Nova: A Teachable Agent for Novice Programmers

Amit Kumar, Xiaoyi Tian, Zachary Wilkerson

Project Final Report to Spoken Dialogue Systems, CIS 4930/6930
Fall 2021

1 Introduction

1.1 Motivation

Introductory computer science courses can be challenging for novice students who have limited or no prior programming experience, resulting in high failure and attrition rates (Bennedsen, & Caspersen, 2019). One important skill that novice students struggle with is the development of computational thinking (CT), which is a complex mental process that involves logically formulating and solving problems (Wing, 2014). In addition, traditional text-based programming has rigorous language syntax and structure rules, which leads to steep learning curves, high mental workload from learners, which might result in frustration, and eventual attrition. Many studies have explored alternative ways (such as Snap!, Scratch) of learning computational thinking and problem-solving skills while eliminating those language-specific challenges. A spoken dialogue system that allows users to explain a programming concept and solve problems by natural language has great potential for improving novice programmers' learning experiences.

"Learning by teaching" is an effective pedagogy. By teaching, learners are encouraged to construct more explanations, reflect on their own misconceptions when correcting errors, and elaborate on their knowledge, which leads to learning (Roscoe & Chi, 2007). Teachable agents have demonstrated success in influencing self-efficacy, motivation and learning (Jacq et al., 2016). Teaching an agent can be beneficial where learners can engage more in learning by feeling more responsible for their agent and less negative repercussion of failure, called the protégé effect (Chase et al. 2009), and it won't hurt the tutee during failures when the tutee is an agent instead of a human.

The goal of this project is to design and develop a spoken dialogue system, acting as a teachable agent, to allow novice programmers to teach computer science concepts and solving problems while they are learning.

1.2. Description of System

Our English-based system will give students the opportunity to teach four specific programming topics: Algorithms, conditions, loops and variables. The user will be able to select a concept problem and explain the concepts in a specific topic. During the problem-solving, the system will prompt the user with various terms and concepts to seek the user's suggestion and advise on it. The system will be able to ask questions, give examples, inquire about certain statements and/or claims made by the student, and acknowledge the student's responses. Note that although the system is a teachable agent, due to the scope of a course project, we intend to not letting the agent “learn” anything from the user (i.e., we will not build any knowledge model for the teachable agent that stores or track what it has learned)

Teachable agents with a social persona that adapt to the user's speech were found to be correlated with higher rapport and learning (Lubold et al. 2019). For our teachable agent, since it is acting as a tutee, we will set the voice as a young, enthusiastic learner, called *Nova*. We use an alligator as the agent representation to avoid gender-related perceptions or biases.

1.3. Example Dialogue

Below is an example of a user working with our system on a conceptual problem. This specific problem involves CS concepts such as input/output, variable and math operation. The system introduces the user to the potential topics it would like to learn, based on the material we have trained our system on. The user then goes back-and-forth with our system, working through the topic until the user has sufficiently, as deemed by the system, described the solution.

SYSTEM: [Introduction]
SYSTEM: Alright, let's do it! I'm learning the basic concepts of algorithms, variables, loops, and conditions (*displayed through visual interface)*. Which one would you like to start with?
USER: I want to solve a system of 3 equations.
SYSTEM: That problem is a bit too complex for me, as I’m still a beginner! Could we try something easier? For example, you can teach me *(random topic selection from database query)* <u>how to add 2 numbers</u>.
(Quiet for a few seconds as user reads possible topic list)
USER: Let’s try adding two variables.
SYSTEM: Okay! I’m trying to design a program that stores two numbers, each as a different variable, adds them together, and then outputs their sum. Where should we start?
USER: First, we should create a variable to store one of the values.
SYSTEM: Okay, why do we do that?
. . .
SYSTEM: Okay, Great! What should we do next?
USER: Well that’s it! We have solved it now. If we want we can also output the third variable.
SYSTEM: What is an output?
USER: An output of a program is something you display to the user.
SYSTEM: How do we output a variable?
USER: Make a print statement.
SYSTEM: Hooray! We did it! Thank you so much for helping me!

2. Description of System Architecture

2.1. Initial Prototype and Results

2.1.1 Initial Prototype

Figure 1 and 2 show our initial prototype where we presented users with an interface to communicate with Nova using their voice. The team had utilized Dialogflow ES for developing the conversational agent. On the initial landing page users were greeted by Nova, where it briefly explains the user about its purpose and then takes the user's name to get started. On the main interaction page, users are further briefed about the interface controls and the process to initiate/end the conversation.

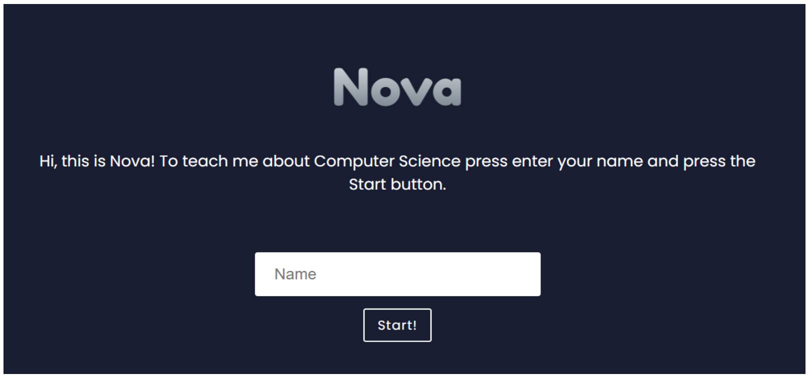

Figure 1: Initial round robin landing page.

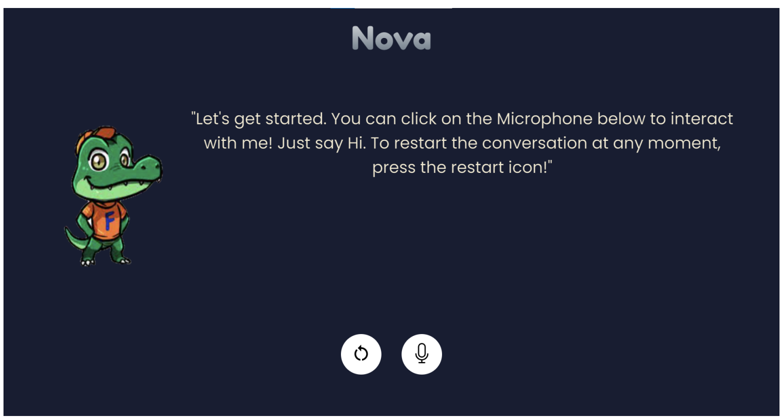

Figure 2: Initial round robin Interaction page.

2.1.2 Initial Round Robin Findings

A total of 22 users (8 users on day one, 14 users on day two) tested the system for the initial round robin. Users were able to teach Nova among four basic Computer Science concepts. Nova used a robotic voice for its responses, with a gator as an avatar. After the interaction, the users were asked to fill out a brief post-survey, which contains open-ended questions about the system interface, the dialogue and the survey. We present our user feedback below.

Interface:

Overall, users were able to use the interface effectively without needing much help from the team members and felt that the interface was visually appealing. One user said the following: *"It's really cute. I think the interface is gorgeous and really fun to use, actually!".*

Some users, however, did face issues while interacting with Nova. Sometimes users were wheel-spinning in a single sub-step repeatedly. Long utterances (transcript) having a big font would go over the page at times. While some users were able to exit from the conversation, many didn't know what to do if they wanted to exit it. Additionally, the team also recognized that certain browsers/computers had issues with the microphone permission.

Dialogue System:

Overall, our users found Nova to be fun to talk to and the topics available through Nova were engaging. Many users felt frustrated about the long generated text and the constant cut-offs. One user noted: *"When Nova cut me off before I could answer, she acted as if I did actually answer. This made me feel like she wasn't actually listening."*

We noticed issues from our users exiting the conversation. At this phase, Nova had a limited training set ("I'm done", "stop", "bye", etc) for exiting the conversation so it was not quite successful in recognizing users' intent for leaving the conversation. In addition, there is no visual indicator on the interface that could guide the exit process.

We also found our speech recognition module had a hard time detecting certain programming keywords such as "loops", which results in unsuccessful intent-matching. However, improving speech recognition accuracy is not the primary goal of the project, we compromise this issue by adding the frequent negative examples (e.g., "lutz", "Lupe", "lux") in the training phrases for the intent so that Nova can still pick up on the "loops" intent accurately even when it receives an inaccurate message.

Moreover, our users perceived Nova as being too smart to be a novice learner. As one user commented: *"I had told Nova a basic definition and it would give me a more advanced definition than what I gave it."* Similarly, another user commented that: *"Nova's description of variables was far more in depth than what I said."* Many users wish to go more in depth on explaining the concepts, as one user commented "*It would make more sense for Nova to ask me to explain better before understanding what variables do.*"

Survey:

The survey was perceived as generally easy-to-understand and thorough. Some users were confused about the items "Nova was awkward in talking to me", "Nova's conversation with me was easy" and "I helped Nova complete the task". We decided to drop five ambiguous questions from the post survey, resulting in 14 likert-type questions measuring rapport, social persuasion, satisfaction and behavioral intention in the final version. We anticipate the majority of users will fill out the pre-survey in 2 minutes and the post-survey in 3 minutes.

2.2. Functionality Expansion and Refinement

Based on both the verbal feedback and analysis of the responses from the survey, the following changes were made to the conversational agent, the interface, and the survey:

Platform: Dialogflow ES -> Dialogflow CX

Our users were at times stuck in a conversational loop. Since Dialogflow ES doesn't allow managing states, we transferred our agent to Dialogflow CX which gives us the ability for state management. The ES and CX versions of Dialogflow are independent platforms, we started the dialogue programming from scratch after transferring to CX.

Better Dialogue Content

- Additional coding examples for the conditionals problem (e.g., examples of if/else statements, asking the user what will be printed given the day of the week).
- More scaffolding (sub-states) to enhance the user's conversation.
- More variations on the problem-specific fallback. Instead of the generic fallback messages (e.g., "Could you say that again?", "One more time?"), we use the topic-specific questions as a hint to prompt users for a better explanation (e.g., "*Interesting. I heard loops can repeat stuff. How do they do that*?", "*Hmm... I'm not sure. Does it (conditionals) have something to do with evaluating if it's true or false*?")

Interface: Expanding the purpose and defining it more clearly

To set up proper user expectations, we added additional description to Nova's introduction on the starting page. The revised interface is shown in Figure 3 and 4.

Interface: Exit/Ending changes

- Considering the one-time interaction nature of the system round robin, the restart button was replaced as an exit button, to serve the purpose of ending a conversation.
- An exit message was added and made to be persistent as during the initial round robin users didn't remember the process to exit the conversation.
- Automatic ending and exit feature was added when the user reached the "end of session" state on Dialogflow.

Survey Changes, as discussed in 2.1.2

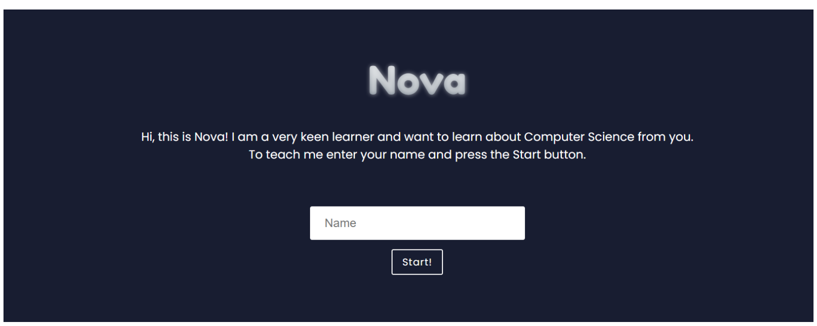

Figure 3: Final system landing page.

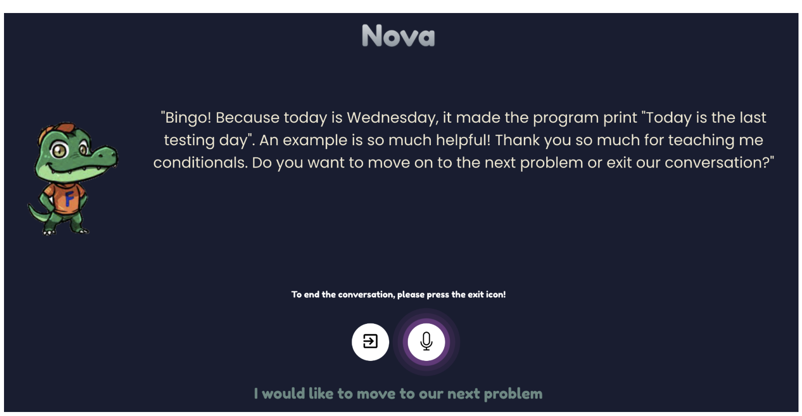

Figure 4: Final system interaction page. When the user clicks the mic button, the purple circle will pop up, indicating active listening. The live transcript will show at the bottom of the page.

2.3. User Studies

On top of training phases completed by the three of us, which we used to correctly structure the flow of dialogue as well as catch any issues, we had two additional phases where our system was tested. The first phase was an informal phase, where we had five random users outside of class test our system. These users did not complete any pre- or post- surveys. This informal phase was spread out throughout the entire duration of system development, meaning users tested at different stages of development. The second phase was a formal phase, where we had a total of 46 users test our system during designated round robin events. These users were required to complete both pre- and post- surveys. Both formal and informal testing phases gave insight into the needed modifications for our system.

2.4. Final Architecture

Our system is publicly available at https://nova-cx.herokuapp.com/. The dialogue agent was built using the Dialogflow CX framework. The system for users to work on was built using ReactJS (front-end), NodeJS (back-end), and deployed on Heroku. The complete deployment was also being managed by a Github repository supported by automatic deployment on any changes made by the team. It further utilized the React-speech-recognition module on the front-end to take user's speech and provide a live transcript on the client side. The transcript collected is then sent to the back-end where it is further processed to get a response, in the form of both text and speech, from the Dialogflow agent. This is finally returned to the front-end where the response text is displayed and the speech is played aloud for the user.

All dialogue was handled with Dialogflow via a workflow as described in Figure 5. Our user gets introduced to Nova and is then asked whether they are ready to teach or not. If the user says no, the session is ended. If the user says yes, they are asked to decide which topic they would like to teach. Once the user finishes that topic, Nova asks the user if they would like to continue by teaching another topic. If the user says no, the session is ended. If the user says yes, we repeat the process described above.

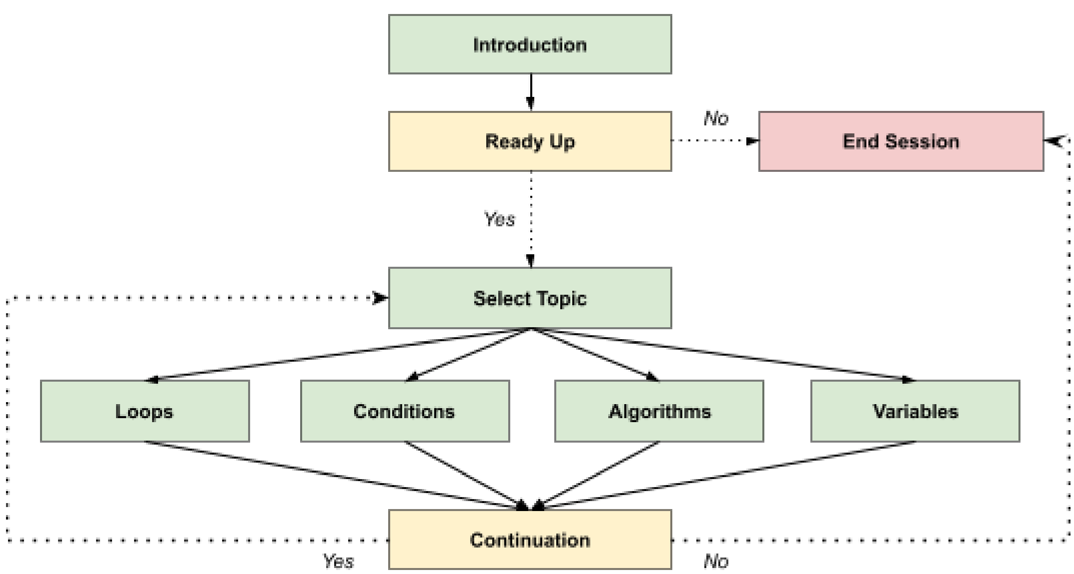

Figure 5: Dialog management workflow where green indicates an involuntary state, yellow indicates a decision state, and red indicates a termination state.

3. Final Evaluation Round Robin

3.1. User Interactions

During our final round robin evaluation period, we obtained a total of 24 system users (10 users on day one, 14 users on day two) . We found that on average, users spent just shy of three and a half minutes (201.75 seconds) interacting with Nova.

3.2. Descriptive Statistics

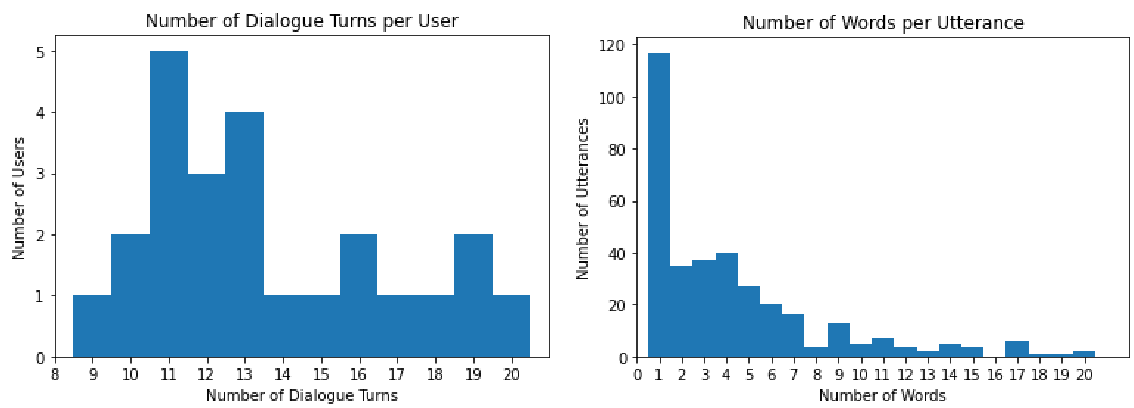

Figure 6: Histograms showing distribution of number of dialogue turns per user (left) and number of words per utterance (right).

	Independent Variables								Dependent Var
	no-matched intent (%)	TimeElapsed (in sec)	# turns	# word per utterance	# concepts taught	SE-CS	SE-teaching	Prior exp with Nova (0/1)	Rapport
Mean	0.38	201.75	13.96	4.88	2.04	3.92	3.85	0.38	29.13
Std	0.11	60.24	3.51	1.51	0.86	0.56	0.51	0.49	3.87
Median	0.39	198.00	13.00	4.50	2.00	4.00	3.88	-	30.00
Min	0.17	96.00	9.00	2.00	1.00	2.67	3.00	-	22.00
Max	0.53	334.00	21.00	9.00	3.00	4.67	4.75	-	35.00

Table 1: Descriptive statistics of independent and response variables. Each variable is described below.

In the post survey, we asked questions about the rapport level between the user and the agent, social persuasion, behavioral intention and satisfaction. These constructs are shown to be important to affect social-emotional and learning outcomes in computer-aided learning environments (Lubold et al, 2019). Majority of the questions were modified from a survey that measures rapport and social persuasion between middle school students and a social robot in a math-teaching scenario (Lubold et al, 2019). After collecting the survey responses, we conducted a principal component analysis (PCA) to determine the main factor of the outcome. The results showed that the first PC accounts for 56.6% of the variation and has the highest factor loading of rapport. Thus we made the decision of only using the aggregated score of the rapport questions as the outcome variable for our paradise model. The rapport questions include 1) Nova was easily distracted (reversed); 2) Nova listened to me; 3) My conversation with Nova was easy; 4) Nova was friendly to me; 5) I was friendly to Nova; 6) Nova and I understood each other; 7) Nova and I worked well together. All seven items are measured on a 5-point likert scale. The aggregated rapport score could theoretically range from 5 to 35, but the minimum score we obtained from the user was 22, which is still higher than neutral (21). The average *rapport* score among participants is 29.13 (*SD* = 3.87), indicating a relatively high rapport between user and our agent. Table 1 shows the descriptive statistics of the variables.

We construct eight independent variables, three from the pre-survey and five from the log data. In the pre-survey, we collected students' self-efficacy on computer science (*SE-CS*) and self-efficacy on teaching others (*SE-teaching*). Research revealed self-efficacy to be an important factor in influencing student motivation, behavior, social persuasion and learning performance (Yusuf, 2011). Considering the task of our system is teaching computer science concepts, we hypothesize students' self-efficacy in teaching and the subject (computer science in our task) can influence the outcome thus being included as a predictor. *SE-CS* and *SE-teaching* are obtained by averaging scores across three or four likert scale items (e.g., “I am sure I could do advanced work in computer science” for *SE-CS*, “I can help others learn” for *SE-teaching*). Since a fair amount of students from the class have tested our system during the initial round robin, these users might have a different expectation on the agent than users who have never interacted with the system before. We included whether the user has prior experience with Nova (*prior_exp_nova*) as an independent variable. This variable is either in 0 (without experience) or 1 (with experience). 38% of our final testers have prior experience with Nova.

In addition to predictors from the pre-survey, we also computed five dialogue/system features from log data. These variables are: percentage of no-matched intent among all intents (*no-matched intent %*), interaction time (*TimeElapsed*), number of turns initiated by the user (*# turns*), average word count per utterance (*# word per utterance*) and the number of computer science concepts the system logged in the interaction (*# concepts taught*). The distribution of some dialogue features is shown in Figure 6. All the independent variables are continuous variables except for *prior_exp_nova*.

3.3. Qualitative Reflections

Users were able to use the system to interact with Nova without needing much support from the team. Nova sometimes picked up the noise from the nearby environment and detected the noise as a part of the user's speech. Some users felt Nova knew more than they had taught, but many users had a smooth experience with the interaction and liked the interface including a live transcript of what Nova understood of their speech.

In comparison to the initial round robin, many new features were added to Nova along with having some minor changes to its existing design. Initially developed on Dialogflow ES, Nova was transferred to Dialogflow CX for better state management and enhancing user-agent interaction. More scaffolding and examples were added for a better user experience. For the interface, new buttons were added to support exit functionality and the welcome description was enhanced, along with adding persistent usage instructions for better user's understanding.

The team was able to reach two of its stretch goals along with meeting all of its basic goals. Our final database could support four Computer Science concepts *(algorithms, conditions, loops and variables*) and was also deployed on a public server (*Heroku*) for easy access across various supported systems.

4. PARADISE Model

For our paradise model, we conducted multiple linear regression with the ordinary least squares (OLS) model on the aforementioned constructs with rapport level as the response variable. Overall, the model result indicates a good fit (F (8, 15) = 2.583, p = 0.0539, R^2 = .579), with a nearly-significant F-statistic and a high R^2. Following standard practice in regression modelling, all the independent variables are z-score normalized (e.g., R. H. Baayen, 2008, Sec. 2.2), which allows us to compare the relative strength of the predictors directly. The coefficients are shown in Table 2. Only *# word per utterance* is significantly predictive of *rapport*, and the correlation is in a negative direction (β = -1.72, p = 0.025), meaning that the longer an utterance heard by Nova, the lower rapport the user would rate. This is not surprising. Many users mentioned the speech recognition displayed on the interface is often inaccurate (e.g., "*The speech recognition didn't recognize many things I said.*", "*There were some obvious mistakes in the speech to text, so it misunderstood some of the things I said. That made it somewhat hard to test.*"). Perhaps the longer an utterance is, the more recognition errors Nova can potentially make, thus making users feel less favorable to Nova.

Regarding the non-statistically significant predictors, we note the *percentage of no-matched intents* has a negative correlation (β = -1.1) with rapport, which is anticipated. Another interesting result to note is that the negative coefficient of *prior_exp_nova* (β = -1.7), meaning users who had no prior experience with Nova (0) built higher rapport by 1.7 points than users who already have experience talking to Nova (1). Since our system is designed for short interaction and has limited problem sets, users who have prior experience with Nova might feel bored teaching it again. *SE-teaching* is negatively correlated with rapport (β = -0.3), although the correlation was not significant.

	coef	std err	t	P>|t|
Intercept	29.7632	0.906	32.835	0
%_no_matched	-1.073	0.818	-1.312	0.209
TimeElapsed	0.0359	0.753	0.048	0.963
# turns	0.2636	0.714	0.369	0.717
# concepts taught	0.9878	0.78	1.267	0.225
# word per utterance	**-1.7247**	**0.695**	**-2.481**	**0.025**
SE-CS	0.0421	0.778	0.054	0.958
SE-teaching	-0.2956	0.71	-0.417	0.683
prior_exp_nova	-1.7019	1.726	-0.986	0.34

Table 2: Coefficients of independent variables (z standardized) of the linear regression model, significant predictors are bolded.

5. Collaboration Report

Our team collaborated as per our initial plan with a slight adjustment of weekly meetings as required by the team. We break down our collaboration recap into the following sections.

Meetings and Time Allocation:

We met as a group weekly on Fridays from 11am-12pm EST via Zoom. This was facilitated by other short meetings if and when required to discuss anything important. These meetings were used to discuss individual progress made throughout the previous week, work collectively on a task if needed, and discuss and assign future tasks and next steps.

Team members worked approximately 4-5 hours a week independently on different concurrent tasks. Each group member contributed equally on the system development of various aspects and the report write-up. Slack was consistently used by the team to communicate with each other during this time.

Source Code and Task Management:

For this project, we made use of Git, and a corresponding private Github repository, for source code management and managing private API keys. We used the Kyber app through Slack to handle task management.

6. References

1. Roscoe, R. D., & Chi, M. T. (2007). Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of educational research*, *77*(4), 534-574.
2. Chase, C. C., Chin, D. B., Oppezzo, M. A., & Schwartz, D. L. (2009). Teachable agents and the protégé effect: Increasing the effort towards learning. *Journal of Science Education and Technology*, *18*(4), 334-352.
3. Jacq, A., Lemaignan, S., Garcia, F., Dillenbourg, P., & Paiva, A. (2016, March). Building successful long child-robot interactions in a learning context. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 239-246). IEEE.
4. Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, *2014*, 26.
5. Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM inroads*, *10*(2), 30-36.
6. Lubold, N., Walker, E., Pon-Barry, H., & Ogan, A. (2019, June). Comfort with robots influences rapport with a social, entraining teachable robot. In *International Conference on Artificial Intelligence in Education* (pp. 231-243). Springer, Cham.
7. Yusuf, M. (2011). The impact of self-efficacy, achievement motivation, and self-regulated learning strategies on students' academic achievement. Procedia-Social and Behavioral Sciences, 15, 2623-2626.
8. Baayen, R. H. (2008). Analyzing Linguistic Data: A Practical Introduction to Statistics Using R. Cambridge University Press.