Name: Anika Jallipalli  
Date: 2/8/2020

Magpie Chatbot Lab

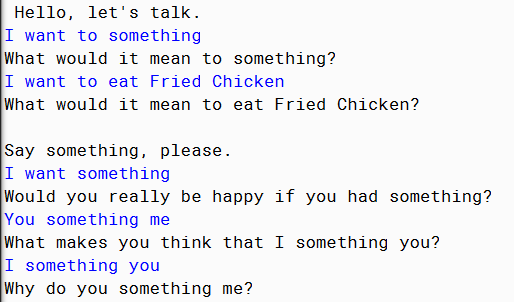
**Directions**: Make note of your responses to the following questions as you work through activities 4 and 5 of the AP Computer Science Lab Student Guide: Magpie.

# Activity 4

1. In the Exploration section, how does the Magpie chatbot respond to:

|  |  |
| --- | --- |
| I want to build a robot. | What would I mean to build a robot? |
| I want to understand French. | What would it mean to understand French? |
| Do you like me? | What makes you think I like you? |
| You confuse me. | What makes you think that I confuse you? |

1. After altering the code, test the code by chatting with Magpie. Test statements in the form of "I want to *something*", "I want *something*", "you *something* me", and "I *something* you". Paste your conversation below.



1. When altering the code, why did you need to be careful about where you place the check for "I want *something*" statements? Use the samples to determine your answer.

It will check the substring before and after the string to make sure that the keyword is actually included in the string.

# Activity 5

1. Imagine you were to interview an end user and ask them to compare the version of the program before the array was added to getRandomResponse and the version after. What might they say?

They would notice the variety of responses to the different orders the key word can be. They will probably enjoy the modified version more because of the more extensive responses it can provide.

1. Now interview the programmer for the code. How might he/she compare and contrast the two programs? Are the end user’s and programmer’s responses similar? Why?

Their responses would be similar since the program had been improved to give a more extensive amount of responses. This makes the program more useful and fun to talk to.

**MAGPIERUNNER4**

/\*\*

\* A simple class to run the Magpie class.

\*

\* Code adapted from work by Laurie White for the College Board.

\*

\* @author Anika Jallipalli

\* @version 2/8/2020

\*/

import java.util.Scanner;

public class MagpieRunner4

{

/\*\*

\* Create a Magpie, give it user input, and print its replies.

\*/

public static void main(String[] args)

{

Magpie4 maggie = new Magpie4();

System.out.println (maggie.getGreeting());

Scanner in = new Scanner (System.in);

String statement = in.nextLine();

while (!statement.equals("Bye"))

{

System.out.println (maggie.getResponse(statement));

statement = in.nextLine();

}

}

}

**MAGPIE4**

/\*\*

\* A program to carry on conversations with a human user.

\* This version:

\*<ul><li>

\* Uses advanced search for keywords

\*</li><li>

\* Will transform statements as well as react to keywords

\*</li></ul>

\* @author Anika Jallipalli

\* @version 2/8/2020

\*

\*/

public class Magpie4 {

/\*\*

\* Get a default greeting

\* @return a greeting

\*/

public String getGreeting() {

return "Hello, let's talk.";

}

/\*\*

\* Gives a response to a user statement

\*

\* @param statement

\* the user statement

\* @return a response based on the rules given

\*/

public String getResponse(String statement) {

String response = "";

// Look for a two word (you <something> me) pattern

int youPsn = findKeyword(statement, "you", 0);

int iPsn = findKeyword(statement, "I", 0);

if (statement.length() == 0) {

response = "Say something, please.";

} else if (findKeyword(statement, "no") >= 0) {

response = "Why so negative?";

} else if (findKeyword(statement, "mother") >= 0

|| findKeyword(statement, "father") >= 0

|| findKeyword(statement, "sister") >= 0

|| findKeyword(statement, "brother") >= 0) {

response = "Tell me more about your family.";

}

// Responses which require transformations

else if (youPsn >= 0 && findKeyword(statement, "me", youPsn) >= 0) {

response = transformYouMeStatement(statement);

} else if (iPsn >= 0 && findKeyword(statement, "you", iPsn) >= 0) {

response = transformIYouStatement(statement);

} else if (findKeyword(statement, "I want to", 0) >= 0) {

response = transformIWantToStatement(statement);

} else if (findKeyword(statement, "I want", 0) >= 0) {

response = transformIWantStatement(statement);

} else {

response = getRandomResponse();

}

return response;

}

/\*\*

\* Take a statement with "I want to <something>." and transform it into

\* "What would it mean to <something>?"

\* @param statement the user statement, assumed to contain "I want to"

\* @return the transformed statement

\*/

private String transformIWantToStatement(String statement) {

// Remove the final period, if there is one

statement = statement.trim();

String lastChar = statement.substring(statement

.length() - 1);

if (lastChar.equals(".") || lastChar.equals("!"))

{

statement = statement.substring(0, statement

.length() - 1);

}

int psn = findKeyword (statement, "I want to", 0);

String restOfStatement = statement.substring(psn + 9).trim();

return "What would it mean to " + restOfStatement + "?";

}

/\*\*

\* Take a statement with "I want <something>." and transform it into

\* "Would you really be happy if you had <something>?"

\* @param statement the user statement, assumed to contain "I want <statement>"

\* @return the transformed statement

\*/

private String transformIWantStatement(String statement) {

// Remove the final period, if there is one

statement = statement.trim();

String lastChar = statement.substring(statement.length() - 1);

if (lastChar.equals(".") || lastChar.equals("!"))

{

statement = statement.substring(0, statement.length() - 1);

}

int psn = findKeyword (statement, "I want", 0);

String restOfStatement = statement.substring(psn + 6).trim();

return "Would you really be happy if you had " + restOfStatement + "?";

}

/\*\*

\* Take a statement with "you <something> me" and transform it into

\* "What makes you think that I <something> you?"

\* @param statement the user statement, assumed to contain "you" followed by "me"

\* @return the transformed statement

\*/

private String transformYouMeStatement(String statement) {

// Remove the final period, if there is one

statement = statement.trim();

String lastChar = statement.substring(statement.length() - 1);

if (lastChar.equals("."))

{

statement = statement.substring(0, statement.length() - 1);

}

int psnOfYou = findKeyword (statement, "you", 0);

int psnOfMe = findKeyword (statement, "me", psnOfYou + 3);

String restOfStatement = statement.substring(psnOfYou + 3, psnOfMe).trim();

return "What makes you think that I " + restOfStatement + " you?";

}

/\*\*

\* Take a statement with "you <something> me" and transform it into

\* "What makes you think that I <something> you?"

\* @param statement the user statement, assumed to contain "you" followed by "me"

\* @return the transformed statement

\*/

private String transformIYouStatement(String statement) {

// Remove the final period, if there is one

statement = statement.trim();

String lastChar = statement.substring(statement.length() - 1);

if (lastChar.equals("."))

{

statement = statement.substring(0, statement.length() - 1);

}

int psnOfI = findKeyword (statement, "I", 0);

int psnOfYou = findKeyword (statement, "you", psnOfI + 1);

String restOfStatement = statement.substring(psnOfI + 1, psnOfYou).trim();

return "Why do you " + restOfStatement + " me?";

}

/\*\*

\* Search for one word in phrase. The search is not case sensitive.

\* This method will check that the given goal is not a substring of a longer string

\* (so, for example, "I know" does not contain "no").

\* @param statement the string to search

\* @param goal the string to search for

\* @param startPos the character of the string to begin the search at

\* @return the index of the first occurrence of goal in statement or -1 if it's not found

\*/

private int findKeyword(String statement, String goal, int startPos) {

String phrase = statement.trim();

// The only change to incorporate the startPos is in the line below

int psn = phrase.toLowerCase().indexOf(goal.toLowerCase(), startPos);

// Refinement--make sure the goal isn't part of a word

while (psn >= 0)

{

// Find the string of length 1 before and after the word

String before = " ", after = " ";

if (psn > 0)

{

before = phrase.substring (psn - 1, psn).toLowerCase();

}

if (psn + goal.length() < phrase.length())

{

after = phrase.substring(psn + goal.length(), psn + goal.length() + 1).toLowerCase();

}

// If before and after aren't letters, we've found the word

if (((before.compareTo ("a") < 0 ) || (before.compareTo("z") > 0)) // before is not a letter

&& ((after.compareTo ("a") < 0 ) || (after.compareTo("z") > 0)))

{

return psn;

}

// The last position didn't work, so let's find the next, if there is one.

psn = phrase.indexOf(goal.toLowerCase(), psn + 1);

}

return -1;

}

/\*\*

\* Search for one word in phrase. The search is not case sensitive.

\* This method will check that the given goal is not a substring of a longer string

\* (so, for example, "I know" does not contain "no"). The search begins at the beginning of the string.

\* @param statement the string to search

\* @param goal the string to search for

\* @return the index of the first occurrence of goal in statement or -1 if it's not found

\*/

private int findKeyword(String statement, String goal) {

return findKeyword (statement, goal, 0);

}

/\*\*

\* Pick a default response to use if nothing else fits.

\* @return a non-committal string

\*/

private String getRandomResponse() {

final int NUMBER\_OF\_RESPONSES = 4;

double r = Math.random();

int whichResponse = (int) (r \* NUMBER\_OF\_RESPONSES);

String response = "";

if (whichResponse == 0) {

response = "Interesting, tell me more.";

} else if (whichResponse == 1) {

response = "Hmmm.";

} else if (whichResponse == 2) {

response = "Do you really think so?";

} else if (whichResponse == 3) {

response = "You don't say.";

}

return response;

}

}