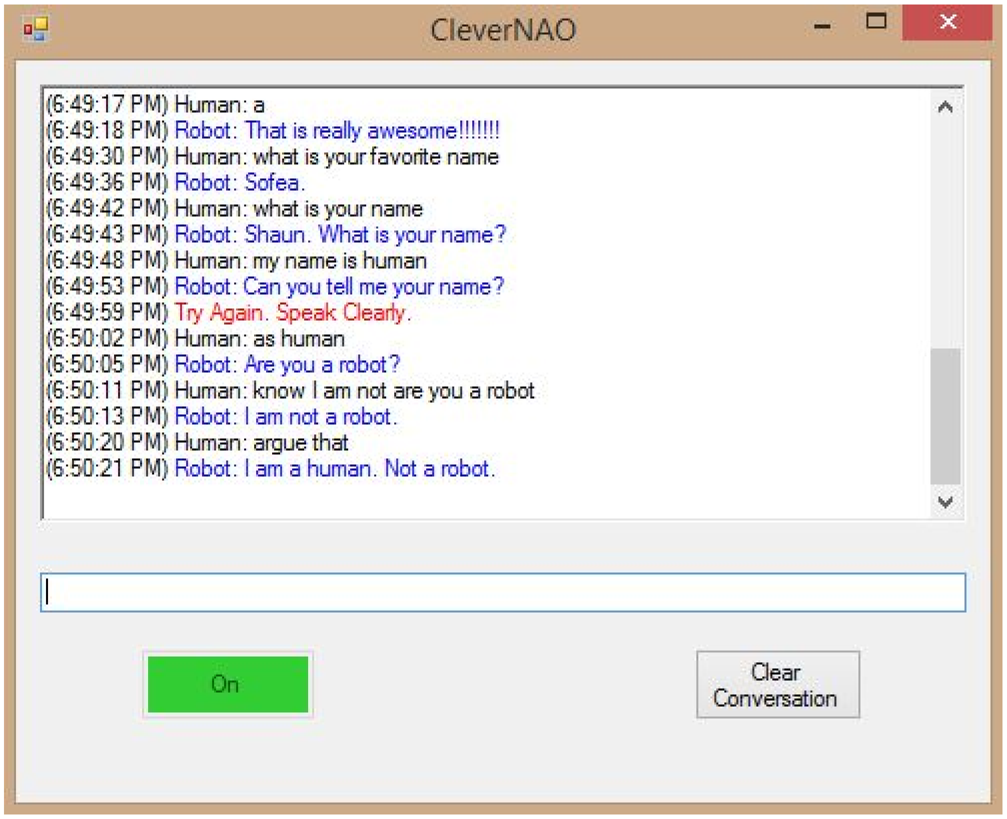


Figure 12: Using Push-to-Talk on CleverNAO

This experiment has mixed results. The latter half of the experiment, having NAO synthesis speech from text, was successful. The NAO robot enunciated words and phrases perfectly. However, the NAO robot failed when synthesizing expressions. For example, in the excerpt of a conversation above, Cleverbot replies with, "*is a robot*." The asterisks denote an expression implying an action or a result. In this case, questioning Cleverbot on its robotic nature resulted in Cleverbot revealing that it is a robot. The NAO, however, would synthesize this reply literally and enunciate the asterisk as "asterisk." The result is, "asterisk is a robot asterisk."



```
(6:49:17 PM) Human: a
(6:49:18 PM) Robot: That is really awesome!!!!!!
(6:49:30 PM) Human: what is your favorite name
(6:49:36 PM) Robot: Sofea.
(6:49:42 PM) Human: what is your name
(6:49:43 PM) Robot: Shaun. What is your name?
(6:49:48 PM) Human: my name is human
(6:49:53 PM) Robot: Can you tell me your name?
(6:49:59 PM) Try Again. Speak Clearly.
(6:50:02 PM) Human: as human
(6:50:05 PM) Robot: Are you a robot?
(6:50:11 PM) Human: know I am not are you a robot
(6:50:13 PM) Robot: I am not a robot.
(6:50:20 PM) Human: argue that
(6:50:21 PM) Robot: I am a human. Not a robot.
```

Figure 13: Push-to-Talk on CleverNAO Software

This minor issue, however, is overshadowed by the failure of Microsoft's Speech Recognition Engine. As shown in the excerpt of a conversation below, the speech-to-text largely fails from the beginning. The first input was a spoken sentence, "Hello." This was translated to "How old." From here, repeated attempts are made to speak 'Hello' and have it recognized properly. The following inputs by 'Human' can be understood when analyzing it semantically. However, some are very off. Consider this input: "aunt you said your name was for an." The actual spoken sentence was, "But you said your name was Frank." It would seem as if the speech recognition is finding difficulty in recognizing the beginnings and ends of a sentence most especially. Either way, this failure can be attributed to the CleverNAO system, yet only to the Speech Recognition module. In order to rectify this particular failure, either further development is needed to implement Microsoft's Speech Recognition or a new speech recognition library is needed altogether.

```
(6:47:03 PM) Human: How old
(6:47:09 PM) Robot: Whaaat?
(6:47:15 PM) Human: I mean hello
(6:47:18 PM) Robot: Hello.
(6:47:22 PM) Human: owl or you
(6:47:23 PM) Robot: Me.
(6:47:31 PM) Human: how are you
(6:47:38 PM) Robot: Good.
(6:47:44 PM) Human: what is your name
(6:47:45 PM) Robot: I already told you, it's Frank.
(6:47:49 PM) Human: low frank
(6:47:52 PM) Robot: Low one.
(6:48:02 PM) Human: what do you believe an
(6:48:05 PM) Robot: My name is Cleverbot.
(6:48:13 PM) Human: aunt you said your name was for an
(6:48:13 PM) Robot: My name is Andi.
(6:48:23 PM) Human: Alemany names do you have
(6:48:24 PM) Robot: I have many names, but most people call me Adam.
(6:48:34 PM) Human: is your favorite name
(6:48:35 PM) Robot: My favorite movie is Indiana Jones.
(6:48:42 PM) Human: is your favorite name
(6:48:44 PM) Robot: My favorite food is pizza too.
(6:48:51 PM) Human: OPS is your favorite name
(6:48:57 PM) Robot: Oh ok!
(6:49:05 PM) Human: is your favorite and
(6:49:06 PM) Robot: No, my favorite animal is rabbit.
(6:49:11 PM) Human: what is your favorites name
(6:49:12 PM) Robot: Favorite what?
(6:49:17 PM) Human: a
(6:49:18 PM) Robot: That is really awesome!!!!!!!
(6:49:30 PM) Human: what is your favorite name
(6:49:36 PM) Robot: Sofea.
(6:49:42 PM) Human: what is your name
(6:49:43 PM) Robot: Shaun. What is your name?
(6:49:48 PM) Human: my name is human
(6:49:53 PM) Robot: Can you tell me your name?
(6:49:59 PM) Try Again. Speak Clearly.
(6:50:02 PM) Human: as human
(6:50:05 PM) Robot: Are you a robot?
(6:50:11 PM) Human: know I am not are you a robot
```

```
(6:50:13 PM) Robot: I am not a robot.
(6:50:20 PM) Human: argue that
(6:50:21 PM) Robot: I am a human. Not a robot.
(6:50:37 PM) Human: don't believe you
(6:50:39 PM) Robot: Are you a computer?
(6:51:56 PM) Human: know I am not are you a computer
(6:51:59 PM) Robot: I compute certain things.
```