Using Chatbots to Teach Socially Intelligent Computing Principles in Introductory Computer Science Courses

Alan Shaw, Ph.D.
CS Department
Kennesaw State University
Kennesaw, GA, USA
ashaw8@kennesaw.edu

***Abstract*—Teaching "socially intelligent computing" concepts in introductory computer science courses is possible through curricular projects that involve students developing their own collaborative chatbots. Chatbots are programs that interact with users in a conversational context. Introductory CS students can write these types of programs using a framework of Peer-to-Peer (P2P) libraries presented in this poster paper. When the students build chatbots that collaborate with one another, the students are introduced to a branch of machine intelligence called socially intelligent computing.**

Keywords-CS 1, CS 2, Pedagogy, Socially Intelligent Computing, Collaborative Programming

I. INTRODUCTION

The concept of artificial intelligence in computers has been around for generations, and it was started as an academic discipline as far back as 1956. [1] However, there is a somewhat recent offshoot of the basic idea of machine intelligence called "socially intelligent computing" that has been gaining interest, as evidenced by the following statement from a recently established NSF program:

> "The Social-Computational Systems (SoCS) program seeks to reveal new understanding about the properties that systems of people and computers together possess, and to develop theoretical and practical understandings of the purposeful design of systems to facilitate socially intelligent computing. By better characterizing, understanding, and eventually designing for desired behaviors arising from computationally mediated groups of people at all scales, new forms of knowledge creation, new models of computation, new forms of culture, and new types of interaction will result." [2]

The statement by the NSF goes on to define "socially intelligent computing" as systems with "emergent behaviors" that arise "from human-computer partnerships that range in scale from a single person and computer to an Internet-scale array of machines and people."

Developing CS 1 and CS 2 curricula that involves teaching students how to write simple social computing applications can serve as an introduction into the emerging study of socially intelligent computing systems and help students to explore its relevance. Along these lines, we have developed a peer-to-peer (P2P) framework that can be used by students to develop networked applications without the normal degree of difficulty involved. This framework also makes it possible for CS 1 and CS 2 students to develop socially intelligent computing applications.

II. THE PEER-TO-PEER FRAMEWORK

Using a centralized P2P server, the client-server framework connects one programmer's program to another program written by a peer using the P2PSession class:

```
P2PSession p2p = new P2PSession("Student 1");
p2p.talkTo("Student 2");
p2p.sendString("What is your nick name? ");
String nickname = p2p.receiveString();
p2p.sendString("Hello " + nickname);
```

In the above lines, "Student 1" and "Student 2" would be usernames chosen by two students developing separate programs that will communicate together over a P2P network. When the P2PSession constructor is run, it registers the username of the first student with the server. On the second line, the P2PSession object sends the server the username of the other student with whom the first student will be communicating. After that, neither student has to worry about how the server maintains a connection between their programs. The programs just send and receive communications between each other in real time.

Using this framework students can create chat applications in which the two programs enter a loop that send messages and responses back and forth until one of them types in some sentinel value to end the loop like so:

```
String outputLine, inputLine;
Scanner keyboard = new Scanner(System.in);
```

978-0-7695-4654-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ITNG.2012.70

```
P2PSession p2p = new P2PSession("Student 1");
p2p.talkTo("Student 2");
do  {
   System.out.print("Message to send: ");
   outputLine = keyboard.nextLine();
   p2p.sendString(outputLine);
   if (outputLine.equalsIgnoreCase("quit"))
      break;
   inputLine = p2p.receiveString();
   System.out.print("Message received: " + inputLine);
} while (! inputLine.equalsIgnoreCase("quit"));
```

To create more sophisticated chat programs, students are given a ChatBot library for processing inputs and generating replies. With the framework, a student's program provides a list of keywords and phrases to search for in order to trigger a list of corresponding sentences as replies whenever a keyword, or a group of keywords or phrases is matched. In this way, students can create simple chatbot programs with which other user's can interact:

Sample Chatbot Java Program Source Code:

```
public class RobotChat
{
  public static void main(String[] args)
  {
     ChatBot chat = new ChatBot("The Chatbot");
     chat.talkTo("The Student");
     chat.welcomeMessage("Hello! Can you guess my favorite movie?");
     chat.keyResponseAny("Nope.","western","comedy","love");
     chat.keyResponseAny("Yep.", "neo","morpheus","trinity","agent");
     chat.keyResponseAll("Yep, that's the genre.","science","fiction");
     chat.keyResponseAll("You got it!  Congratulations!","matrix");
     chat.noMatchMessage("Oh come on, you can do better than that!");
     chat.noMatchMessage("You're getting closer!");
     chat.noMatchMessage("That's a strange thing to say?");
     chat.startConversation();
  }
}
```

Sample Output Produced:

```
-> The Chatbot said: Hello! Can you guess my favorite movie?
-> The Student said: Is it a western?
-> The Chatbot said: Nope.
-> The Student said: Is it a comedy?
-> The Chatbot said: Nope.
-> The Student said: Is it a science fiction movie?
-> The Chatbot said: Yep, that's the genre.
-> The Student said: Is it Star Trek?
-> The Chatbot said: Oh come on, you can do better than that!
-> The Student said: 2001?
-> The Chatbot said: You're getting closer!
-> The Student said: Blade Runner
-> The Chatbot said: That's a strange thing to say?
-> The Student said: Is Morpheus in it?
-> The Chatbot said: Yep.
-> The Student said: Is it the Matrix?
-> The Chatbot said: You got it!  Congratulations!
```

In the above RobotChat code, after establishing the ChatBot connection, a welcome message is set, then, on the next two lines the keyResponseAny() method is given a reply as the first argument, followed by any number of keyword arguments. The reply is sent as output whenever any of the keywords from the latter arguments are given as input. Then two keyResponseAll() methods are also given a reply followed by keyword arguments. The reply is sent whenever all of the keywords given are in the input. The three noMatchMessage() methods get string arguments which are sent as output consecutively whenever the user's input doesn't match any of the keyword rules.

III. SOCIALLY INTELLIGENT CHATBOTS

The above code is an example of a chatbot that a single student might create. Since there is no way for one student to predict all of the possible things that a user might type in, it helps for the student to have access to additional matching rules. This is possible because the P2P framework allows chatbots written by different users to share their set of rules. When a chatbot has no rule to matches an input, it can send the input to other chatbots to try and find a match from them without resorting to using a "No Match Messages." This system of interconnected chatbots is an example of a socially intelligent system because the responses result from the collaborative intelligence the chatbots. As such, the "emergent behaviors" that arise might be somewhat unpredictable.

Students building these systems can reflect on how distributed intelligence is different from intelligence that is centered on a single source. And this distinction, at least in part, is at the heart of what make socially intelligent computing an interesting branch of artificial intelligence.

IV. CONCLUSION

Chatbot projects in introductory CS classes can explore issues concerning socially intelligent computing, and students engaged in this work may gain a better sense of some of the current and future issues involving social computing applications. In the future, we will seek to develop other curricular units that use our collaborative P2P framework to allow students to create different socially intelligent computing systems that explore new types emergent and intelligent system behaviors.

REFERENCES

[1] McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd.

[2] National Science Foundation, SoCS Program. http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503406