Too long to wait and not much to do: Modeling student behaviors while waiting for help in online office hours.

Zhikai Gao
North Carolina State University
zgao9@ncsu.edu

Collin Lynch
North Carolina State University
cflynch@ncsu.edu

Sarah Heckman
North Carolina State University
sarah_heckman@ncsu.edu

ABSTRACT

Promptly addressing students' help requests on their programming assignments has become more and more challenging in computer science education. Since the pandemic, most instructors use online office hours to answer questions. Prior studies have shown increased student participation with online office hours. This popularity has led to significantly longer wait times in the office hours queue, and various strategies for selecting the next student to help may impact wait time. For example, prioritizing students who have not been seen on the day of the deadline will extend the wait time for students who are frequently rejoining the queue. To better understand this problem, we explored students' behavior when they are waiting in the queue. We investigate the amount of time students are willing to wait in the queue by modeling the distribution of cancellation time. We find that after waiting for 49 minutes, most students will cancel their help request. Then, we looked at students' coding actions during the waiting period and found that only 21% of students have commits while waiting. Surprisingly, students who waited for hours did not commit their work for automated feedback. Our findings suggest that time in the queue should be considered in addition to other factors like last interaction when selecting the next student to help during office hours to minimize canceled interactions.

Keywords

Office Hours, Computer Science Education, Queue Management

1. INTRODUCTION

Computer Science has grown increasingly popular as a major [2]. As demand for CS education has grown so too has the size of the courses. As a consequence, many instructors are facing a seriously high demand for student help requests for their programming assignments. Coupled with this, the COVID-19 pandemic has prompted many instructors to move their office hours sessions online. While many institutions have returned to in-person courses they have continued with online office hours due to increased convenience and lingering safety concerns. Prior research has also shown that the online format increases students' participation in office hours [7]. This additional participation however, creates new challenges as instructors and TAs must manage longer hours and longer wait times. This situation forces the instructor to think more about how to distribute the office hour resources fairly. One popular strategy for selecting the next student to help is simply first-in-first out thus prioritizing students who have not received any help recently. This strategy ensures more students can get some help, however, it prolongs the wait time for some students. Usually, students are patient and willing to wait for help, but it is not realistic to expect students to wait for an hour or more on a regular basis. Therefore, finding a way to identify students who have waited "Long Enough" before running out of patience can be a help with queue management. Additionally, instructors should consider what are the students doing during the waiting period and how long have they waited as factors in that strategy. Encouraging students to keep working on the problem, try different approaches, and test their implementations are often considered helpful for progress. Separating students who cannot make progress without help from the students who just have a minor issue could optimize the selection strategy for instructors. In this study, we try to identify guidelines for office hour queue management by examining past student behavior when they are waiting for help during online office hours.

In this study, we aim to address the following research questions:

- RQ1: What is a tolerable amount of time for students to wait in the office hours queue before we expect them to give up?
- RQ2: While students are stuck in the queue, do they continue to make progress on their work?

In addressing RQ1 we will attempt to model the amount of time students are willing to wait through the wait time distribution of canceled requests. In addressing RQ2, we will investigate whether students are making any coding actions while waiting for help and consider factors related to such behavior that may support student success.

2. PRIOR WORK

Several research projects have been focusing on analyzing and categorizing what help is actually needed through some public forum (like a MOOC discussion forum or a Piazza thread). Xu et al. applied deep learning approaches to classify student question topics in MOOC discussion forums [17]. Because the nature of this help-seeking method is done by text, the content and the process of the help-seeking can be easily recorded. On the other hand, the private verbal discussions during office hours interaction are hard to track. Therefore, prior studies around office hours have been conducted by surveying students' or teachers' opinions of their office hours experience. This type of methodology has been used frequently to discover the under-utilization problem of office hours. On the other hand, studies about office hours analysis using the pure data-driven method with automatic data collection are quite rare. The usage of a ticketing system [14] does help with automatic data collection. Gao et al. [8], for instance, collected data through a ticketing system called My Digital Hand and categorized students' help requests by how well they described their problems. The results show that students mostly describe their problems with insufficient detail.

To understand why office hours are underutilized, many researchers try to analyze the factors behind students' office hours attendance and propose different ways to improve it. Smith et al. used a survey to investigate the reason why students are reluctant to use office hours and found that most students are unclear about how to use them correctly [15]. Ryan et al. also found that social competence plays a big part in whether students go to office hours, where students with weaker social skills tend to avoid help-seeking behaviors [13]. Macwilliam et al. changed their office hours place to a more social place and witness an increase in popularity[10].

Also, one of the biggest changes in how office hours operate since the COVID pandemic is that many instructors have moved their office hours online. Since then, researchers have been investigating the impact on students' behaviors. For instance, Gao et al.[7] compared the attendance of online and in-person office hours and also explored the relationship to students' characteristics when changing the format of office hours. They found that online office hours are generally more encouraging and popular, especially for students with low confidence and low enjoyment of computer science problem-solving. However, some researchers derived contrary conclusions. Malan et al. did an experiment [11] for Harvard's introductory CS course and found that the attendance in virtual office hours was similar to that of in-person office hour attendance.

Analyzing students' coding actions and behaviors are also a quite popular research topic. Erickson et al. used a set of progress indicators, previously designed by Li et al. [4], to evaluate how much progress students made in each coding commit during each phase of their projects [5]. Gitinabard et al. [9] categorized students' code commits based on the commit message and built a robust model to automatically classify them.

While student coding behavior and office hours usage has been well analyzed in those studies, researchers have rarely sought to link the two together nor have they investigated how the coding behavior changes during different periods of the help-seeking process. Therefore, in this work, we try to fill this gap.

3. DATASET

3.1 Course Information

Our data originates from the Fall 2020 semester of a CS2 course with 303 total students. Students are expected to finish 3 guided projects(GP1, GP2, and GP3), 2 projects(P1P2 and P2P2), and 12 labs in this course using Java. For all the coding assignments, students were required to follow a UML design that was provided by the course instructor. P2P2 is more challenging than P1P1; while the three Guided Projects are far more easier than the two projects.

The instructor used a Github[1] Enterprise server to manage students' code and progress for each assignment. Each repository is tied to a job on the Jenkins[2] continuous integration system. Whenever a student makes changes and pushes them to Github, Jenkins will pull the changes, compile their code, run static analysis tools (e.g. Checkstyle[3], PMD[4], SpotBugs[5]), testing the functionality of student's code through teaching staff test cases, and run student-written tests to collect coverage metrics. Depending on how far along in the project the student is, Jenkins will provide different types of feedback.

3.2 Office Hours Data

The course uses MyDigitalHand [14] (MDH) to facilitate office hour management. Students raise their hands by filling out a form indicating what they are working on, what they are stuck with, and what they have tried thus far. They are then placed in a queue, where the available teaching staff members pull from to help. After the request is over, both the teacher and student will be directed to a post-survey asking if they made any progress during the interaction. In Fall 2020, the office hours moved fully online due to the pandemic, where teachers host office hours through Zoom. In this format, the teacher can notify the students to find them in their Zoom room by clicking the "call the student" button on MDH, and start the interaction once the student arrives in Zoom. There were 16 members of the instructional staff who held office hours for at least two hours each week.

In this study, the data we collected includes the student request time, call time, start time, cancel time, and a complete time for each office hours interaction. We also collected the student's ticket content and their post-survey.

3.3 Commit Data

For this study, we also pulled data related to generic commit information, such as author, repository name, and timestamp. This commit data is collected from the Github server.

4. METHODS

4.1 Data Pre-processing

[1]`https://github.com/`
[2]`https://www.jenkins.io/`
[3]`https://checkstyle.sourceforge.io/`
[4]`https://pmd.github.io/`
[5]`https://spotbugs.github.io/`

Before we start the analysis, we first observed that a large number of help requests were completed within 1 minute (over 10%). According to the instructor, this abnormal spike of extremely short interaction is caused by two things. First, the TA will sometimes start interacting with a student but will forget to open the associated ticket on MDH, and they usually notice their mistake when the interaction ends, which results in opening and closing the ticket in the system before moving to the next student. And according to the teaching staff's report, this happens relatively frequently, especially since office hours moved online due to working with both MDH and Zoom. Second, there is no way to cancel requests from a student who has been called to receive help. If the student does not respond to a call to receive help, the teaching staff member may reopen and then cancel the ticket leading to a short interaction time.

Our prior work solves this problem by detecting the teacher's abnormal inactivity on the system[6]. Specifically, if a teacher did not open any tickets for at least five minutes while some students are still waiting in the queue, we assume the teacher was interacting with a student and just forgot to open it on MDH; in such cases, we would just simply move the next served interaction's start time accordingly until it satisfies the five-minute rule. After addressing the situation where the teacher forgets to open a ticket when the interaction starts, we marked all interactions last less than ten seconds as canceled so that the second situation in the last paragraph is also addressed.

4.2 RQ1: Tolerable Wait time

In RQ1, we focus on investigating how much time is a student willing to wait in the office hours queue. To achieve such a goal, we focus on the distribution of wait time for canceled tickets. We assume that if a student cancels their request after waiting for t minutes, then this student is only willing to wait for t minutes for that request. Based on this assumption, for any given time t, we calculate the probability of a canceled ticket's wait time being bigger than t to represent the probability of students willing to wait t time. In other words, we are modeling: $P(W_s > t|s \subset cancel)$, where s represent an interaction and W_s represents its wait time; in another word, we are calculating the Complementary Cumulative Distribution Function(CCDF) of the canceled interactions' wait time. Then we fit the resulting curve of this probability into a mathematical regression model using Levenberg-Marquardt algorithm [12] and calculate the R^2 score[3] to evaluate the accuracy of the resulting model. This resulting model can help us understand what is the probability that a student will keep waiting after t minutes, and the instructor can estimate the tolerable wait time for students based on this model. We recommend the instructor to pay attention to the value of t when the probability is 0.5 since after that students are more likely to leave the queue than stay waiting.

Besides the modeling process above, we also want to explore for different assignments, if students' tolerable amount of wait time varies. So we also group the interaction based on the corresponding assignment and calculate the $P(W_s > t|s \subset cancel)$ for each of them.

4.3 RQ2: Coding Action

The purpose of RQ2 is to explore the student's coding actions when they are waiting for help. Since the only coding action that is collectible is students' commit data, we use it as the only coding action in this study. We first build a Markov Chain model based on our data to illustrate how students' actions are transmitted when they are waiting. Specifically, all students start with a help request, then they can either cancel the request, start the interaction, or make a commit while waiting; if they make a commit, the next action could also be to either cancel the request, start the interaction, or making another commit while waiting. We calculate the probability of each type of transaction between those actions, which can give us a general idea about students' behavior during the waiting period.

Then we are also curious about when students make commits during the waiting period. So we define the time difference between the student's first commit during wait time and the request time as FC. For the interaction with no commits during the wait time, the FC sets as infinite. A smaller FC indicates the students commit very early when they wait for help. With this concept, we are able to measure: for any given time t minutes, if a student s has waited t minutes in the queue, the probability of they already made a commit. In another words, we try to calculate $P(FC_s < t|W_s > t)$. Please be aware that this probability is not a cumulative distribution function(this is not equal to $P(FC_s < t)$); meaning theoretically, this probability does not always necessarily increase when t increases. However, We believe this probability should generally grow as t increases, which means the longer the students wait, the more likely they will try something by themselves and make a commit for more feedback.

Moreover, we also analyzed whether the commit action was related to the assignment. Do students commit more frequently when seeking office hour help for different assignments? Also, since some assignments have much longer wait times, we would also like to see what is the percentage of students making commits in the first t minutes, where the t is decided by the result of RQ1.

Finally, we worked to tie the success of the interaction with the commit during wait period behavior. As part of this analysis we filtered out all the canceled requests in this part since the interaction does not exist for them. We measure the success of the interaction by the post-survey. After each interaction, the student can choose from four options to indicate how much help the interaction provides. This first option is *"I finished"*, indicating the problem is completely solved during the interaction; the second one is *"I'll be able to finish by myself"*, which indicates that the interaction points out how to solve the problem and students completely understand; the next option is *"I need more help, but I do make progress"*, which suggests that the instructor did not fully address their problem; And finally, the last option is *"I did not make any progress"*. As you can see, the first and second responses are positive reviews for the interaction outcome; while the last two are suggesting less successful interaction. We then calculate the percentage of each response for interactions with a wait commit and without a wait commit separately. Our initial hypothesis is interactions with one or more wait commits will have more positive success responses than interactions without any wait commit. We make this

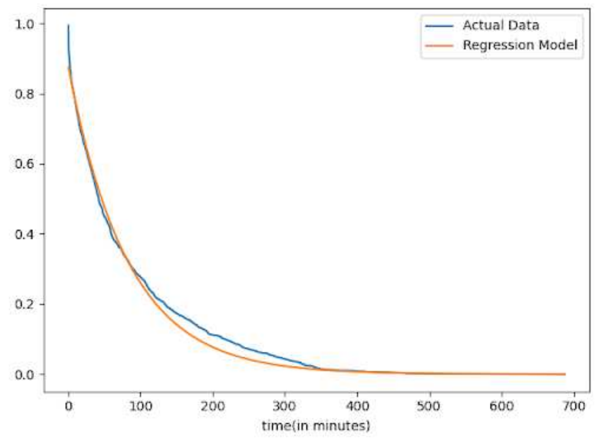

Figure 1: RQ1 modeling results

hypothesis because actively making commits could indicate that students are actively seeking answers to their questions, which might provide a better chance of finding or understanding the answer during the interaction.

5. RESULTS

5.1 RQ1 Results

The following equation is our resulting model of the odds that a student is willing to wait after t minutes:

$$P(W_s > t|s \subset cancel) = 0.875e^{-0.012t} \quad (1)$$

In Figure 1, it shows that the above regression model is very close to the actual data with a R^2 score of 0.992.

When $t = 49$, the probability drops to 0.5, so we know less than 50% of students are willing to keep waiting when they have been in the queue for 49 minutes. Therefore, 49 minutes is a reasonable time to separate long wait requests from regular ones. We therefore consider it to be a tolerable wait time in our course.

We also group the canceled requests by the assignment and see the distribution of their wait time. In Figure 2, we can see for Project 1 Part 2 (P1P2), students are willing to wait longer than with other assignments. Since Project 1 and Project 2 are the most complex assignment, and Project 1 happens before Project 2, this results corresponds with Ruth's et al. finding of students have higher help demands on more complex assignment and earlier assignment[1].

5.2 RQ2 Results

Figure 3 is the resulting Markov Chain model we generated from the data. We only witness 21% of students making a commit while waiting for help, the majority have no commits before the interaction starts (52%) or before they cancel the request(27%). These results tell us in general, actively making commits for more feedback is not a popular action when students wait in the office hours queue. Also, after made any commit during wait time, 71% of the students will made more commits before the interaction starts or is canceled.

Things get more interesting when we see how likely would students make any commit as they wait longer in the queue.

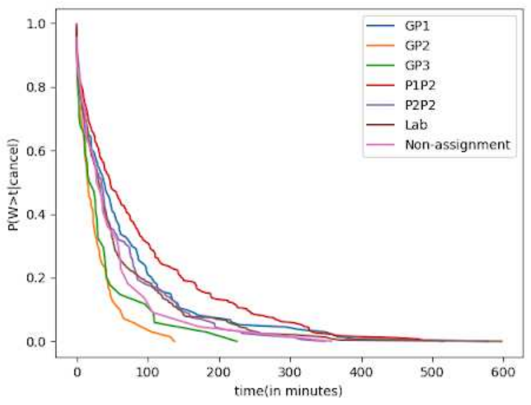

Figure 2: The CCDF of wait time distribution for canceled interaction in different assignments

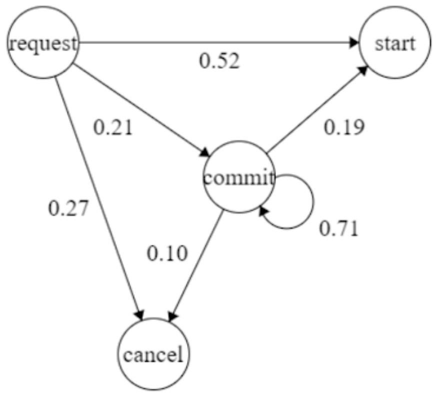
$$\begin{tikzcd}
\text{request} \arrow[rr, "0.52"] \arrow[dr, "0.21"] \arrow[ddr, "0.27"'] & & \text{start} \\
& \text{commit} \arrow[ur, "0.19"'] \arrow[loop right, "0.71"] \arrow[d, "0.10"] & \\
& \text{cancel} &
\end{tikzcd}$$

Figure 3: Action transmission during wait period

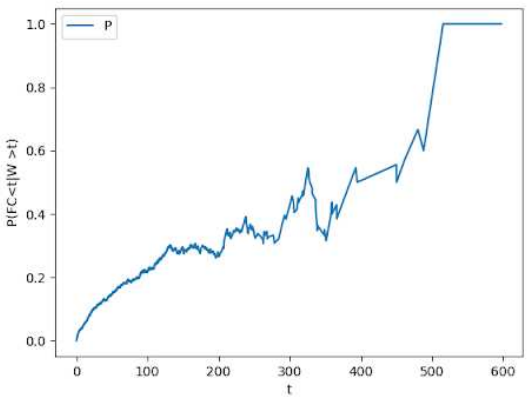

Figure 4: Probability of student made any commit within t minutes of waiting

	GP1	GP2	GP3	P1	P2	Labs
Total Request	409	252	180	879	598	812
Have wait commits	76(19%)	28(11%)	39(22%)	295 (34%)	204(34%)	192(24%)
Have wait commits within 49 minutes	53(12%)	24(9%)	33(18%)	129(15%)	96(16%)	116(14%)

Table 1: Relationships between assignment and committing during wait period

Student's response	I finished	Be able to finish on my own	Need more help	No progress	No Response	Total
No wait commits	394(24%)	290(18%)	257(16%)	26 (2%)	683(41%)	1650
Have wait commits	119(22%)	116(21%)	160(30%)	17(3%)	130(24%)	542

Table 2: Relationships between student's survey response and committing during wait period

In Figure 4, we can see as students wait for longer, they do tend to be more likely to make a commit, however, this increase is very slow. We can see for the first 2 hours, this probability grows steadily, then it stays around 0.3 until it changes drastically again around $t = 240$. The probability reaches 0.5 when t is 312 minutes. In RQ1, the results tell us that students waiting longer than 49 minutes likely already consider the request as a long wait time. Therefore, the result suggests that even when students are waiting a long time, the majority still do not make any commit. Another interesting thing to see is that around 350-400 minutes, the probability decreased greatly; which suggests that students who waited at least 6 hours in queue with no wait commit are also unlikely to commit during the 6-7 hours waiting period. Because the first office hours usually open at 10 AM, and if a student got in the queue around 10:30 AM, then 6-7 hours later they are very likely having dinner or on a commute; therefore the inactivity in that period corresponds with students' schedule. This theory also indicates that during busy office hours day, some students might get in the queue early just to reserve a spot; and during the long waiting period they are not just sit in front of the computer waiting for help, they might have other schedules to handle, but still didn't want to cancel the office hours requests, because they might need it later.

Table 1 shows there are significantly more interactions that have wait commits in the two major projects (p<0.05). However, for the first 49 minutes, we found no difference in the percentage of interactions that have wait commit among different assignments, which might suggest the longer wait times in the two projects is the main factor in having more wait commits.

The results that we observed with successful interactions were more surprising. In Table 2, we found students are much more likely to fill out the post-survey when they made commits during the wait time, Also, there are significantly more students who say they need more help after the interaction ($p = 0.013$ [16]), when they have a wait commit. This suggests that making a wait commit is somehow linked to less successful interactions, which is very counter-intuitive. One possible explanation could be when students encounter a very complicated problem, they might try a lot of different approaches when they are waiting, which means a high chance of a wait commit. However, complex problems also lower the chance of students fully understanding the answers or the hints the teaching staff provides, which results in a less positive review of the interaction outcome.

6. LIMITATION

Our data covers only one offering of a single CS2 course. Therefore the data is limited by that and may not generalize cleanly to other courses or institutions. In the future, we plan to include more data from different semesters to verify our results and findings.

Furthermore, the commit data does not give us a complete picture of students' coding actions. Offline code changes and recompilation is not included in our dataset and it is possible that students who are facing some types of problems will make changes but will not bother with a resubmit. Therefore, in RQ2, we cannot draw any precise conclusions on whether students are working on the problem while they are waiting for help. To solve this problem, we can collect students' local code changes through a plugin on their IDE. In this way, we can better investigate whether students work on their projects as they wait for help. However, that still cannot indicate whether the changes the students are making are for addressing the problem they are asking about during office hours. Students might work on a different part of the assignment before the question is answered in the interaction. Therefore, we still need to find a way to identify the intent of students' code changes and if they are associated with the help request.

7. DISCUSSION

In this study, we first successfully build a model for predicting how likely students are to leave the office hours queue after a certain amount of waiting time; our results show that 50% of students will leave the queue after 49 minutes of waiting. For the instructors, these results suggest that queue management strategies should consider wait time even for students who have already been helped that day. In RQ2, we found that only 21% of students were actively trying to make commits for more automated feedback on their problem, and students who waited for hours are still quite reluctant to make commits. This also indicates that instructors might need to encourage students to work on solving problems while they wait for help. Also, the strange link between commit behavior during wait time and unsuccessful interaction outcomes might suggest that the complexity of the student's question could factor in the student's problem-solving strategies and queue behavior. This should be examined more carefully in the future.

Back to our objective, what's the best selection strategies when the office hours resources are limited? This study provides a detailed analysis on the wait time factor for the instructor to consider. However, we still don't have a complete picture in what students are doing during the wait time. So for the next step, we should focus on identifying the most struggling student in the queue. Based on their detailed coding action, we could possibly detect which part of the code they are struggling with and evaluate the complexity of their potential questions. Not only can instructors have a better understanding of who needs the help most, but also be possible to build an automatic Q&A system to help the students with simple questions when they wait in the queue. With those theories and tools, we could witness a huge improvement of office hours efficiency in the future of CS education. Moreover, we could run simulations with different selection strategies, and compare the queue peformance(average wait time, percentage of canceled students, etc.) among those strategies to help instructors understand how they should use in their courses.

8. ACKNOWLEDGEMENT

This work is supported by NSF under grants # 1714538 (Co-PI: Collin F. Lynch, Tiffany Barnes, and Sarah Heckman).

9. REFERENCES

[1] R. O. Akintunde, S. Heckman, T. Barnes, and C. Lynch. Exploring class level forum usage for instructor insights.

[2] C. R. Association et al. Generation cs: Computer science undergraduate enrollments surge since 2006.(2017), 2017.

[3] A. C. Cameron and F. A. Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, 77(2):329–342, 1997.

[4] S. Edwards and Z. Li. Towards progress indicators for measuring student programming effort during solution development. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, page 31–40, New York, NY, USA, 2016. Association for Computing Machinery.

[5] B. Erickson, S. Heckman, and C. F. Lynch. Characterizing student development progress: Validating student adherence to project milestones. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 15–21, 2022.

[6] Z. Gao, B. Erickson, Y. Xu, C. Lynch, S. Heckman, and T. Barnes. You asked, now what? modeling students' help-seeking and coding actions from request to resolution. *Journal of Educational Data Mining*, 14(3):109–131, Dec. 2022.

[7] Z. Gao, S. Heckman, and C. Lynch. Who uses office hours? a comparison of in-person and virtual office hours utilization. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2022, page 300–306, New York, NY, USA, 2022. Association for Computing Machinery.

[8] Z. Gao, C. Lynch, S. Heckman, and T. Barnes. Automatically classifying student help requests: a multi-year analysis. In *Proceedings of The 14th International Conference on Educational Data Mining*, EDM '21, pages 81–92, 2021.

[9] N. Gitinabard, R. Okoilu, Y. Xu, S. Heckman, T. Barnes, and C. Lynch. Student teamwork on programming projects: What can github logs show us? *arXiv preprint arXiv:2008.11262*, 2020.

[10] T. M. MacWilliam and D. J. Malan. Scaling office hours: managing live q&a in large courses. 2012.

[11] D. J. Malan. Virtualizing office hours in cs 50. *SIGCSE Bull.*, 41(3):303–307, July 2009.

[12] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[13] A. M. Ryan and P. R. Pintrich. " should i ask for help?" the role of motivation and attitudes in adolescents' help seeking in math class. *Journal of educational psychology*, 89(2):329, 1997.

[14] A. J. Smith, K. E. Boyer, J. Forbes, S. Heckman, and K. Mayer-Patel. My digital hand: A tool for scaling up one-to-one peer teaching in support of computer science learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 549–554, New York, NY, USA, 2017. Association for Computing Machinery.

[15] M. Smith, Y. Chen, R. Berndtson, K. M. Burson, and W. Griffin. " office hours are kind of weird": Reclaiming a resource to foster student-faculty interaction. *InSight: A Journal of Scholarly Teaching*, 12:14–29, 2017.

[16] R. J. Tallarida and R. B. Murray. Chi-square test. In *Manual of pharmacologic calculations*, pages 140–142. Springer, 1987.

[17] Y. Xu and C. F. Lynch. What do you want? applying deep learning models to detect question topics in mooc forum posts?