HOW TO PROGRAM A CHATBOT – AN INTRODUCTORY PROJECT AND STUDENT PERCEPTIONS

Ju Long*  Texas State University, San Marcos, United States  julong@txstate.edu
Michael Juntao Yuan  Ringful Health, Austin, United States  michael@ringful.com
Hsun-Ming Lee  Texas State University, San Marcos, United States  samlee@txstate.edu

* Corresponding author

ABSTRACT

Aim/Purpose  One of the most fascinating developments in computer user interfaces in recent years is the rise of “chatbots”. Yet extent information system (IS) curriculum lacks teaching resources on chatbots programming.

Background  To better prepare students for this new technological development and to enhance the IS curriculum, we introduce a project that teaches students how to program simple chatbots, including a transactional chatbot and a conversational chatbot.

Methodology  We demonstrated a project that teaches students how to program two types of simple chatbots: a transactional chatbot and a conversational chatbot. We also conducted a survey to examine students’ perceptions on their learning experience.

Findings  Our survey on students’ perception of the project finds that learning chatbots is deemed very useful because chatbot programming projects have enabled the students to understand the subject better. We also found that social influence has positively motivated the students to learn chatbot programming. Though most of the students have no prior experiences programming chatbots, their self-efficacy towards chatbot programming remained high after working through the programming project. Despite the difficult tasks, over 71% of respondents agree to various degrees that chatbot programming is fun. Though most students agree that chatbot programming is not easy to learn, more than 70% of respondents indicated that they will use or learn chatbots in the near future. The overwhelmingly positive responses are impressive given that this is the first time for the students to program and learn chatbots.

Accepting Editor: Eli Cohen  |  Received: April 16, 2018  |  Revised: July 15, October 2, December 31, 2018, March 12, March 26, 2019  |  Accepted: March 29, 2019.

(CC BY-NC 4.0) This article is licensed to you under a Creative Commons Attribution-NonCommercial 4.0 International License. When you copy and redistribute this paper in full or in part, you need to provide proper attribution to it to ensure that others can later locate this work (and to ensure that others do not accuse you of plagiarism). You may (and we encourage you to) adapt, remix, transform, and build upon the material for any non-commercial purposes. This license does not permit you to use this material for commercial purposes.
How to Program a Chatbot

Recommendations for Practitioners
In this article, we introduced a step by step project on teaching chatbot programming in an information systems class. Following the project instructions, students can get their first intelligent chatbots up and running in a few hours using Slack. This article describes the project in detail as well as students’ perceptions.

Recommendations for Researchers
We used UTAUT model to measure students’ perception of the projects. This study could be of value to researchers studying students’ technology learning and adoption behaviors.

Impact on Society
To our best knowledge, pedagogical resources that teach IS students how to program chatbots, especially the introductory level materials, are limited. We hope this teaching case could be of value for IS educators when introducing IS students to the wonderful field of chatbot programming.

Future Research
For future work, we plan to expand the teaching resources to cover more advanced chatbot programming projects, such as on how to make chatbot more human-like.

Keywords
information system education, chatbot, artificial intelligence, student perceptions

INTRODUCTION

One of the most fascinating developments in computer user interfaces in recent years is the rise of “chatbots”. Some people hail it as the “new command line” (Roach, 2016; Zyane, 2017), but it is much more than that. Chatbots are Artificial Intelligence “robots” used in human users’ preferred environments, mostly in messaging applications. Chatbots converse with users in a natural language, understand what users want, and do the humans’ bidding through a large network of connected services with natural and pervasive user interfaces. With the rising popularity of messaging apps, users are increasingly interested in this mode of communication—the next step in the continuous user interface evolution from text command line to Graphical User Interface to web apps to mobile apps and then to “bots”.

One of the first chatbots, ELIZA (Weizenbaum, 1966), was developed 50 years ago at the Massachusetts Institute of Technology. It simulated a Rogerian psychotherapist, someone who just repeats the human user’s words back to the human. In recent years, with the development of smartphone technology, smartphone-based chatbots have gained wide interest in the industry with high profile products such as the Apple Siri and Amazon Alexa as examples. Popular messaging platforms, such as Facebook Messenger and Skype, announced developer programs to support chatbot applications. At the same time, smaller messaging platforms, such as Slack and Telegram, launched “bot stores” and investment funds to attract developers.

There are already many innovative uses of bots. For example, Github’s intelligent bot HuBot performs tasks from running software builds on demand, deploying software to production websites, scheduling and running meetings, and even translating between human languages. All these tasks were once done by highly paid human employees (https://hubot.github.com/, accessed December 2017).

Clearly, as chatbots become more prevalent in business applications, chatbot programming will become more a desirable skill for IS students. However, since this field is still quite new and developing fast, there are relatively little pedagogical resource on teaching IS students how to program chatbots. Most of the existing resources are reference type of books or magazine articles. An exception is some pioneering research done in education setting to facilitate student learning. Griol, Molina, and Sanchis De Miguel (2014) studied how to use a conversational agent to help children learn to appreciate their environment. Oliveira, Silva, and Silva (2016) examined the mechanisms of human lan-
language formation to simulate a more human-like conversational chatbot. However, despite this early research on chatbot, there is no step-by-step instructional material to introduce chatbot programming to beginning learners, especially students in the information systems fields. Given the rising popularity of chatbots and their increasingly extensive applications in the industry, the skills set to be able to program chatbot becomes more marketable and desirable. To better prepare students for this new technological development, to help students build more marketable skillsets, and to enhance the information system (IS) curriculum, we introduce a project that teaches undergraduate students major in computer information systems how to program a simple chatbot. Following the project instructions, students can get their first intelligent chatbot up and running in a few hours using Slack. This article describes the project in detail as well as students’ perceptions. To our best knowledge, pedagogical resources that teach IS students how to program chatbots, especially the introductory level materials, are limited. We hope this teaching case could be of value for IS educators when introducing IS students to the wonderful field of chatbot programming.

**LITERATURE REVIEW**

A chatbot can be defined as “an artificial construct that is designed to converse with human beings using natural language as input and output” (Brennan, 2006). Conversational agents, or chatbots, provide a natural language interface to their users (Kerly, 2007). Over the years, their design has become increasingly sophisticated and their use adopted in education (e.g., Jia, 2004) and commerce (e.g., Statista, 2017). More recently, smartphone-based chatbots have gained wide interest in the industry with the introduction of high profile products such as the Apple Siri and Amazon Alexa, as well as messaging platforms such as Facebook Messenger and Skype. For instance, More than 2 billion business-related messages are sent through Facebook Messenger chats according to Facebook IQ (2018). 67% of that study participants expressed expectations of having the possibility to get in touch with the sellers and service providers by means of chats.

According to 2017 worldwide Statista survey (Statista, 2017), 34% of respondents declared they would prefer to answer the questions from AI by means of a chatbot or a virtual assistant in regard to e-commerce. Healthcare (27%) and telecommunications (25%) were detected to be the second and the third correspondingly.

ELIZA (Weizenbaum, 1966) was developed as one of the first chatbots. ELIZA analyzed input sentences and created its response based on reassembly rules associated with a decomposition of the input. Dr. Richard Wallace (2009) developed A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) as a chatbot built using Artificial Intelligence Markup Language (AIML). The chatbot is based on categories containing a stimulus, or pattern, and a template for the response. Category patterns are matched to find the most appropriate response to a user input. Furthermore, AIML tags provide for consideration of context, conditional branching, and supervised learning to produce new responses. Overall, A.L.I.C.E. is a viable and experienced system.

There are scarce resources on teaching IS students how to program chatbots. The main reason for the lack of extent literature on teaching chatbot programming is probably because modern chatbot development is still a relatively nascent field. However, we do find that chatbots have been used in education to facilitate student learning. For instance, based on natural language processing, intelligent tutoring, text-to-speech generation, voice recognition, and virtual reality, a prototype of a chatbot was developed to serve as a language instructor assistant. The chatbot could simulate an intelligent dialogue capable of conducting a conversation, correct some grammar and spelling mistakes, as well as provide definitions and translations of words and phraseological units (Pietroszek, 2007). Another simulation system focuses on supplying a virtual chatting partner (chatbot), which can chat in English with the English learners anytime, anywhere. It generates communicative responses according to the user input, the dialogue context, the user’s and its own personality knowledge, common sense knowledge, and inference knowledge (Jia, 2009).
Besides assisting language learning, chatbots have been used as a tool for network management training. Dorothy, a network management-aware chatbot, was created to interact not only with the user, but also with the managed network via management protocols (Leonhardt, Tarouco, Vicari, Santos, & Silva, 2007). A multimodal conversational agent has also been developed to help children appreciate and protect their environment (Griol et al, 2014). Mechanisms of human language formation (semantic, syntactic, pragmatic, and morphologic analysis) were used to simulate a normal conversation in an almost human way, which could eventually replace a human being (Oliveira, et al., 2016). In another research, a community of students, teachers, and interested developers programmed the database of the educational chatbot with educational content, including questions and answers on school subject information and other digital content, to add data to an educational chatbot’s knowledge base (Bii & Too, 2016).

To extend the existing research on chatbot, it is valuable to examine ways to include teaching chatbot programming in the IS curriculum. Thus our project introduces a step-by-step project on how to program for chatbots. IS educators could use this project as an introduction to students interested in chatbot programming. Given the increasingly extensive development and utilization of chatbot applications in the IT industry, teaching students how to program chatbots could enhance students’ marketable skill sets. In the following section, we will describe the project we designed to teach undergraduate students in the IS field how to program chatbots.

**RESEARCH METHODOLOGY**

In this project, students are instructed to develop two types of chatbots – transactional bots and conversational bots – and our project is designed as a phased project. We divide the project into two phases because we focus on different types of chatbots in each phase: students first create a transactional bot (the Movie Info bot) and then a conversational bot (the Movie Line bot). The project uses Slack as the platform to build the chatbot on. We also suggested to students the free web hosting services (e.g., Heroku, IBM Bluemix, RedHat OpenShift) to host the PHP code.

Three class sessions (each session is 80 minutes) were devoted to explaining and programming the projects. These sessions were held in the computer lab. During each session, students spent about 40 minutes working in their groups and coding for the chatbot based on the codes and instructions given by the instructor. During the second 40 minutes, the instructor explained the codes and helped students debug. There is no student tutor available. Students learned from interactions with their peers and the instructor.

To gauge the students’ interests in programming for chatbots, and subsequently examine how to improve the teaching methodology and learning experience, we conducted a survey on students’ perceptions and we share the survey results in this section too.

**TWO TYPES OF INTELLIGENT BOTS**

First, it is crucial to understand that there are two types of intelligent bots. The first is transactional bots. These bots respond to user commands to complete a single task at a time. Of course, the transactional task could be quite complicated, such as “join this meeting and take notes.” The key here is that the user simply tells the bot what needs to be done. There is no “conversation” per se. The above-mentioned GitHub HuBot is a good example of transactional bots, as is Apple Siri. Example tasks for a transactional bot include report weather, look up movie times or ratings, build software, translate a term, schedule a reminder, schedule a meeting, etc.

The second kind of intelligent bots are conversational bots. These type of bots can carry a conversation with human users and figure out what the human users want before executing the task. Conversational bots are much harder to program than transactional bots because they need to keep track of the conversation context (i.e., when the user says “this is correct,” what does “this” refer to?) and...
detect the user’s sentiment. The conversational bots also need to derive a certain conclusion at the end of the conversation and act on it.

**Slack Chatbot Basics**

For this teaching project, we build our bots on Slack. Slack is a group messaging platform for work-related teams. As of September 2017, it had 6 million daily active users (Konrad, 2017), and it is growing fast. Slack is one of the pioneers in the current wave of chatbot renaissance. Slack provides one of the first “bot stores” in the industry so that teams can discover and install bots easily. As expected, most Slack bots are work and productivity related. We choose Slack to build our bots because of the following reasons. First and foremost, Slack provides a ready-to-use infrastructure to plug-in students’ bots. It already has a very user-friendly interface for a sophisticated messaging app. Second, Slack allows users to create an unlimited number of teams and messaging channels, making it easy to test students’ bots in private. Third, Slack is very popular, and many people are already familiar with it. Fourth, Slack is free for small teams such as student teams. Fifth, and finally, Slack provides compelling tools to help students make their bots available to other teams and users in Slack’s vast community.

**The Movie Info Bot: A Transactional Bot**

In Slack, there are many ways to build bots. The first part of the project is to create a transactional bot. For transactional bots, one approach is to take advantage of the slash commands. Slash commands consist of a slash followed by a command keyword. The objective is for students to build a custom command called /movie that can give the user movie information. Any user can invoke this command followed by a movie title, and it will give him or her brief information about this movie. For example, type in /movie Star Wars episode IV (Figure 1):

![Figure 1. Slash command example](image1)

The response should be as shown in Figure 2.

![Figure 2. Movie Info result](image2)
To create a bot for slash command /movie, students need to first create this command in Slack’s customization tool. It is under the “Custom integrations → Slash Commands” menu. Here, students need to specify an external web server to respond to the command (Figure 3).

Then, when users type:

/movie Star Wars episode IV

Slack sends an HTTP POST to the specified server. The POST parameter “text” will have the value “Star Wars episode IV”:

text=Star Wars episode IV

Slack then waits for the server to respond and displays whatever text the server returns. In our project, the server queries the OMDB API for the movie title. OMDB API returns movie information in a JSON object like this:

```json
{
    "Title":"Star Wars: Episode IV - A New Hope",
    "Year":"1977",
    "Rated":"PG",
    "Released":"25 May 1977",
    "Runtime":"121 min",
    "Genre":"Action, Adventure, Fantasy",
    "Director":"George Lucas",
    "Writer":"George Lucas",
    "Actors":"Mark Hamill, Harrison Ford, Carrie Fisher, Peter Cushing",
    "Plot":"Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, a wookiee and two droids to save the galaxy from the Empire's world-destroying battle-station, while also attempting to rescue Princess Leia from the evil Darth Vader.",
    "Language":"English",
    "Country":"USA",
    "Awards":"Won 6 Oscars. Another 38 wins & 27 nominations.",
    "Metascore":"92",
}
```
The server then extracts relevant information from the JSON and assembles and formats the HTTP response text like this:

`Star Wars: Episode IV — A New Hope’ 1977. Rating:8.7 IMDB: http://www.imdb.com/title/tt0076759/ Plot: Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, a wookiee and two droids to save the galaxy from the Empire’s world-destroying battle-station, while also attempting to rescue Princess Leia from the evil Darth Vader.

The full server code for the omdb_title.php file in PHP is below. We suggested students use many free web hosting services that can run PHP (e.g., Heroku, IBM Bluemix, RedHat OpenShift, etc.) to host the PHP code.

```php
<?php
$json_str = file_get_contents("http://www.omdbapi.com/?t=" . urlencode($_REQUEST["text"]) . "&y=&plot=short&r=json");
$json = json_decode($json_str, true);
?>
```

But, in the case of getting movie information, having to enter a full title of a movie and getting back a short description left much to be desired. Therefore, the next phase of the project is to create a conversational bot.

**The Movie Line Bot: A Conversational Bot**

In this part of the project, students are instructed to build a conversational bot that can interact with the user to discover exactly what movie information the user wants.

In Slack, a good place to build a conversational bot is a channel. A channel is a type of “group chat” in Slack. Any user can join a channel and start chatting. All the messages in the channel are shown to every user in the channel. However, besides human users, the channel itself can also respond to messages, making it a “bot” in some circumstances. We set up a channel called #movie_line. The screen shots that follow (Figure 4 to Figure 8) show the sequence of commands to drill from an ambiguous title search to show times. Please note that in Figure 4, we could not show all the options in the screenshot due to space limitations, and Figures 6 and 7 show the continuous flow of the commands.
How to Program a Chatbot

Figure 4. Search by movie title example

The chatbot has found the following movies matching your search. Please respond with a number to learn more about a movie.
1 Star Wars: Episode IV - A New Hope(1977)
2 Star Wars: Episode V - The Empire Strikes Back(1980)
3 Star Wars: Episode VI - Return of the Jedi(1983)
4 Star Wars: Episode I - The Phantom Menace(1999)
5 Star Wars: Episode III - Revenge of the Sith(2005)
8 Star Wars: The Clone Wars(2008)
10 Star Wars: Clone Wars(2003–2005)

Figure 5. Multiple results returned
Figure 6. Drill down to one result.

Figure 7. Search by showtime
In Slack, you could add “Custom integration → Outgoing WebHooks” to any channel. Here is how to specify another PHP script on the server to receive messages from this channel (Figure 9 and Figure 10).
When anyone posts a message in the channel, Slack sends an HTTP POST to the server. The POST contains the message in the “text” parameter, as well as other parameters describing the sender of the message:

```
token=OpDFurkJD0XMKVXtPinfeHV1
team_id=T00000
channel_id=C00000000
channel_name=test
timestamp=1355517523.000005
user_id=U0000000000
user_name=juntao
text=Star Wars
```

When the server responds, it needs to send back a JSON text to describe what the bot “says.” Compared with the plain text response from the previous Slash command-based transactional bot, this JSON response allows for structure and rich formatting of the response. For instance, the simple JSON response below indicates that the text should be formatted using a modified markdown convention.

```
{
  "text": "*bold* `code` _italic_",
  "mrkdwn": true
}
```

Now, the students are instructed to write the server PHP script. This script is more complicated than the previous transactional bot since it needs to keep track of conversation state (i.e., is the user asking to search a title or responding to one of the choices presented by the bot?) In this project, for the sake of simplicity, we create a cache file for each user session. The cache file is named after the token submitted by Slack in the HTTP POST to uniquely identify the conversation channel.

The overall flow of the PHP script is as follows:

First, the script checks if the cache file already exists. If not, the user is asked to search for a new movie. The search result is saved in the cache file. The cache file now exists and a new session starts.
If the cache file exists, the script examines its content. If the content is in the cached search result, the script will check if the user entered a number to select a movie from the results list, and save that selected movie into the cache file.

If the cache file content is a single selected movie, the script asks the user to drill down into details of the movie. If the user says “done,” the script deletes the cache file and waits for a new search query.

The code below shows the PHP script structure:

```php
<?php
// The bot should NOT respond to its own messages!
if ($_REQUEST["user_name"] == "slackbot") {
    exit;
}

// This is the cache file name
$cache_file = $_REQUEST["token"] . '.txt';

if (!file_exists($cache_file)) {
    // 1. No cache file. This is a new conversation. Start with search
    // The search results JSON from OMDB is in $json_str, we will store
    // it in the cache file now
    file_put_contents($cache_file, $json_str);
    // RESPONSE: Create movie list and ask the user to select one.
} else {
    $cache_content = file_get_contents($cache_file);
    if (strlen($cache_content) < 10) {
        // 2. The cached file has a selected IMDB movie ID. We are expect-
        // ing a "detail" command here
        $command = strtolower($_REQUEST["text"]);
        if ($command == "done") {
            unlink ($cache_file); // Delete the cache file
            exit;
        }
        // RESPONSE: Give the user the details link
    } else {
        // 3. The cached file has the search results. We are expected a
        // number to select a movie.
        // Save the selected movie's IMDB ID to the cache file
        // RESPONSE: ask the user to enter a "detail" command for the se-
        // lected movie
    }
}
```

The PHP script queries the OMDB server for a movie search, transforms the OMDB results into Slack JSON, returns the Slack JSON as the response, and then saves the OMDB results in a cache file. The OMDB return result is as follows:

```
{"Search":{
    "Title":"Star Wars: Episode IV - A New Hope",
    "Year":"1977",
    "imdbID":"tt0076759",
```
"Type":"movie",
"Poster":"http://ia.media-imdb.com/images/M/MV5BMTU4NTczODkwM15BMl5BanBnXkFtZTcwMzEyMTIyMw@@._V1_SX300.jpg"
},
{"Title":"Star Wars: Episode V - The Empire Strikes Back",
"Year":"1980",
"imdbID":"tt0080684",
"Type":"movie",
"Poster":"http://ia.media-imdb.com/images/M/MV5BMjE2MzQwMTgxN15BMl5BanBnXkFtZTcwMDQzNjk2OQ@@._V1_SX300.jpg"
},
... ...
... ...
"totalResults":"294",
"Response":"True"

After the above PHP script processes it, the JSON data sent back to Slack becomes:

```json
{
"text": "I have found the following movies matching your search. Please respond with a number to learn more about a movie.

1 Star Wars: Episode IV — A New Hope(1977)
2 Star Wars: Episode V — The Empire Strikes Back(1980)
3 Star Wars: Episode VI — Return of the Jedi(1983)
4 Star Wars: Episode I — The Phantom Menace(1999)
8 Star Wars: The Clone Wars(2008)
"mrkdwn": true
}
```

Below is PHP script in more detail to show how the search is performed.

```php
<?php
if (!file_exists($cache_file)) {
    // This is a new conversation. Start with search
    $json_str = file_get_contents("http://www.omdbapi.com/?s=" . urlencode($_REQUEST["text"]) . "&r=json");
    file_put_contents($cache_file, $json_str);

    $movies_list_str = "";
    $json = json_decode($json_str, true);
    for($i=0; $i<count($json["Search"]); $i++) {
        $movies_list_str = $movies_list_str . ($i+1) . " " . $json["Search"][$i]["Title"] . "(" . $json["Search"][$i]["Year"] . ")\n";
    }

    {
        "text": "I have found the following movies matching your search. Please respond with a number to learn more about a movie." ,
        "mrkdwn": true
    }
}
```
Next, the user sends in a number to select a movie from the list. At this point, the cache file should already exist, and the content of the cache file is the OMDB result. Once the PHP script detects a valid number, it selects the movie and writes its IMDB ID into the cache file. The script responds with a message that highlights the selected movie and asks the user to further specify what he or she wants to know about that movie. The PHP script section is as follows.

```php
// This is number selector on JSON array
$json = json_decode($cache_content, true);

$num = intval($_REQUEST['text']);
if ($num < 1 || $num > count($json['Search'])) {
    
    "text": "Sorry, I did not understand. Please respond with a number in the list",
    "mrkdwn": true
}

else {
    $movie = $json['Search'][$num - 1];
}

"text": "You selected *<?= $movie['Title'] ?>* (<?= $movie['Year'] ?>)
Please respond with one of the following: _detail_, _showtime_, _cast_, _trivia_, _quotes_ or _done_ for more",
"mrkdwn": true

file_put_contents($cache_file, $movie['imdbID']);
exit;
```

Finally, the user sends a message to ask for details for that movie. The PHP script creates a custom IMDB link to send back to the Slack channel. If the user says “done,” the PHP script will delete the cache file and be ready for the next movie query.

```php
// this is IMDB ID
$command = strtolower($_REQUEST['text']);
if ($command == "detail") {
    $imdb_url = "Follow this link for detail: http://www.imdb.com/title/" . $cache_content;
} elseif ($command == "showtime" || $command == "showtimes") {
    $imdb_url = "Follow this link for showtimes: http://www.imdb.com/showtimes/title/" . $cache_content;
} elseif ($command == "cast") {
    $imdb_url = "Follow this link for full cast and crew:
```
The full script of omdb_search.php can be found in Appendix A. It is more elaborate than the code listings above, as it handles basic user input errors.

**STUDENT PERCEPTION SURVEY**

A survey was conducted at the end of the semester to examine the students’ perception of the chatbot programming project during the semester of Spring 2017. The goal of the survey is to help us understand the drivers of students’ acceptances of the chatbot programming project and chatbot technology. It could also provide insights on designing interventions targeted at the students who may be less inclined to use or program chatbots.

The respondents consisted of undergraduate students majoring in Computer Information Systems. None of them had background experiences in chatbot programming. A total of 35 valid responses were collected and analyzed. The survey was anonymous and institutional review board (IRB) review was exempted. The survey was voluntary and was given at an IS project management class. The level of students were seniors.

The survey instrument was created based on the Unified Theory of Acceptance and Use of Technology (UTAUT) model (Venkatesh, Morris, Davis, & Davis, 2003). The UTAUT model is a widely-used instrument to assess the likelihood of success for new technology introductions. To determine new technology usage intention and behavior, UTAUT theory holds that there are several key constructs: 1) performance expectancy, 2) effort expectancy, 3) attitude, 4) social influence, 5) facilitating conditions, 6) self efficacy, and 7) anxiety. These are direct determinants of usage intention and behavior. Based on the UTAUT model, we have examined the above determinants of intention and usage. We also collected responses on Behavioral Intention, specifically the intention to use and learn chatbots in the near future. The answers to the questions were created and coded in a 7-point Likert Scale (Strongly Disagree is coded as 1, Strongly Agree is coded as 7). The survey instrument is included in Appendix B of this article.

The means and standard deviations of the responses to each survey question are listed in Table 1. Figure 11 shows the response mean of each factor. For each factor, we calculated the average of the responses to the survey questions, and use the average in Table 2 to Table 8 and Figure 12 to Figure 18. For instance, if the respondent chooses 6 in Performance Expectancy question 1, 5 in question 2, 6 in question 3, the computed score for this response is 5.67, the average of 6, 5, and 6.
# Table 1. Means and Standard Deviations of the Responses

<table>
<thead>
<tr>
<th>Factors</th>
<th>Questions</th>
<th>Mean</th>
<th>S.D.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Expectancy</td>
<td>1. I find learning chatbots is useful.</td>
<td>5.60</td>
<td>1.42</td>
</tr>
<tr>
<td></td>
<td>2. Developing chatbots enables me to understand the CIS subject better</td>
<td>5.43</td>
<td>1.44</td>
</tr>
<tr>
<td></td>
<td>3. Learning chatbots improves my knowledge in CIS.</td>
<td>5.71</td>
<td>1.47</td>
</tr>
<tr>
<td>Effort Expectancy</td>
<td>4. It is easy for me to become skillful in developing the chatbots.</td>
<td>3.74</td>
<td>1.56</td>
</tr>
<tr>
<td></td>
<td>5. I find developing chatbots is easy to learn.</td>
<td>2.80</td>
<td>1.43</td>
</tr>
<tr>
<td></td>
<td>6. Learning to program chatbots is easy for me.</td>
<td>2.77</td>
<td>1.61</td>
</tr>
<tr>
<td>Attitude</td>
<td>7. Programming chatbots is fun.</td>
<td>4.49</td>
<td>1.72</td>
</tr>
<tr>
<td></td>
<td>8. I like programming chatbots.</td>
<td>4.26</td>
<td>1.70</td>
</tr>
<tr>
<td></td>
<td>9. Programming chatbots makes the class more interesting.</td>
<td>4.91</td>
<td>1.79</td>
</tr>
<tr>
<td>Social Influence</td>
<td>10. People on my team motivate me to learn chatbots.</td>
<td>4.31</td>
<td>2.18</td>
</tr>
<tr>
<td></td>
<td>11. The instructor of this course encourages me to learn chatbots.</td>
<td>5.54</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>12. My team has supported each other in learning chatbots.</td>
<td>5.00</td>
<td>1.91</td>
</tr>
<tr>
<td>Facilitating Conditions</td>
<td>13. I have the resources necessary to develop the chatbots.</td>
<td>4.14</td>
<td>2.02</td>
</tr>
<tr>
<td></td>
<td>14. I have the knowledge necessary to program the chatbots.</td>
<td>3.49</td>
<td>1.80</td>
</tr>
<tr>
<td></td>
<td>15. A specific person (or group) is available for assistance with</td>
<td>4.14</td>
<td>1.94</td>
</tr>
<tr>
<td></td>
<td>development difficulties.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self Efficacy</td>
<td>16. I could develop the chatbots if there is no one around to tell me</td>
<td>3.11</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>what to do.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17. I could develop the chatbots if I have a lot of time to complete the</td>
<td>5.46</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>tasks.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18. I could develop the chatbot if I could find help information when I</td>
<td>6.03</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>got stuck.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anxiety</td>
<td>19. I feel anxious about programming the chatbot.</td>
<td>4.40</td>
<td>1.77</td>
</tr>
<tr>
<td></td>
<td>20. Programming chatbots is intimidating to me.</td>
<td>4.29</td>
<td>1.98</td>
</tr>
<tr>
<td></td>
<td>21. I hesitate to develop the chatbot for fear of making mistakes.</td>
<td>3.03</td>
<td>1.85</td>
</tr>
<tr>
<td>Behavioral Intention</td>
<td>22. I plan to use chatbots in the near future.</td>
<td>4.40</td>
<td>1.56</td>
</tr>
<tr>
<td></td>
<td>23. I intend to learn more about chatbots in the near future.</td>
<td>5.06</td>
<td>1.63</td>
</tr>
<tr>
<td></td>
<td>24. I predict I will use/learn chatbots in the near future.</td>
<td>4.89</td>
<td>1.41</td>
</tr>
</tbody>
</table>
Performance expectancy (Table 2, Figure 12) is the highest rated factor among all the determinants. A bit over Seventy-four percent of responses agree or strongly agree that learning chatbots is very useful, chatbot programming projects have enabled them to understand the CIS subject better, and learning chatbots improves their knowledge in CIS.

Table 2. Frequency and Cumulative Percentage of Responses - Performance Expectancy

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Neutral</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>5</td>
<td>14.29%</td>
</tr>
<tr>
<td>Agree</td>
<td>17</td>
<td>48.57%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>9</td>
<td>25.71%</td>
</tr>
</tbody>
</table>
Students also tend to agree that Social Influence has positively motivated them to learn chatbot programming (Table 3, Figure 13). More than 65% students agree to some extent that people on their project team motivated them to learn chatbots, the instructor encouraged them to learn chatbots, and their teams have supported each other in learning chatbots.

**Table 3. Frequency and Cumulative Percentage of Responses – Social Influence**

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>1</td>
<td>2.86%</td>
</tr>
<tr>
<td>Disagree</td>
<td>1</td>
<td>2.86%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>4</td>
<td>11.43%</td>
</tr>
<tr>
<td>Neutral</td>
<td>6</td>
<td>17.14%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>4</td>
<td>11.43%</td>
</tr>
<tr>
<td>Agree</td>
<td>10</td>
<td>28.57%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>9</td>
<td>25.71%</td>
</tr>
</tbody>
</table>
Though most of the students have no prior experiences programming chatbots, their self-efficacy towards chatbot programming remains quite high after working through the programming project (Table 4, Figure 14). Over seventy-four percent of respondents agree to various extents that they could develop chatbots even if there is no one around to tell them what to do, if they have a lot of time to complete the tasks, and if they could find help information when they became stuck.

**Table 4. Frequency and Cumulative Percentage of Responses – Self Efficacy**

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Neutral</td>
<td>7</td>
<td>20.00%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>11</td>
<td>31.43%</td>
</tr>
<tr>
<td>Agree</td>
<td>12</td>
<td>34.29%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>3</td>
<td>8.57%</td>
</tr>
</tbody>
</table>
Despite the difficult tasks, over 71% of respondents agree to various degrees that chatbot programming is fun, they like programming chatbots, and programming chatbots makes the class more interesting. Apparently, they truly enjoyed the programming projects (Table 5, Figure 15).

**Table 5. Frequency and Cumulative Percentage of Responses – Attitude**

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>4</td>
<td>11.43%</td>
</tr>
<tr>
<td>Neutral</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>12</td>
<td>34.29%</td>
</tr>
<tr>
<td>Agree</td>
<td>9</td>
<td>25.71%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>4</td>
<td>11.43%</td>
</tr>
</tbody>
</table>
With regards to effort expectancy questions (Table 6, Figure 16), a little over 22% of responses somewhat agree that it is easy to become skillful in developing chatbots, and it is easy to learn chatbot programming. About 54% of students find it difficult (strongly disagree, disagree, or somewhat disagree). The rest of the students responded neither agree nor disagree. This result is understandable since chatbot programming is completely new technology for students. In addition, when programming for the chatbot, students also need to use a new programming language (PHP), set up web hooks, and interact with APIs. To most of the students, this is new knowledge to learn aside from chatbot programming. Therefore, it is not surprising most students agree that chatbot programming is not easy to learn and becoming skillful in developing the chatbots takes great effort.

Table 6. Frequency and Cumulative Percentage of Responses – Effort Expectancy

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>3</td>
<td>8.57%</td>
</tr>
<tr>
<td>Disagree</td>
<td>6</td>
<td>17.14%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>10</td>
<td>28.57%</td>
</tr>
<tr>
<td>Neutral</td>
<td>8</td>
<td>22.86%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>6</td>
<td>17.14%</td>
</tr>
<tr>
<td>Agree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>0</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
Besides effort expectancy, we also measured the students’ anxiety factor (Table 7, Figure 17). Again, because programming chatbots are new to all the students, it is not surprising to see the anxiety level is high. Around 48% of the respondents agree to various extents that they feel anxious about programming the chatbot, programming chatbots is intimidating, and they hesitate to develop the chatbot for fear of making mistakes.

**Table 7. Frequency and Cumulative Percentage of Responses – Anxiety**

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Disagree</td>
<td>8</td>
<td>22.86%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>5</td>
<td>14.29%</td>
</tr>
<tr>
<td>Neutral</td>
<td>5</td>
<td>14.29%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>5</td>
<td>14.29%</td>
</tr>
<tr>
<td>Agree</td>
<td>11</td>
<td>31.43%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>1</td>
<td>2.86%</td>
</tr>
</tbody>
</table>
The most encouraging result is the positive responses on Behavioral Intention (Table 8, Figure 18), which measures students’ intention to learn and use chatbots in the near future. More than 70% of respondents indicated that they will use or learn chatbots in the near future; among them, 40% of responses agree or strongly agree with those statements. The overwhelmingly positive responses are impressive given that this is the first time for the students to program and learn chatbots. Despite the high effort expectancy and high level of anxiety in learning chatbot programming, students are not discouraged by the challenging projects. Rather, their interests in learning chatbots remain high and the performance expectancy also is ranked high. In future research, we plan to examine more the motivational factors for students to learn chatbots, and also design intervention to reduce the high level of anxiety.

Table 8. Frequency and Cumulative Percentage of Responses – Behavioral Intention

<table>
<thead>
<tr>
<th>Responses</th>
<th>Frequency</th>
<th>Percent %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Somewhat Disagree</td>
<td>2</td>
<td>5.71%</td>
</tr>
<tr>
<td>Neutral</td>
<td>6</td>
<td>17.14%</td>
</tr>
<tr>
<td>Somewhat Agree</td>
<td>11</td>
<td>31.43%</td>
</tr>
<tr>
<td>Agree</td>
<td>9</td>
<td>25.71%</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>5</td>
<td>14.29%</td>
</tr>
</tbody>
</table>
DISCUSSION

In this article, we introduced a teaching case on teaching chatbot programming in an information systems class. We explained the development environment using Slack with step-by-step project instructions. Additionally, we conducted a survey on students’ perceptions of the projects and discussed the results. This teaching case could be useful for educators in the IT education field when introducing chatbot programming to students.

In this introductory project, we focused our discussions on how to construct an intelligent bot to query movie information. It has the three key components for any intelligent bot:

- A messaging user interface. This project uses Slack. However, it could be other messaging platforms or even a custom app the students build themselves.
- A server to receive messages and create responses. In our project, students create simple PHP scripts for this purpose.
- Backend services are needed to provide the knowledge required for the conversation or perform actions the user demands. In this project, we use OMDB web services for movie information.

There are many enhancements students can do to improve their bots. The first and most obvious is to make the bot a “person” in the chat application. In our example, we used slash commands or messaging channels as bots. They served our purposes, but a “person” could do even more. For instance,

- Any user can have one-on-one conversation with the bot user.
- The bot user can be brought into a message channel to participate in a conversation, and then leave once it is no longer needed.
- The bot user can also proactively message human users to start a conversation rather than simply respond to a human user’s requests.
In addition, the “intelligence” of the bot is determined by the backend service it has access to. In our example, all the knowledge about movies comes from OMDB web services. In the “API economy,” more and more services are available as web services, and the bots can do more and more. A good example is the previously mentioned HuBot in GitHub—it can already do many human tasks around the office. Thus, students could utilize many interesting APIs to enhance their bots.

CONCLUSION

We believe this teaching case is among the first to introduce IS students to chatbot programming. For our students, learning this new technology and learning how to actually program a chatbot has tremendous benefits. The skills they learned can be directly applied in their future career given how many companies are developing the chatbots in serving customers better. Although currently there is a lack of teaching cases and resources on chatbot programming, as chatbots become more prevalent in business applications and as chatbot programming becomes more a desirable skill for IS students, hopefully, more teaching resources will be available.

There are some limitations of this teaching case, which hopefully we will be able to address in the future. Given the small size of the class, we only had 35 valid survey responses. 35 data records do not give sufficient statistical power for us to conduct structural equation modeling this time around. However, we plan to continue collecting data using the same UTAUT survey instrument. Once the number of data records is sufficient for structural equation modeling, we will analyze how determinants, including performance expectancy, anxiety, social influence, attitude, self-efficacy, etc., may have impacts on behavior intention. Hopefully, more insights will be generated. Moreover, in future research, it will be interesting to track the actual adoption behavior instead of behavioral intention. Last but not least, given the high level of anxiety and effort expectancy, in the future, more support from the instructors and more help resources for the students are essential to ease the anxiety level in learning chatbots, especially for first-time learners.

REFERENCES


How to Program a Chatbot


APPENDIX A. OMDB_SEARCH.PHP CODE

```php
omdb_search.php
<?php
if ($_REQUEST['user_name'] == "slackbot") {
    exit;
}
$cache_file = $_REQUEST['token'] . '.txt';
if (!file_exists($cache_file)) {
    // This is a new conversation. Start with search
    $json_str = file_get_contents("http://www.omdbapi.com/?s=" . urlen-
code($_REQUEST['text']) . "&r=json");
    file_put_contents($cache_file, $json_str);
    $movies_list_str = "";
    $json = json_decode($json_str, true);
    for($i=0; $i<count($json['Search']); $i++) {
        $movies_list_str = $movies_list_str . ($i+1) . " " . $json['Search'][$i]['Title'] . "(" . $json['Search'][$i]['Year'] . ")\n";
    }
?>
{
    "text": "I have found the following movies matching your search. Please respond with a number to learn more about a movie.\n"." . $movies_list_str . "",
    "mrkdwn": true
}
<?php
} else {
    $cache_content = file_get_contents($cache_file);
    if (strlen($cache_content) < 10) {
        // this is IMDB ID
        $command = strtolower($_REQUEST['text']);
        if ($command == "detail") {
            $imdb_url = "Follow this link for detail: http://www.imdb.com/title/" . $cache_content;
        } elseif ($command == "showtime" || $command == "showtimes") {
            $imdb_url = "Follow this link for showtimes: http://www.imdb.com/showtimes/title/" . $cache_content;
        } elseif ($command == "cast") {
```
$imdb_url = "Follow this link for full cast and crew: http://www.imdb.com/title/" . $cache_content . "/fullcredits";

} elseif ($command == "trivia") {
    $imdb_url = "Follow this link for trivia: http://www.imdb.com/title/" . $cache_content . "/trivia";
} elseif ($command == "quotes") {
    $imdb_url = "Follow this link for quotes: http://www.imdb.com/title/" . $cache_content . "/quotes";
} elseif ($command == "done") {
>
{
"text": "Bye <==$_REQUEST["user_name"]?>. See you next time!",
"mrkdwn": true
}
<?php
    unlink($cache_file);
    exit;
} else {
>
{
"text": "Sorry, I did not understand. Please respond with one of the following: _detail_, _showtime_, _cast_, _trivia_, _quotes_ or _done_",
"mrkdwn": true
}
<?php
    exit;
} else {
>
{
"text": "<== $imdb_url ?>",
"mrkdwn": true
}
<?php
} else {
    // This is number selector on JSON array
    $json = json_decode($cache_content, true);
    $num = intval($_REQUEST["text"]) ;
    if ($num < 1 || $num > count($json["Search"])) {
>
Long, Yuan, & Lee

{  
"text": "Sorry, I did not understand. Please respond with a number in the list",
"mrkdwn": true
}

<?php
    exit;
} else {
    $movie = $json['Search'][$num - 1];
?>

{  
"text": "You selected *<?= $movie['Title'] ?>* (<?= $movie['Year'] ?>)

Please respond with one of the following: _detail_, _showtime_, _cast_, _trivia_, _quotes_ or _done_ for more",
"mrkdwn": true
}

<?php
    file_put_contents($cache_file, $movie['imdbID']);
    exit;
}  
}  
}

?>
APPENDIX B. SURVEY INSTRUMENT

CHATBOT PERCEPTION SURVEY

All questions have the same choices based on 7 Point Likert Scale:

- Strongly disagree (1)
- Disagree (2)
- Somewhat disagree (3)
- Neither agree nor disagree (4)
- Somewhat agree (5)
- Agree (6)
- Strongly agree (7)

Performance Expectancy

Q1. I find learning chat bots is useful.
Q2. Developing chat bots enables me to understand the CIS subject better.
Q3. Learning chat bots improves my knowledge in CIS.

Effort Expectancy

Q4. It is easy for me to become skillful in developing the chat bots.
Q5. I find developing chat bots is easy to learn.
Q6. Learning to program chat bots is easy for me.

Attitude Toward Developing Chat Bots

Q7. Programming chat bots is fun.
Q8. I like programming chat bots.
Q9. Programming chat bots makes the class more interesting.

Social Influence

Q10. People on my team motivate me to learn chat bots.
Q11. The instructor of this course encourages me to learn chat bots.
Q12. My team has supported each other in learning chat bots.

Facilitating Conditions

Q13. I have the resources necessary to develop the chat bots.
Q14. I have the knowledge necessary to program the chat bots.
Q15. A specific person (or group) is available for assistance with development difficulties.

Self Efficacy

Q16. I could develop the chat bots if there is no one around to tell me what to do.
Q17. I could develop the chat bots if I have a lot of time to complete the tasks.
Q18. I could develop the chat bot if I could find help information when I got stuck.

Anxiety

Q19. I feel anxious about programming the chat bot.
Q20. Programming chat bots is intimidating to me.
Q21. I hesitate to develop the chat bot for fear of making mistakes.

Behavioral Intention to Use/Learn Chatbot

Q22. I plan to use chat bots in the near future.
Q23. I intend to learn more about chat bots in the near future.
Q24. I predict I will use/learn chat bots in the near future.
BIOGRAPHIES

**Dr. Ju Long** is an associate professor of Computer Information Systems at the McCoy College of Business at Texas State University. She has a Ph.D. degree from the University of Texas at Austin, and a Master degree from the University of Michigan in Ann Arbor. She studies technology's impact on business and how new technologies are changing our world. She has published in many academic journals in the areas of information technology security, open source software development, mobile technologies, and healthcare IT.

**Dr. Michael Juntao Yuan** received his PhD from University of Texas at Austin. He is currently the chief scientist at the CyberMiles Foundation. His primary research interests include technology innovation, healthcare IT, and entrepreneurial studies.

**Dr. Hsun-Ming Lee** received his PhD in Industrial Engineering with a focus on information engineering from Arizona State University, USA. He is currently an Associate Professor of Computer Information Systems in the McCoy College of Business Administration at Texas State University, USA. His research interests include web engineering, technology-enhanced learning, and decision support systems.