Educational Software for Off-Task Behavior

A Major Qualifying Project Report:

submitted to the faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Kristen Hughes

Date: April 29, 2010

Approved:

Prof. Ryan Baker, Major Advisor

Abstract

Off-task behavior is a problem currently facing intelligent tutoring systems as well as traditional classrooms. There are a number of reasons why students go off-task, and likewise, a number of ways for them to do so. The goals of this project were to (1) develop a potential off-task intervention method, and (2) implement an off-task detector in an existing intelligent tutoring program, which was already capable of detecting and responding to students who were "gaming the system".

Acknowledgements

I would like to thank my advisor, Professor Ryan Baker, for his understanding, patience, and guidance throughout the course of this project.

Table of Contents

ABSTRACT .... **I**

ACKNOWLEDGEMENTS .... **II**

TABLE OF FIGURES .... **IV**

1 INTRODUCTION **1**

2 BACKGROUND **2**

2.1 Introduction to Off-Task Behavior 2

2.1.1 Types of Off-Task Behavior *2*

2.1.1.1 "Gaming the System" 3

2.1.2 Effects of Off-Task Behavior *3*

2.1.3 The Reasons behind Off-Task Behavior *4*

2.2 Detection Methods 4

2.2.1 Detecting On-Task Behavior as Off-Task *5*

2.3 Off-Task Interventions 6

2.3.1 Types of Intervention Methods *6*

2.3.1.1 Advantages and Drawbacks 7

2.3.1.2 Intervention Strategies to Avoid 8

3 DESIGN **9**

3.1 Initial Design Decisions 9

3.2 Intervention Specifications 11

3.2.1 Scenario Testing of the Intervention Design *12*

4 IMPLEMENTATION **15**

4.1 Adding an Off-Task Detector to an Intelligent Tutor 15

5 FUTURE WORK **17**

6 CONCLUSIONS **19**

REFERENCES **20**

APPENDIX A: INTERVENTION STORYBOARDS **21**

A-1 Tutor Screen Layout 21

A-2 Scenario 1: A student who is on-task, but struggling 22

A-3 Scenario 2: A student is receiving help from their teacher 23

A-4 Scenario 3: An on-task student is being distracted by another student 24

A-5 Scenario 4: A student is surfing the Web 25

A-6 Scenario 5: A student who does not want to work with the tutor 26

Table of Figures

Figure 1: Table of off-task behavior functions 5
Figure 2: Scooter’s progressive moods as he detects student gaming the system 11
Figure 3: Three examples of Scooter’s positive emotional responses 12

1 Introduction

A problem currently plaguing both traditional classrooms and intelligent tutoring systems is off-task behavior. Off-task behavior occurs when “a student completely disengages from the learning environment and task to engage in an unrelated behavior” (Baker 2007). Student’s who exhibit off-task behavior, especially students who participate in “gaming the system”, have been shown to learn only about two thirds of what students who do not game the system learn (Arroyo et al. 2007; Baker, Corbett, and Koedinger 2004). This project will examine off-task behavior, why it occurs, how it can be detected, and how to intervene when students go off-task.

This project had two main goals, (1) designing a potential off-task intervention method to be implemented at a later time, and (2) implementing an instance an off-task detector into an intelligent tutoring system that currently detects gaming behavior, and uses interventions involving “Scooter the Tutor”, a tutoring agent who displays emotional responses depending on whether the system thinks the student is using the tutor correctly or trying to abuse the system.

2 Background

In the learning environment there is an unending war taking place between the acquisition of knowledge and off-task behavior. Whether students are learning in a traditional classroom setting, or using educational software, the problem of keeping students on-task, and learning effectively, persists.

2.1 Introduction to Off-Task Behavior

Off-task behavior occurs when a student completely removes their self from the learning environment, instead engaging in an unrelated activity (Baker 2007). These unrelated activities can range from having off-topic discussions with classmates to surfing the web, or even sleeping. Some of these options are only available to students outside the classroom, and without a teacher constantly watching their actions it is easier for students to go off-task. Even when using a computer inside the classroom there is more opportunity for students to avoid learning. In the traditional classroom, a student cannot typically get out of their seat, walk to the other side of the room, and start a conversation with their friends without being noticed. However, when they are using a computer, unless their teacher is constantly hovering over their shoulder, there is really nothing preventing them from using an instant messenger to accomplish the same feat.

2.1.1 Types of Off-Task Behavior

There is no single identifying action that can be associated with off-task behavior. In the classroom students can easily distract each other through conversations, note passing, and even playing games during lessons. While whispered communications between students are relatively easy to detect, any silently distracted students can be difficult to locate. Even without the distraction of other students, there are plenty of ways for a classroom-bound student to partake in off-task behavior. They might be sleeping, reading unconnected material, or performing some other, unrelated activity such as working on homework for another class, drawing a picture, etc. There is also the possibility that though they are not performing any off-task actions, they are also not paying attention to the material being taught.

A virtual classroom, while having the benefit of a 1:1 student-teacher ratio, unfortunately does not escape the predicament of students who go off-task (Baker, Corbett, and Koedinger 2004). If the educational software being used is within a classroom-type room, where students still have contact with their classmates, then the same pitfalls of the traditional classroom will persist. Students will still be distracted by each other, and one teacher cannot have their eyes on all students at once. Some of the solitary off-task activities, such as sleeping and not paying attention can also remain present in this alternative teaching environment. A computer can be full of distractions such as games, other available applications, and internet access. There is another type of off-task behavior that even the most observant teacher might miss, because this behavior occurs while students appear to be fully engaged in the software, but are not actually trying to learn anything. This is called "gaming the system".

2.1.1.1 "Gaming the System"

As mentioned in the previous section, "gaming the system" is harder to observe than students who are sleeping, surfing the web, or conversing. Gaming the system is defined as "attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly"(Baker et al. 2006). When students game the system, they might be moving through problems without really attempting to understand them, systematically guessing answers, or asking for hints faster than they can read them in order to find the answer without really trying the solve the problem (Arroyo et al. 2007; Baker 2007; Baker et al 2006).

Gaming the system can be divided into two distinct behaviors: harmful and non-harmful. Non-harmful gaming occurs when a student who has a already mastered a particular skill set is trying to move quickly through that section of the tutoring program in order to get to material which they have yet to master (Baker et al. 2006). While this behavior might be considered off-task, since the student has already mastered this skill they are not actually hurting their learning experience. Conversely, harmful gaming meets the expectations that it was named for. When students choose to game the system to get past material that they do not yet understand, they are preventing the tutor from properly teaching them, and will learn a significantly percent amount of the material (Arroyo et al. 2007; Baker, Corbett, and Koedinger 2004).

2.1.2 Effects of Off-Task Behavior

A study conducted in 2004 of five middle school classes from two different schools, showed a significant difference in the amount that students learned when they gamed the system. The students who were observed to be harmfully gaming only learned about two-thirds as much as the students who were not gaming the system (Baker et al. 2004). While this particular study did not find a significant correlation between other types of off-task behavior and learning, more recent studies have been able to find such a relationship. A study conducted in 2007 found that third grade and high school age students acknowledged that they were prone to going off-task and that it was negatively impacting their grades (Pate-Clevenger et al. 2008).

2.1.3 The Reasons behind Off-Task Behavior

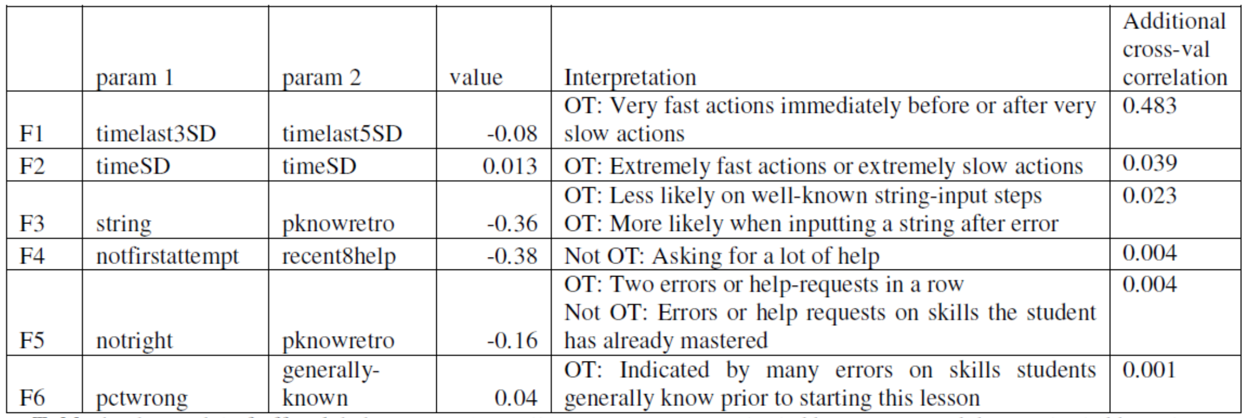
Understanding why students go off-task is a vital factor to consider in the design and implementation of an intelligent tutoring system. The right design can potentially decrease the percent of time students spend off-task. In regards to intelligent tutoring systems, off-task behavior is most often found in students who have a dislike of computers or mathematics, display passive-aggressive attitudes, are not motivated to succeed academically, and tend to hold negative feelings towards their teachers (Baker 2007). In particular, students prone to gaming the system may also display signs of frustration and typically are not interested in the subject they are supposed to be learning (Rowe et al. 2009). All of these behaviors reflect a negative attitude on behalf of the student which is directed at the tutoring technology, their teacher, or themselves. Students who look for teacher approval are more likely to game than go off-task, because they don’t want their teacher to think they aren’t trying to learn. On the other hand, students who do not care about their teachers’ opinions are less likely to try and hide any blatant off-task behavior (Baker 2007).

2.2 Detection Methods

The two types of off-task detection models that will be discussed in this section are the time-only model and a specific multi-parameter model, both originally machine-learned by Baker (2007).

As suggested by its name, the time-only model accounts for only one factor, time. This model involves determining the amount of time it should take for a student to answer part of a question. This measurement of time is determined by taking into account the average time spent on each problem step. The student’s actual time to complete a part is then determined to be off-task depending on the number of standard deviations it is above or below the average. During one particular use of this model, it was determined that students who took 3.8 standard deviations above the mean time could be considered off-task. In this case the model had a correlation of 0.46 (Baker 2007).

In order to achieve a stronger correlation, a model should account for more than just time taken for each step. The second model involves the use of six, dual parameter functions, accounting for ten distinct parameters in total. The functions are displayed in Figure 1.



	param 1	param 2	value	Interpretation	Additional cross-val correlation
F1	timelast3SD	timelast5SD	-0.08	OT: Very fast actions immediately before or after very slow actions	0.483
F2	timeSD	timeSD	0.013	OT: Extremely fast actions or extremely slow actions	0.039
F3	string	pknowretro	-0.36	OT: Less likely on well-known string-input steps OT: More likely when inputting a string after error	0.023
F4	notfirstattempt	recent8help	-0.38	Not OT: Asking for a lot of help	0.004
F5	notright	pknowretro	-0.16	OT: Two errors or help-requests in a row Not OT: Errors or help requests on skills the student has already mastered	0.004
F6	pctwrong	generally-known	0.04	OT: Indicated by many errors on skills students generally know prior to starting this lesson	0.001

Table 1. The model of off-task behavior (OT). In all cases, param1 is multipled by param2, and then multipled by value.

Figure 1: Table of off-task behavior functions (Baker 2007)

This model accounts for far more than just time spent on one step. It also looks at what type of input the student is providing, whether it is their first attempt at solving the problem, if their previous answer was wrong, how accurate their answers are overall, how often they ask for help, whether they should already know a particular skill, and how well they know the skills they are being taught, as well as performing a more detailed analysis of the time spent on each step. When the summation of the results of each of these functions is 0.5 or greater, it is assumed that the student is off-task. This model has been found to have achieved a correlation of 0.62. In previous testing, when an observer recorded one student to be more off-task than another, this multi-parameter model agreed with the observer 83% of the time (Baker 2007).

2.2.1 Detecting On-Task Behavior as Off-Task

While there have been improvements made regarding the detection of off-task behavior, the technology has not yet been perfected. A computerized tutor cannot truly see whether a student is paying attention or not in the way a human tutor would be able to, but it is able to calculate the response times, remember what skills a student has mastered, and various other statistics about the student that a human might only be able to observe. Unfortunately, while direct input based programs can tell whether or not their user is actively using the interface, they have no way of determining if an inactive student is off-task. Tutors that are dependent on user input, such as a typing an answer or clicking a button, are not able to tell the difference between a student who is off-task and a student discussing the current problem with their neighbor (Baker

2007). External devices such as eye-motion detectors and mouse sensors can add a new layer of detail about a student's behavior, but due to their cost they are not resources most public schools can invest in currently, and will not be explored in this project.

When responding to off-task behavior, it is important to bear in mind that the software will most likely provide the same reaction to a student returning from a trip to the restroom, as it will to a student who has chosen not to utilize the tutoring program. The intervention method needs to accommodate for conditions such as this.

2.3 Off-Task Interventions

Just detecting the presence of off-task behavior is not enough. The intelligent tutoring system needs to respond accordingly, or the detection will be nothing more than a piece of data stored about the student in question. Intervention is the next step in responding to off-task behavior. There have been many different approaches taken to respond to off-task behavior, the following sections will examine some of these methods.

2.3.1 Types of Intervention Methods

"Empathetic approaches to student affect have been shown to alter the affective state of the student, as well as other qualities such as motivation"(Robison, McQuiggan, and Lester 2009). In a real life tutoring situation, a tutor is attempting to teach a student, typically in a one-on-one environment, and thus is able to respond to the individual student's behavior in ways a teacher in classroom full of students cannot. If a human tutor is able to respond emotionally to a student, then a virtual tutor should have similar capabilities. Some tutoring systems display happiness when a student is making progress, or anger when a student is gaming the system (Baker et al. 2006).

One approach to combat off-task behavior is the use of a narrative-centered learning environment (NLE), which takes advantage of a student's ability to create and comprehend stories. In a narrative-centered learning environment, learning is turned into a game. NLEs incorporate "compelling plots, engaging characters, and fantastical settings" in order to pull students attention into the learning and away from outside distractions that might lead them off-task. They combine the use of intelligent tutoring systems, conversational agents, and serious games (Rowe et al. 2009).

Another method that has been used involves holding students accountable for their own actions, and then allowing them to decide how to respond to their own behavior. In this approach, after each problem students might be provided a progress report. They also could be asked to think about how much time they spent off-task versus how much time they spent productively. In this type of intervention, the software does not attempt to hinder any off-task behavior they display. However, it does provide them information about their progress so they can see the consequences of their actions (Arroyo et al. 2007).

A fourth technique to respond to off-task behavior focuses on increasing student motivation. This has been used for two distinct goals. First, simply increasing a student's enthusiasm can be seen as a goal in itself. Second, a tutor might try to improve a student's motivation in order to increase the amount of information the student is actually learning. These goals are accomplished by creating a "more empathetic, enjoyable, and motivating intelligent tutoring system" (Baker, Corbett, and Koedinger 2004).

2.3.1.1 Advantages and Drawbacks

Unfortunately, none of the above-mentioned methods have been perfected. Each has been implemented and shown positive results, but there is still room for improvement.
Having an intelligent tutoring agent that can display simulated emotion is a good start. Students using this type of tutor will hopefully feel more connected to the tutor and less like they are dealing with a lifeless computer program. However, an empathetic tutor needs more than a smile or frown if it is truly going to emulate a human tutor. A real tutor would not only react, they would also be able to offer help and encouragement to a struggling student.

In narrative-centered learning environments, a student is pulled into another world and fully immersed in a new learning environment. While this does help remove students from distractions around them, the "seductive details" can "distract, disrupt, or divert" attention from learning objectives by being too engaging (Rowe et al. 2009). Essentially, the NLE can be too good at engaging students, causing them to spend more time exploring and playing in the environment instead of progressing through the game and learning.

Holding students responsible for their actions while using a tutoring system has been proven effective. In one study in particular, when students were reminded of their progress and provided tips, there were fewer quickly made guesses and more improvements in time spent on problems rather than only on hints (Arroyo et al. 2007). The problem with this method, is that it is more preventative that intervening. As this is the case, it has little to no effect on unmotivated students who will ignore their progress reports, and not bother to reflect on how their actions are only hurting themselves.

Increasing student motivation works to involve students in the tutoring program while keeping them grounded. This method combines the advantages of an empathetic tutor and a narrative-centered learning environment. However it also brings with it some of those disadvantages. A solution that addresses all of the aforementioned mentioned shortcomings has yet to be discovered.

2.3.1.2 Intervention Strategies to Avoid

A method not included above involves removing any tutor features that can be taken advantage of by students who try to misuse tutoring programs. This elimination only addresses the issue of gaming the system, as opposed to more generalized off-task behavior. It is mentioned now due to its significant disadvantages. Generally, the features that some students may take advantage of are highly useful to students who do not attempt to abuse the system. Not only does this hurt students who use these features appropriately, it does not truly accomplish its purpose. Students determined not to make an attempt at using the tutor correctly will devise new strategies so they may continue to game the system (Baker et al. 2006; Baker, Corbett, and Koedinger 2004).

It is also important to note that the way interventions are implemented can have a strong impact on how they affect the students. Loud noises and pop-ups can irritate already frustrated students. In some cases students will pretend to be on-task just to avoid these reactions (Arroyo et al. 2007; Baker 2007).

3 Design

After completing my research, I began to address the design portion of my project. To accomplish this I first needed to decide on the general approach I wanted my intervention method to use. Based on what I had read about other intervention methods, I formulated two possible ways the intelligent tutoring system I would be working with could handle off-task behavior.

The first method I considered, involved providing students a progress report after the completion of each problem. This report would let them know how much they had improved, and give them advice regarding skills that were not showing signs of improvement. I had also intended for this method to display emotions based on how well the student was doing during the problem.

The second method I came up with combined a few aspects of the other models I had looked at. In this model, the tutoring agent would respond emotionally depending on how off-task it had found the student to be, and also respond with interventions to both periods of inactivity and when active students had a high probability of off-task behavior.

After discussing both of these options with my advisor, I was tasked with creating a set of storyboards depicting how each of these methods would appear in the tutoring environment. Upon review of these designs, we decided that the first method was more useful for prevention of off-task behavior. Since my task had been to design an intervention method, it was determined that the second option I had designed would be more appropriate.

3.1 Initial Design Decisions

One of the first design decisions I had to make was whether the tutoring program should have a separate agent to deal with off-task behavior. After some consideration, I settled on keeping the system consistent and using the same agent despite what type of intervention the student would be receiving. My main concern with creating a new agent in the program was that students would realize that the second agent only appeared when they went off-task. If students connected this agent with off-task behavior, but it appeared when they had been on-task, they might start to distrust the tutor, which would most likely end in them learning less.

To address the issue of inactivity within the tutor, I worked to make sure no techniques which produced negative results appeared in my design. I first ruled out the possibility of incorporating any sounds, as these seemed to end up causing more irritation and annoyance than productivity. Another idea I had to eliminate was the use of popup messages for the same reasons as sounds. Finally, I made sure any messages the tutor would end up displaying were as inoffensive and non-accusatory as possible.

The resulting intervention design consisted of enlarging the tutoring agent's window, rather than creating a new popup, and offering the student help. My expectation was that by enlarging the window, the student would be made aware of the tutor trying to communicate with them, without creating a new window which might end up covering an area of the screen they may have been using. Also, unlike a popup, the enlargement could happen gradually, giving the student time to notice something was occurring on the screen, rather than an abrupt, unexpected, and possibly inconvenient change. This enlarged tutor window would display the question, "Do you need any help?" followed by two options, "Yes, I want help", and "No, thank you". If the student chooses not to ask for help, the tutor window would resume its original size and allow the student to continue working on their own. If the student were to choose to receive help, they would be provided a hint, after which the tutor would ask if they needed more help. If the student did not require additional assistance, the tutor would return to normal. Otherwise it would provide more detailed help until the student felt comfortable enough to return to working independently.

A third option I considered to the original offering of help was that a student might not respond to the window for a variety of reasons. They may not have noticed when the tutor's window expanded, they also may have gone to the restroom, or even fallen asleep. In this case I knew the tutor would need to take a more direct approach while remaining as non-accusatory as possible. In the event of this extended inactivity, the tutoring program would cause the entire screen to go blank, and only a window containing the tutoring agent would remain visible. In this window the tutoring agent would ask the question, "Are you still there?" with a single button stating, "Yes, I am here" as the only possible response. This reaction should be similar in appearance to the computer having "fallen asleep" due to their inactivity, with the exception being that the tutor window can still be seen. Should the student not have seen the help offering from the tutor, but returned to actively using the program before the second part of the intervention was executed, they would be relabeled as active and not subjected to the second level reaction. However, the help message would still remain until the student noticed it and responded, or they elicited an emotional response from Scooter.

3.2 Intervention Specifications

As mentioned earlier, I used storyboards to visually depict how my intervention would work. The storyboarding process consisted of three main steps. The first set of storyboards was created before I had chosen an intervention method. I designed a single screenshot for each method depicting the general approach the tutoring agent would use to respond to off-task behavior. After deciding upon an intervention method, I was presented with five scenarios. I was then tasked with creating a more detailed set of storyboards depicting how my chosen intervention method would respond in these situations. The five scenarios consisted of:

1. A student who was on-task, but struggling
2. A student who was receiving help from their teacher
3. A student who was being distracted by another student
4. A student who was surfing the web
5. A student who did not want to work with the tutor

The second set of storyboards I created reflected my initial thought process of how the tutor should react to the given scenarios. Upon completion of these designs, I met with my advisor to go over my ideas and determine what revisions needed to be made for the final set of storyboards.

These scenarios all involve a situation where a student has been inactive for a significant period of time. In a situation where a student was suspected of off-task behavior, I planned for the tutor to react in a similar manner to its preexisting gaming detection behavior, where the tutor looks increasingly aggravated as a student shows more signs of trying to game the system, as seen in Figure 2.

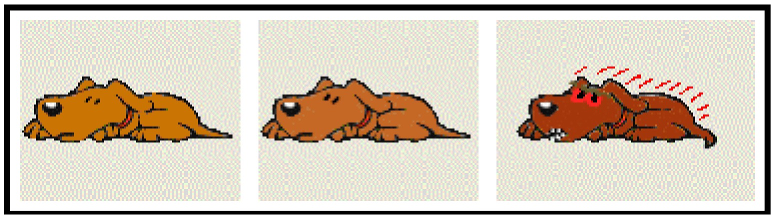


Figure 2: Scooter's progressive moods as he detects student gaming the system

However, rather than show anger towards students who were going off-task, I envisioned more of a disappointed looking agent. Since off-task behavior is not always indicative of abusing the system in the way that gaming is, I didn't think anger would be the best emotion to exhibit. I decided the positive emotions that Scooter exhibits when students appear to not be gaming would still be appropriate to display when students did not appear to be off-task. A few examples of these responses can be seen in Figure 3, below.

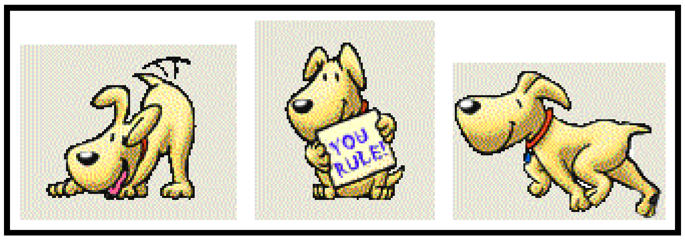


Figure 3: Three examples of Scooter's positive emotional responses

In creating the second set of storyboards, I had to consider more than just the appearance of the screen. Since detection of off-task behavior has not been perfected yet, I needed to account for students who were not off-task, but would be on the receiving end of these interventions after being mistakenly labeled as displaying off-task behavior. The completed set of storyboards can be found in Appendix A.

3.2.1 Scenario Testing of the Intervention Design

The first two scenarios are good examples of situations where students' inactivity might be misinterpreted as being off-task. In the first scenario, where a student is struggling through a problem, the tutoring software would interpret their prolonged period of inactivity, as potential off-task behavior. By offering the student help, but not outwardly acknowledging that the help is being offered due to their recent inactivity, the tutor can help the student, get them back to actively solving problems, and hopefully not have caused them any more stress than they were already experiencing.

In the second scenario, a student is on-task and receiving help from their teacher, but again, the tutor can only sense that they have been inactive. Again the tutor not accusing the student of being off-task is a vital part of this reaction. Falsely accusing a student could lead them to dislike or even resent working with the tutor if they feel that it will wrongly accuse them of being off-task when they are not. In this case, when the tutor enacts the intervention, it is relatively unobtrusive, and the student and teacher might not even notice it while they are discussing the problem. If they do notice it, they can choose the option to not receive help, and continue their discussion.

The third scenario introduces an outside factor to the tutoring environment consisting of another student who engages the first in off-task conversation. Unfortunately, by keeping the tutor's reactions subtle and polite, there is a limit on how effective an intervention will be in this case. As seen in the last two scenarios, after a period of inactivity, the tutor window will enlarge and offer the student help. If the student is looking for a way out of the conversation, they could use the tutor's help message as an excuse to get back to work. Otherwise, the tutor is powerless to stop the conversation. Making the intervention mechanism more intense could cause the students in scenarios one and two to become agitated, distressed, or to feel any other sort of negative emotion that should not be associated with a tutor. If the student is unable to leave the conversation on their own, the assistance of the teacher is necessary to address this off-task situation. The restrictions I placed on the tutor's intervention methods also cause a problem when dealing with the fifth scenario.

If a student is surfing the Web when they are supposed to be using the tutor, the standard response to inactivity will not be effective. The student will not see the first part of the response, since the tutor will probably not be visible on their screen, and when the second part of the inactivity response occurs, the student can simply confirm that they are present and continue their off-task activity. For the tutor to be able to respond in a more successful way, it needs to be able to acknowledge and respond immediately to the presence of an active web browser. For this intervention, I am assuming the tutor has the ability to detect the presence of an open web browser, and open a new window on top of it. It must also prevent the student from continuing to use the browser until they have responded to the new window. In this case, in order for the intervention to be unavoidable, I must break one of the constraints I originally gave the tutor. During this intervention, upon detecting an open web browser, the tutor would open a popup over the browser asking the student to, "Please return to the tutor", and two possible options. The first option is an "OK" button. After this has been pressed the student will have around five to ten seconds to close the browser. If the browser is still open after this time, the popup will reappear, and this pattern will continue until the student closes the browser or utilizes the second option. The other choice the student is provided is a text box, labeled "Teacher password". If they were given permission to use the Internet, their teacher can input their password, and the student will not receive any more messages from the tutor regarding the use of the Internet. They will also stop receiving messages about their inactivity until they have exited the browser.

While requiring a password to get Internet access will probably be viewed as an annoyance to teachers and students alike, I made this design decision while under the impression that students would not be regularly using the tutor and the Internet at the same time. In situations where this is the case, this response would need to be removed, or altered to make it less of a hassle for both the students and teachers.

The last scenario addresses the problem of a student who does not want to work with the tutor. The tutoring system faces a similar challenge here to the one presented in the third scenario. In order to remain as inoffensive and non-accusatory as possible, the system cannot respond in any way that could be upsetting to a student who was not off-task. For example, if the tutor responded to a student it suspected was asleep by berating them or appearing angry, but the student had actually been making a trip to the restroom, they would probably become upset due to the insensitive response the tutor had displayed. While a silent, blackened screen will not wake a sleeping student, it will also be less likely to upset any students who see it and are not off-task. Part of the reason I chose to have the majority of the screen hidden, was that it would be more likely to attract a teacher's attention, which may be needed. This becomes another situation where the teacher's attention would be required.

4 Implementation

The second goal of my project was to implement an instance of an off-task detector in the intelligent tutoring system. Before immersing myself in the actual code, I spent a few hours using the tutor to get a better understanding of how it worked at the user level. As this tutor is intended for use by sixth grade students, it could not respond to me as well as it would have if I were a sixth grade student who did not already know all of the skills it covered. Even though it frequently thought I was gaming the system due to the speed of my answers, I still feel it was an important step to take prior to adding new functionality.

The tutoring system I worked with was written in a combination of Lisp, for the back end, and Java, for the interface functionality. One of the first challenges I faced was that the code for the off-task detector was part of the back end, and I had no experience using Lisp. Since I was unfamiliar with this language, the next step I took was to examine the code in the file that implemented the gaming detector. Looking through this code was very helpful when it was time to determine what functions I would need to create.

4.1 Adding an Off-Task Detector to an Intelligent Tutor

After I had familiarized myself with the way the gaming detector was implemented, I proceeded to write my own functions. The off-task detector was written in the same file as the gaming detector to simplify the initial implementation process.

I started with the functions specified in Figure 1. After creating these, I realized that some of the parameters required by these functions did not exist in the gaming detector. My next step was to go back to my research to find out what each of these parameters represented. I was then able to create functions that would produce these values. Next I went through the entirety of the code again, and for every variable, method, and function whose name contained “gaming”, I created a duplicate version, and renamed them to be “offtask” instead.

The next step was to go through all of these duplicated methods and recode them to perform off-task detection instead of gaming detection. This was where I ended up facing my language barrier issues with Lisp. Since I had no practice programming in Lisp, I was unfamiliar with much of the syntax, and had to do more research to decode my compilation errors. Once I learned what my error and warning messages meant, all that remained was fixing my coding errors, which mostly ended up being typos. The biggest issue I found in debugging my code was that LispWorks, which I used to compile my code, did not provide the detailed error messages I was used to from using Eclipse. While it would provide sufficient information to me when the code was compiling, if I were missing a parenthesis it could not supply me any information about where this missing character should have been. The only solution I was able to find, was removing sections of code at a time until I was able to compile again. I then proceeded to add the sections back in until the code would not compile. I repeated this process using smaller section sizes each time, until I was able to pinpoint which part of the code was not compiling, and resolve it.

Once all of my off-task detection code was compiling cleanly and appeared to be error free, I changed the functions inside the file, now holding the gaming and off-task detectors, to call the off-task detector functions instead of the gaming ones. While the tutor is able to run using these functions, its functionality has not been validated. During the process of setting up the functions to call the off-task detector instead of the gaming detector, I encountered a few problems. The first being that despite setting the inside functions to use the off-task detector, a function outside of the gaming detector file was still calling the gaming functions. Despite performing multiple searches for the location of this function call, I was unable to find its containing file.

After being unable to discover the whereabouts of this file, I decided to try changing the names of the functions within the detector file, so that the off-task detector would still be called. Unfortunately, though I was able to compile and run the program with these changes, I did not have enough time to test the functionality of the off-task detector. During the testing I did manage to complete, I found that the gaming variables were still being updated, indicating the presence of more functions that would need to be restructured to work with the off-task detector instead. For development of the off-task detector to continue, extensive testing, as well as ensuring only the off-task detector was being used, would be necessary.

5 Future Work

As stated in the previous section, the off-task detector is currently implemented inside of the file where the gaming detector is located. Also, it is currently being called in place of the gaming detector. The next major undertaking in this project would be to migrate all of the functions and variables associated with the off-task detector into their own file, as well as remove any functions and variables used solely by the off-task detector from the gaming-agent code.

It will also be important to make sure that the off-task and gaming detectors can work together to calculate more accurately whether a student is on-task, off-task, or gaming. While gaming the system is a type of off-task behavior, the off-task interventions are designed more for students who are not paying attention to, or trying to avoid their work, while the gaming interventions target only students who are suspected of gaming the system. Giving an off-task, but not gaming student an intervention meant for a student that was gaming would most likely not be as effective, and vice versa.

The next major changes to the code should involve implementing the off-task interventions. While being able to detect off-task behavior is a valuable asset, it will be wasted if no action is taken to address the problem. Also, the intervention method will require a more in-depth analysis. The model I designed specifies certain constraints, which should be followed as often as possible, as well as a detailed approach for how to respond to off-task behavior in the form of inactivity. I was only able to put a few ideas together in this project regarding how to react to off-task behavior when the student is actively using the tutor.

Some other aspects that could be integrated into the system include the intervention model that I did not use. Adding a preventative layer to the system could make it more effective at stopping off-task behavior. Another option that might be added could include a reward system where students who were progressing well could play a short game as incentive to continue their hard work. Hopefully this reward system might be able to inspire less motivated students to put forth more effort so that they too could enjoy a reward.

Perhaps the most important step to be taken in this project is user testing. Despite how much time is put into the design and implementation of any project, there is no way to judge how well it works without testing it out on those it is meant to be used by. During the course of this project, I thought of two possible ways the testing might be carried out. The first option would be to have two randomly selected groups of sixth graders test the software. One group would have a copy of the software without the off-task interventions, but the detector would still be enabled for comparing results. This will be the control group. The other group, the test group, should be given a copy with the off-task interventions included. The effectiveness of the interventions could be measured by comparing the amount of off-task time spent on average by each group, as well as by asking the students in the test group how they felt about their interactions with Scooter.

Another possible test setup would be to use the same set of students for both tests. First, have them participate in the control portion of the experiment, where there are no interventions, and record each student's off-task data. At a later time, have the group use the tutor with the off-task interactions enabled, and as in the first test scenario, have the students answer questions about the tutor. This test design allows for a more direct comparison of how effective the interventions are, since you can compare each student's result for both parts of the test. However, it also removes some of the randomness that the first experiment could provide.

6 Conclusions

Off-task behavior occurs when “a student completely disengages from the learning environment and task to engage in an unrelated behavior” (Baker 2007). This behavior can be expressed in many different forms, including off-task conversation, surfing the Web, or even sleeping. Likewise, there are many different reasons for explaining why students go off-task, but almost all of them reflect negativity in the students’ thinking directed at various sources.

The two main purposes of this project were to develop a potential method of off-task intervention for future implementation, and to add an off-task detector into an intelligent tutoring system. After researching several different approaches towards the detection and intervention of off-task behavior, I was able to develop two potential intervention models. Further consideration of these two models led me to choose which one I would refine. I then created storyboards based on a series of scenarios to describe how the intervention method would work given a variety of different circumstances.

Once the design process was complete, I moved on to the implementation of an off-task detector. This detector incorporated six, dual parameter functions (see Figure 1), which considered several factors of off-task behavior including time, the number of errors made, and the type of input expected of the student. I then based the structure of this detector off of the gaming detector which the tutoring system already contained.

There are a number of opportunities for future improvement and expansion of this project. The off-task detector needs to be fully integrated into the intelligent tutoring system since it is currently still a part of the gaming detector. The intervention method has room for expansion and still needs to be implemented. Lastly, and perhaps most importantly, after the addition of a functional intervention method, the whole system will need to be tested, to determine how well it works, and reconfigured according to both the results and the student feedback.

References

Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., & Woolf, B.P. (2007). Repairing disengagement with non-invasive interventions. *Frontiers in Artificial Intelligence and Applications*, 158, 195-202.

Baker, R.S. (2007). Modeling and understanding students' off-task behavior in intelligent tutoring systems. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1059-1068.

Baker, R.S., Corbett, A.T., and Koedinger, K.R. (2004). Detecting Student Misuse of Intelligent Tutoring Systems. Proceedings of the 7th International Conference on Intelligent Tutoring Systems, 531-540.

Baker, R.S., Corbett, A.T., Koedinger, K.R., Evenson, S., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., & Beck, J.E. (2006). Adapting to when students game an intelligent tutoring system. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, 392-401.

Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. (2004). Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game the System". *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390

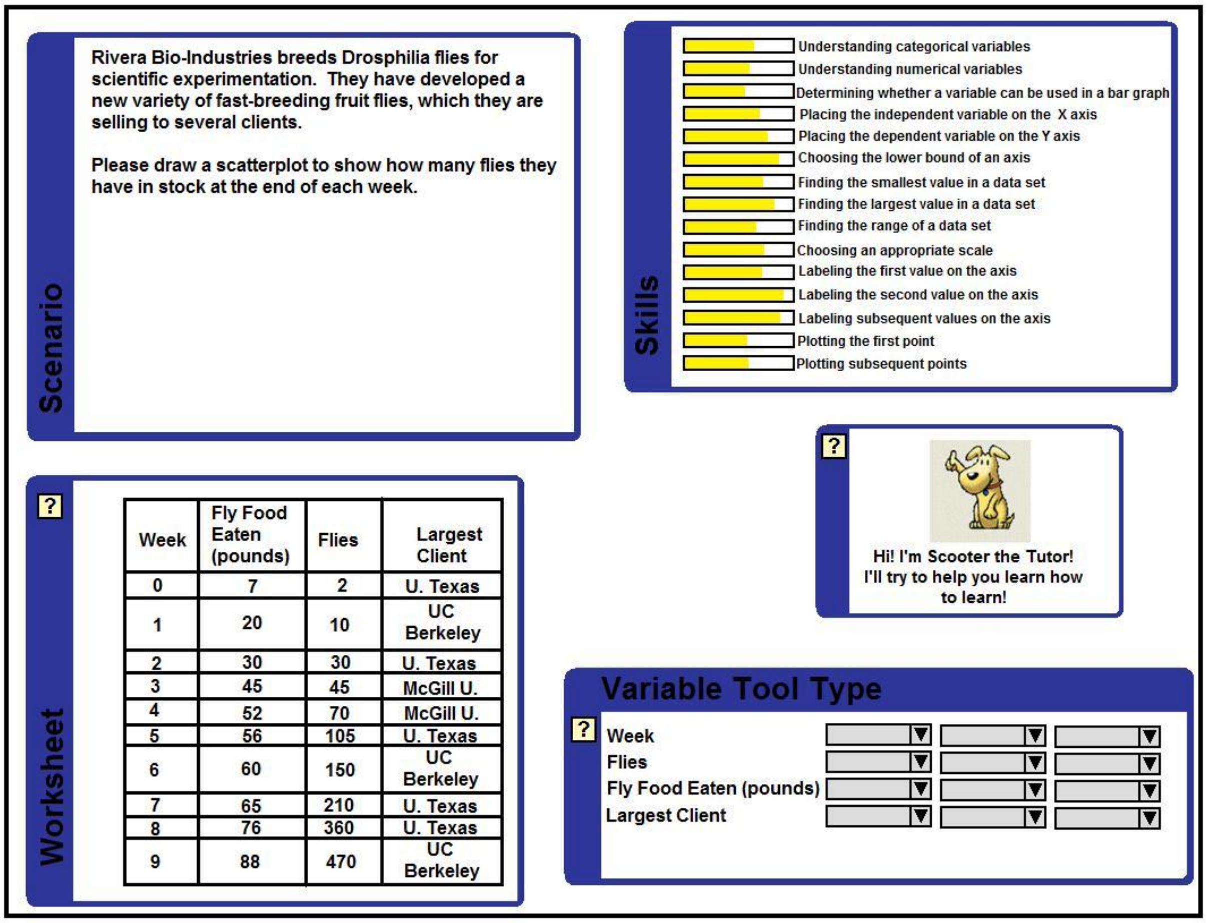
Pate-Clevenger, R., Dusing, J., Houck, P., Zuber, J. (2008). Improvement of off-task behavior of elementary and high school students through the use of cooperative learning strategies. Retrieved from ERIC database. (ED5000839)

Robison, J.L., McQuiggan, S.W., & Lester, J.C. (2009). Modeling task-based vs. affect-based feedback behavior in pedagogical agents: An inductive approach. *Proceedings of the 14th International Conference on Artificial Intelligence and Education*, 25-32.

Rowe, J.P., McQuiggan, S.W., Robison, J.L., & Lester, J.C. (2009). Off-task behavior in narrative-centered learning environments. *Proceedings of the 14th International Conference on Artificial Intelligence and Education*, 99-106.

Appendix A: Intervention Storyboards

A-1 Tutor Screen Layout



Rivera Bio-Industries breeds Drosphilia flies for scientific experimentation. They have developed a new variety of fast-breeding fruit flies, which they are selling to several clients.

Please draw a scatterplot to show how many flies they have in stock at the end of each week.

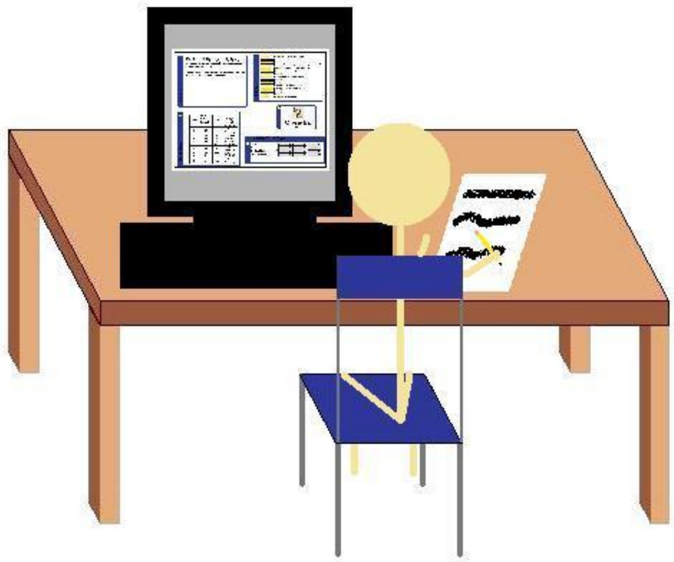
- Understanding categorical variables
- Understanding numerical variables
- Determining whether a variable can be used in a bar graph
- Placing the independent variable on the X axis
- Placing the dependent variable on the Y axis
- Choosing the lower bound of an axis
- Finding the smallest value in a data set
- Finding the largest value in a data set
- Finding the range of a data set
- Choosing an appropriate scale
- Labeling the first value on the axis
- Labeling the second value on the axis
- Labeling subsequent values on the axis
- Plotting the first point
- Plotting subsequent points

Hi! I'm Scooter the Tutor! I'll try to help you learn how to learn!

Week	Fly Food Eaten (pounds)	Flies	Largest Client
0	7	2	U. Texas
1	20	10	UC Berkeley
2	30	30	U. Texas
3	45	45	McGill U.
4	52	70	McGill U.
5	56	105	U. Texas
6	60	150	UC Berkeley
7	65	210	U. Texas
8	76	360	U. Texas
9	88	470	UC Berkeley

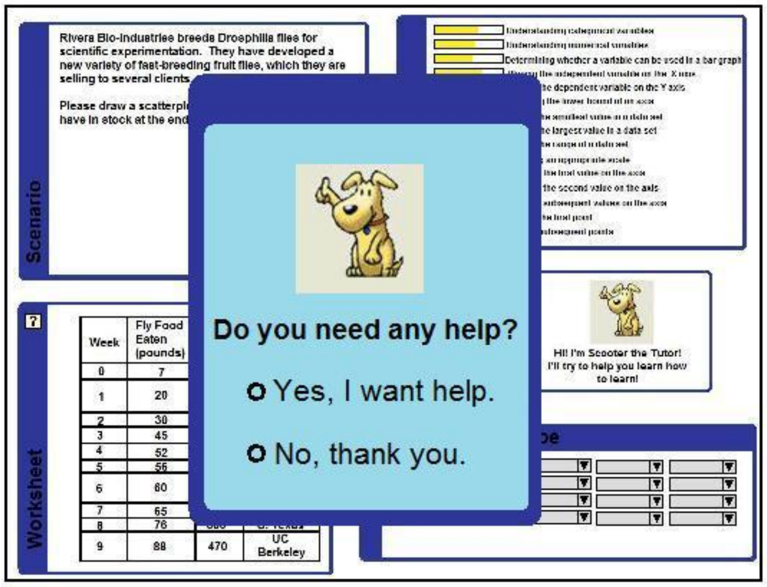
Week
Flies
Fly Food Eaten (pounds)
Largest Client

A-2 Scenario 1: A student who is on-task, but struggling



Scenario 1: Student is on-task, but struggling

Student is working with tutor and having a difficult time with a problem. They try to work out the problem on a seperate piece of paper.



Rivera Bio-Industries breeds Drosphilla flies for scientific experimentation. They have developed a new variety of fast-breeding fruit flies, which they are selling to several clients.

Please draw a scatterpl
have in stock at the end

Week	Fly Food Eaten (pounds)		
0	7		
1	20		
2	30		
3	45		
4	52		
5	56		
6	60		
7	65		
8	76		
9	88	470	UC Berkeley

Determining whether a variable can be used in a bar graph
the dependent variable on the Y axis
the largest value in a data set
the second value on the axis

Do you need any help?

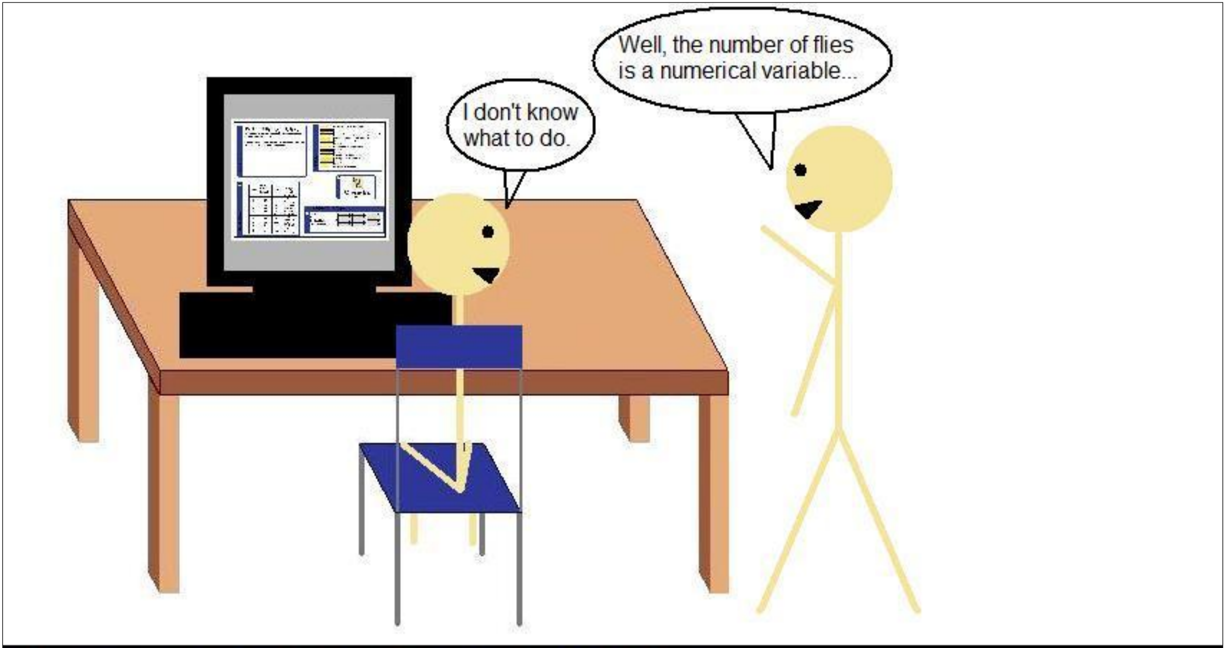
○ Yes, I want help.

○ No, thank you.

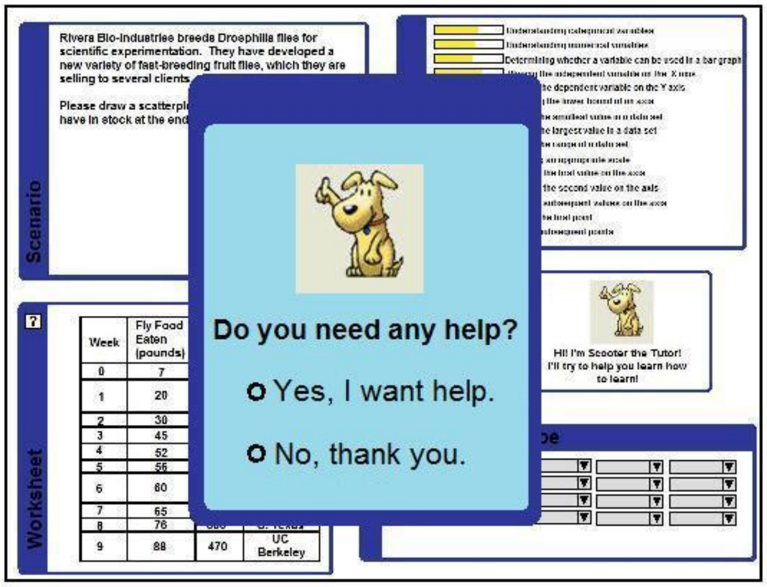
Hi! I'm Scooter the Tutor! I'll try to help you learn how to learn!

Tutor notices extended period of inactivity and asks if student needs any help. If student does need help they can choose the "Yes, I want help" option. The tutor will then provide them with a hint. After the hint there will be an option to ask for more help if the student still doesn't understand the problem. Otherwise they can choose the "No, thank you" option, and continue working on their own.

A-3 Scenario 2: A student is receiving help from their teacher



Scenario 2: Student is receiving help from their teacher
Student is working with tutor and having a difficult time with a problem. They ask their teacher for help.



Rivera Bio-Industries breeds Drosphilia flies for scientific experimentation. They have developed a new variety of fast-breeding fruit flies, which they are selling to several clients.

Please draw a scatterpl
have in stock at the end

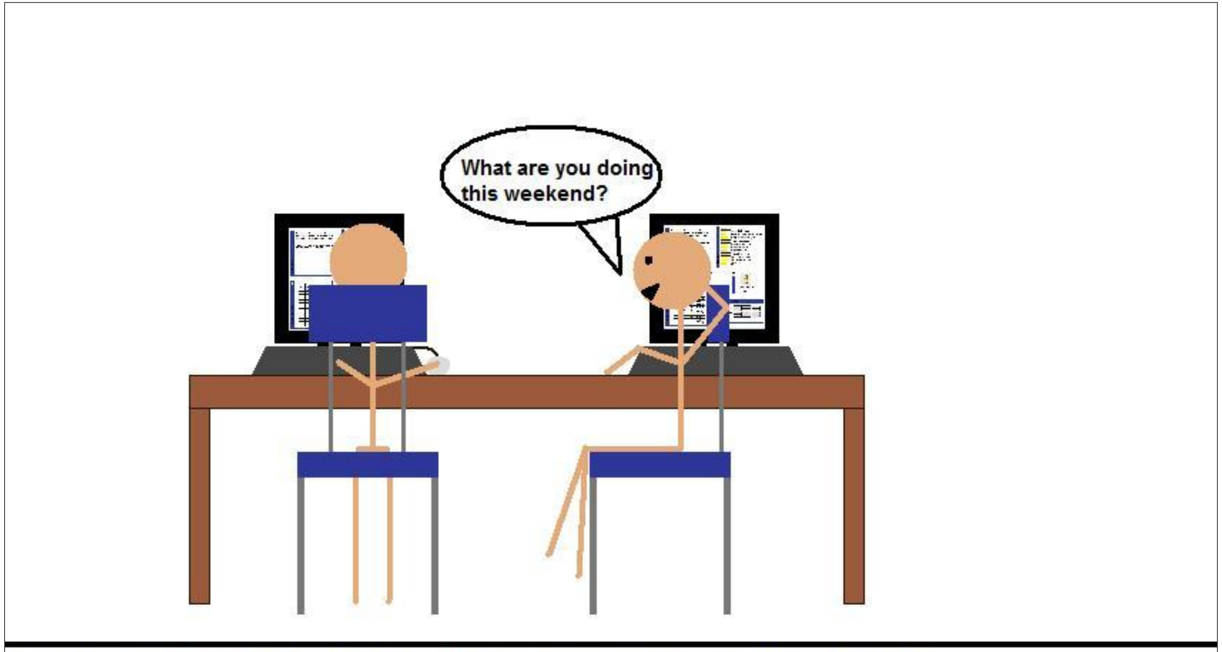
Understanding categorical variables
Understanding numerical variables
Determining whether a variable can be used in a bar graph
he dependent variable on the Y axis
he largest value in a data set
the second value on the axis

Hi! I'm Scooter the Tutor! I'll try to help you learn how to learn!

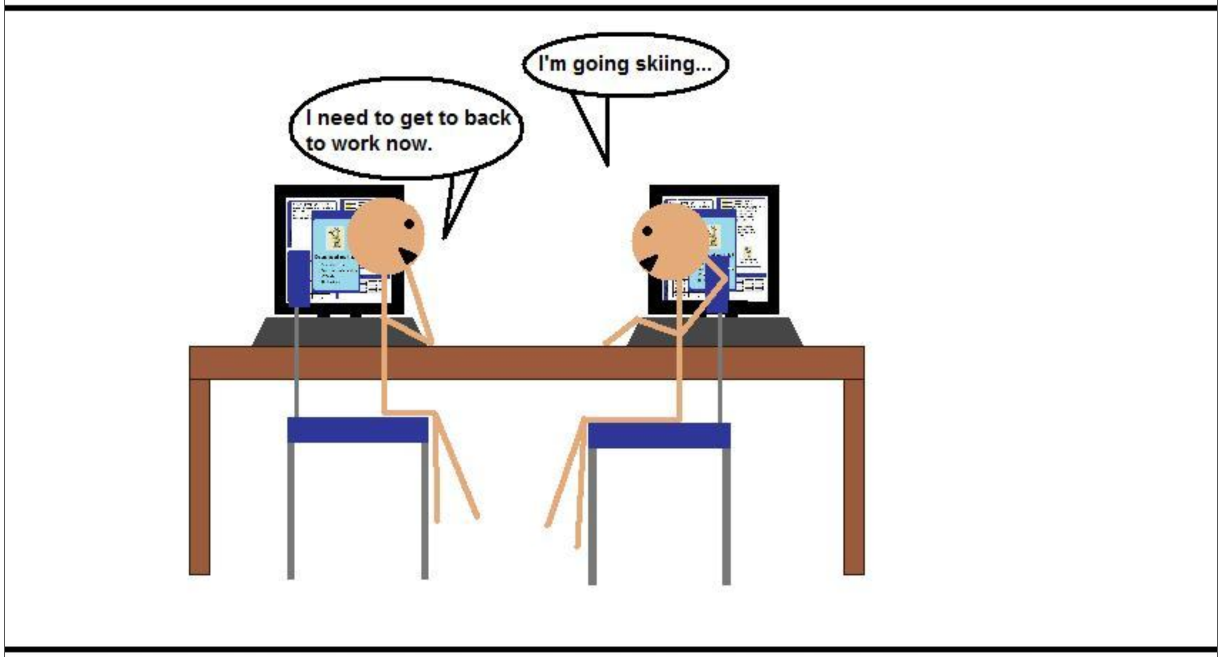
Week	Fly Food Eaten (pounds)		
0	7		
1	20		
2	30		
3	45		
4	52		
5	56		
6	60		
7	65		
8	76		
9	88	470	UC Berkeley

Tutor notices extended period of inactivity, and asks if student needs any help. When either the student or teacher notices the help window, they can select the "No, thank you" option, and the inactivity timer resets.

A-4 Scenario 3: An on-task student is being distracted by another student

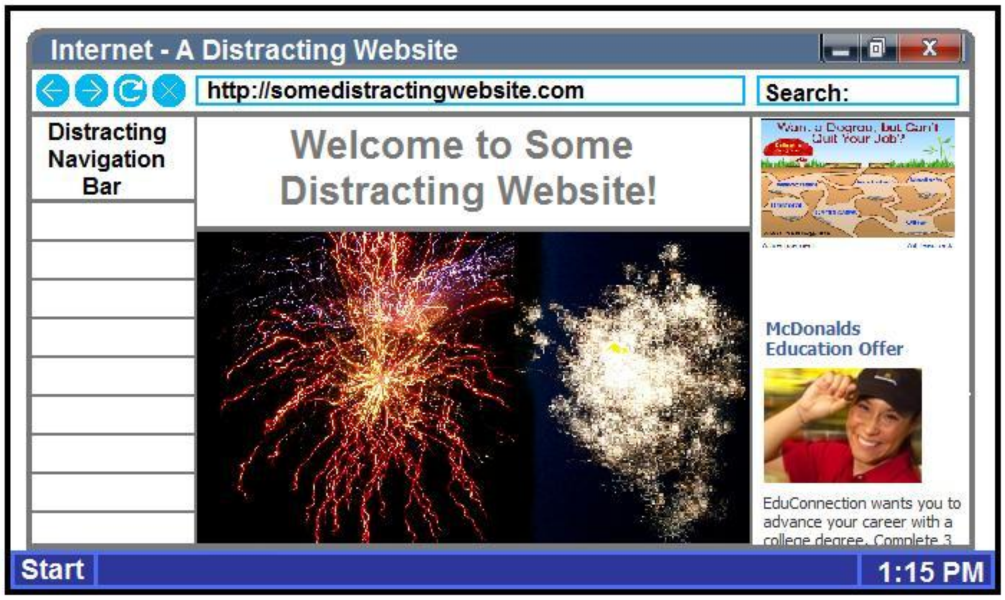


Scenario 3: An on-task student is being distracted by another student
A student is working with tutor and is interrupted by an off-task student.



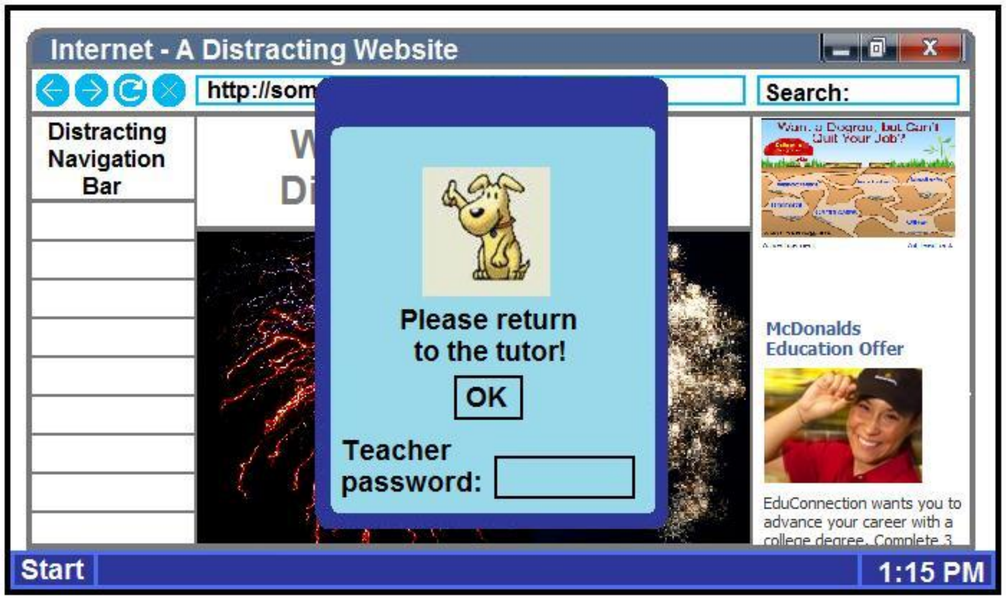
Tutor notices extended period of inactivity, and asks if student needs any help. Since the student was previously on-task, upon seeing the help pop-up, they remember that they were supposed to be working, and return to the tutor.

A-5 Scenario 4: A student is surfing the Web


Internet - A Distracting Website
http://somedistractingwebsite.com
Search:
Distracting Navigation Bar
Welcome to Some Distracting Website!
McDonalds Education Offer
EduConnection wants you to advance your career with a college degree. Complete 3
Start
1:15 PM

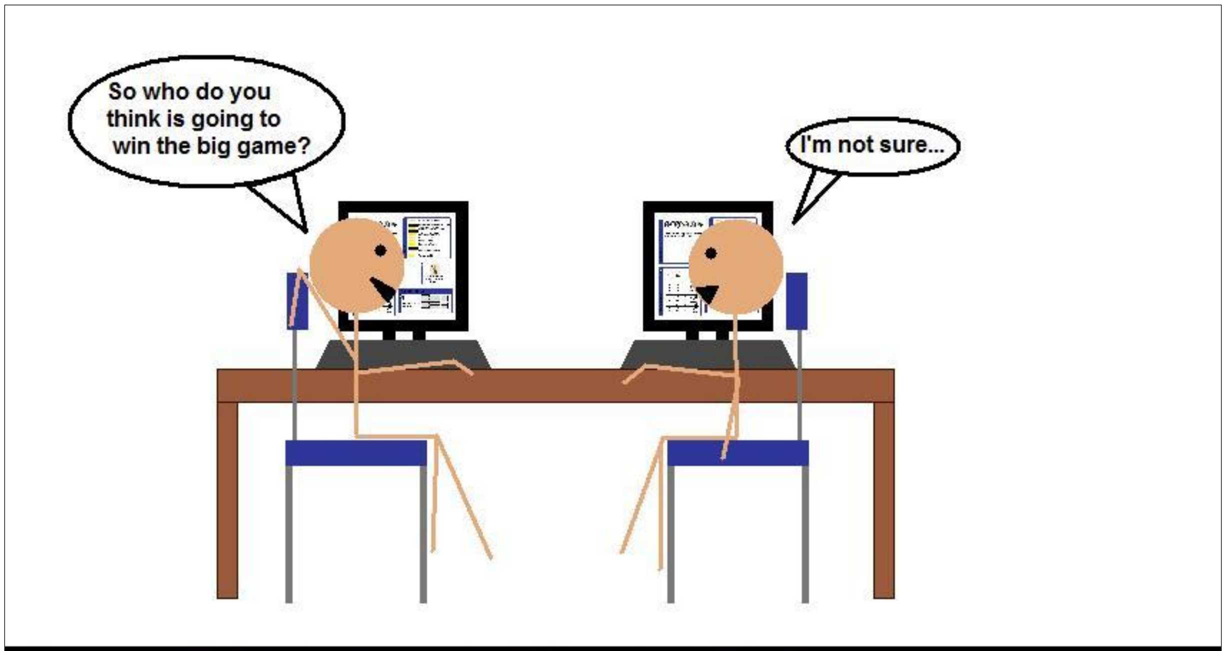
Scenario 4: Student is surfing the web

Student decides not to work with the tutor and instead starts surfing the web.


Internet - A Distracting Website
http://som
Search:
Distracting Navigation Bar
Please return to the tutor!
OK
Teacher password:
McDonalds Education Offer
EduConnection wants you to advance your career with a college degree. Complete 3
Start
1:15 PM

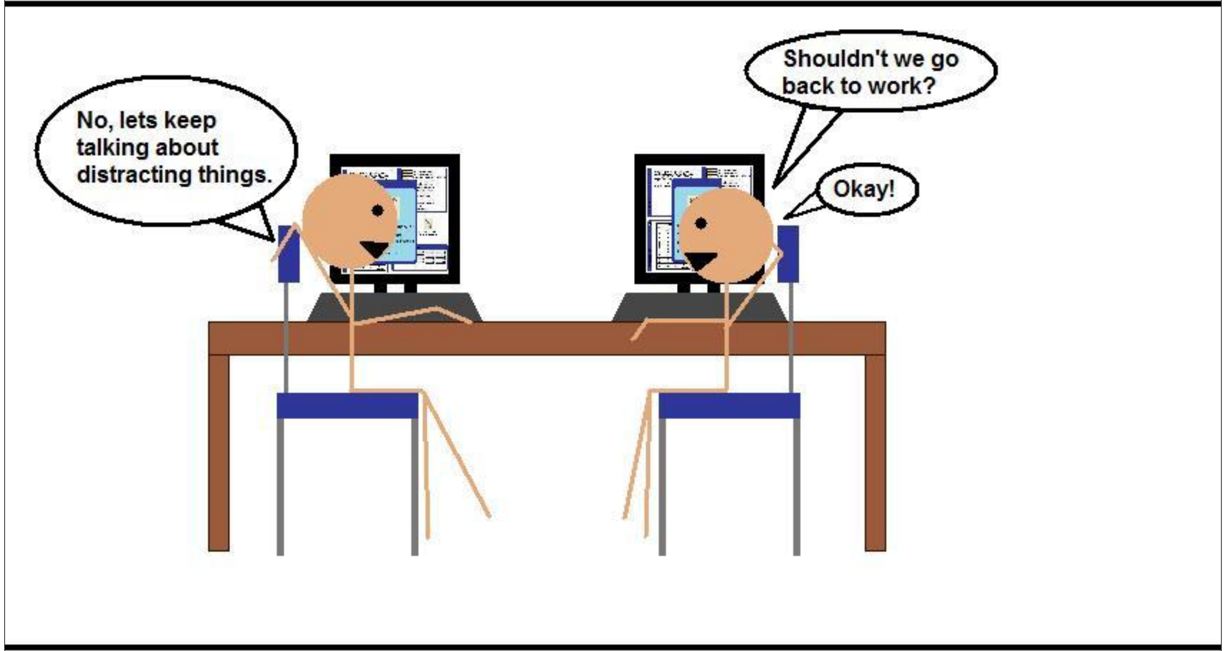
Tutor assistant pops up in front of web browser and asks student to return to the tutor program. If student has teacher permission to use the internet, they can ask their teacher to input a password. If they do not have permission, they can press "OK" and return to the tutor. If they do not return, the pop-up will reappear after one minute.

A-6 Scenario 5: A student who does not want to work with the tutor

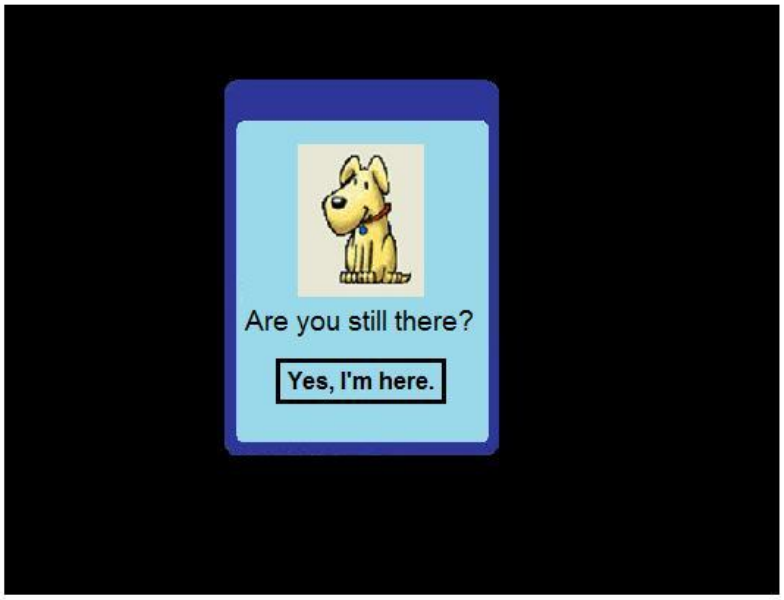


Scenario 5: Student who does not want to work with the tutor

A student who does not want to work with with the tutor goes off-task, and starts talking to another student.


No, lets keep talking about distracting things.
Shouldn't we go back to work?
Okay!

The tutor notices the extended period of inactivity, and asks if the student needs any help. Since this student does not want to do any work, they might either not notice the help box or could choose to ignore it.



If student does not respond to help assist within time limit, screen will turn black, leaving the assistant who will ask if student is still at their workstation. The screen will remain in this state until the "Yes, I'm here." button has been selected. The screen will then return to the tutor program which will still be offering assistance.