Automatically classifying student help requests: a multi-year analysis

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

Tiffany Barnes
North Carolina State University
tmbarnes@ncsu.edu

ABSTRACT

As Computer Science has increased in popularity so too have class sizes and demands on faculty to provide support. It is therefore more important than ever for us to identify new ways to triage student questions, identify common problems, target students who need the most help, and better manage instructors' time. By analyzing interaction data from office hours we can identify common patterns, and help to guide future help-seeking. My Digital Hand (MDH) is an online ticketing system that allows students to post help requests, and for instructors to prioritize support and track common issues. In this research, we have collected and analyzed a corpus of student questions from across six semesters of a CS2 with a focus on object-oriented programming course [17]. As part of this work, we grouped the interactions into five categories, analyzed the distribution of help requests, balanced the categories by Synthetic Minority Oversampling Technique (SMOTE) , and trained an automatic classifier based upon LightGBM to automatically classify student requests. We found that over 69% of the questions were unclear or barely specified. We proved the stability of the model across semesters through leave one out cross-validation and the target model achieves an accuracy of 91.8%. Finally, we find that online office hours can provide more help for more students.

Keywords

Office Hour, Computer Science Education Research, Text Analysis, help-seeking request

1. INTRODUCTION

Over the past decade the popularity of CS majors has increased and enrollments have skyrocketed [2]. This has created challenges for instructors with increasing demands for individual support, collaborative learning, and automated guidance [12, 2, 16, 15]. As the size of courses and cohorts have increased, the demand for office hours has begun to exceed the time that instructors and staff have available [12, 13]. To address these needs instructors have adopted a wide range of innovative support models including virtual office hours [10], peer support [7], and ticketing systems for help-seeking interactions [13]. The last approach is exemplified by My Digital Hand (MDH) [21], an online support system for office hours which allows students to queue for office hours, post questions in advance, and record the outcome of interactions. MDH assists students in structuring their help-seeking interactions with teaching staff. It also assists instructors and teaching assistants (TA) in managing their courses, by allowing them to triage student questions and target their effort during office hours to be efficient and meet group and individual needs. MDH also tracks help-seeking and interaction data throughout the whole semester. Using this data, we can identify patterns in students' help requests and automatically classify the questions. One common challenge for help-seeking interaction on large classes arises when many students ask the same or similar questions but must be dealt with separately thus eating up limited instructor time. One approach to address this is to develop automated Q&A systems which can leverage common problems. In order for this to work however, students must provide sufficient information about their problems so that they can receive targeted support.

Our goal is to develop analytical methods to understand what kinds of help students seek during office hours, how they frame their questions to the instructors, and whether or not we can automatically classify questions to support guidance and time management. By analyzing students' help requests across course offerings we can better understand what kinds of challenges the students are facing, and how the teaching staff can better anticipate students' needs and target their limited support. Moreover, by automatically classifying help requests we can help teaching staff to efficiently triage student questions and identify common problems that may be solved with group support or peer assistance. Over the long term we will develop summary

Zhikai Gao, Collin Lynch, Sarah Heckman and Tiffany Barnes "Automatically classifying student help requests: a multi-year analysis". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 81-92. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

statistics which can be used to support instructors in course management, and we will augment our existing ticketing system with support for automatic categorization.

In this paper we will address four specific research questions in the context of a CS2, object-oriented-focused course:

- RQ1: What types of questions do students ask on MDH during office hours and how do they formulate their description?
- RQ2: How can we automatically classify student help requests and tickets?
- RQ3: How robust is our classification model across different offerings?
- RQ4: Compared to regular office hours, does online office hours provide more benefits?

In order to address RQ1 we analyzed our dataset to identify common patterns of student questions and to classify them into five categories. We then address RQ2 and RQ3 by training an automated classifier for student questions with the goal of evaluating its' stability across semesters. Due to the COVID-19 pandemic, all courses are operating online in Fall 2020, which gives us an opportunity to study the advantages and disadvantages of hosting office hour online. Therefore we analyzed and compared the data pattern on Fall 2020(F20) with other regular semesters in RQ4.

1.1 Background

Prior researchers have analyzed student help requests with the goal of understanding student behaviors. Xu and Lynch, for example applied deep learning approaches to classify student question topics in MOOC discussion forums [24]. In that work Xu and Lynch collected student posts from two offerings of a MOOC on Big Data in Education. The authors classified student questions into one of three types (Course Content Question, Technique Question, and Course Logic Question) and developed an automatic classifier using Recurrent Neural Networks to divide questions' into those three categories. While the models were successful within a single offering they Xu and Lynch, found that they did not generalize across offerings. Thus, the system suffered from a cold start problem on each semester.

Vellukunnel et al. in turn collected Piazza posts from CS2 courses offered at two institutions and analyzed the type and distribution of the questions students asked [22]. As part of this work they manually partitioned the questions into five categories and then analyzed the impact of students' question types on their final grades. They concluded that asking constructive questions can help students to develop a better understanding of the course materials and in turn receive better grades. This analysis has informed our own work. However the Piazza platform, unlike MDH, is designed to support interactive discussion and online peer support through the use of threads and replies. By contrast the MDH system is focused on initial help seeking and not on collaborative dialogue. Therefore it is unclear whether our results will align with theirs.

Prior researchers have also studied how instructors manage office hours and how to make face to face support time more efficient and effective. Guzdial, for example, argued that office hours should incorporate diverse teaching techniques including pair programming, peer instruction, and backward design. These approaches, he argued, would potentially work to reduce wait times and support enhanced learning outcomes [8]. In order to provide more convenience for students, Harvard University introduced virtual office hours to an introductory programming course CS50 so that students can interact with teaching staff online [14]. However, they found that those virtual sessions were often inefficient and took more time to address the students' problems. This research is complicated by the fact that students frequently avoid seeking help from teaching staff when they need it [1]. Some of the factors behind this help-avoidance include a lack of trust in the tutor's abilities, inaccessibility of office hours due to timing or other constraints, and a desire for independence in learning [18]. While our research provides some guidance on the design of office hours and the need to reach out to students, the impact of how students frame their help requests has not yet been analyzed extensively. One notable exception is the work of Ren, Krishnamurthi, and Fisler, who designed a survey-based method to help track the students' help-seeking interactions during office hours in programming-based CS courses [20]. While informative, their approach is difficult to generalize as it depends on requiring the teaching staff to complete a detailed form *after* every interaction. In MDH, by contrast, we collect much of the data upfront as an integral part of the process.

2. METHODS

2.1 MDH system

My Digital Hand (MDH) [21], is a ticketing system for office hours that was developed to facilitate large CS courses. Students using MDH request help during office hours by "raising a virtual hand", that is creating a ticket which lists the topic they need help on, describes the issue they are facing, and the steps they have taken to address it. Once the ticket is created it is visible to the teaching staff who can then use it to prioritize interactions or even group students together for help. Once the interaction is complete the teaching staff can close the ticket and describe how the interaction played out. Students are also given the opportunity to evaluate the help received. These feedback questions are configurable and set by instructors at the start of the semester.

This data allows instructors to identify common issues facing students and to track the time it takes for students to receive support from the teaching staff as well as the duration of each help session. A prior analysis of MDH data, Smith et al. found that 5% of students in a course accounted for 50% of office hour time, and that long individual interaction times, representing students who needed long and detailed guidance, served to delay many other short questions [21]. They concluded that a small but critical group of students are reliant on individual tutoring via office hours, while other students who need intermittent help are often unable to obtain support. These findings have motivated our own focus on developing analytical tools which can be used to analyze, prioritize, and manage help requests so that high-demand students do not shut out their peers.

2.2 Data Collection

We analyzed data from seven semesters of a typical second-semester Object-Oriented programming course [17] at a research-intensive public university in the south-eastern United States. Basic descriptive statistics for the dataset are shown in Table 1. Students produced an average of 1477 tickets per semester with higher volumes in the fall semesters due to larger class sizes. The number of tickets also increased year over year due in part to larger class sizes, greater emphasis on tool use by the course instructor, and higher per-capita demand for office hours.

The course is structured as a single lecture section with 12 small-group lab sessions which are held weekly. Over the course of the semester students complete weekly lab assignments, 2-3 individual or team Projects (C-Projects), and 3 separate Guided Projects (G-Projects). The G-Projects are designed to provide a review of prerequisite materials and introduce students to new concepts. The C-Projects are generally structured as two sub-assignments, one focused on design and system testing, and the other on implementation and unit testing. Students manage their code via the GitHub platform with integrated support for the Jenkins automation testing server. When students commit code they receive automated testing results from instructor-authored test cases as well as test cases that they supplied. The students use feedback from test failures to guide their work and their help-seeking.[6]

In Fall 2020, this course was moved fully online due to the pandemic. All office hours were hosted through zoom meeting where students can share their screen with the teaching staff to show their code or any problems.

Table 1: Number of tickets and students for each semester (F= Fall, S=Spring)

	F17	S18	F18	S19	F19	S20	F20
tickets	1146	609	1224	860	1650	1401	3452
students	208	157	259	174	256	191	303

The interaction data is the most important for our current analysis. The format of the interaction records, along with selected examples is shown in Table 2. For this analysis each ticket consists of three major parts: the participants, time and duration of interaction, and the context.

2.3 RQ1: Categorization of Questions

Our primary focus in RQ1 is to identify the types of questions the students are asking and to understand how they describe their work. MDH allows students to frame their question topic or description in any way that they wish. The platform does not provide a list of suggested topics or mandate content beyond the basic text. This, in turn, lead students to vary widely in the descriptions and content that they provide. We therefore studied two features of the questions with the goal of supporting classification, the students' topic, as contained in the `"I'm working on"` field. And the longer problem description, as stated in the `"my problem is"` field.

2.3.1 *Classified by Topic*

Table 2: Attributes of the interaction data

Attributes	Content Explain	Example
interaction id	Id for each ticket	30072
student id	Id for the student who raised this ticket	1950
teacher id	Id for teacher who deal with the ticket	20810
time raised hand	Timestamp for each tickets that are asked	2019-03-08 19:34:09
time interaction began	Timestamp for each tickets began	2019-03-08 19:38:39
time interaction ended	Timestamp for each tickets ended	2019-03-08 20:01:02
I'm working on	Topic for the question of each ticket	Program1Part1
my problem is	Detail statement for the question of each ticket	Null Pointer on TS test
I've tried	The solution the student tried before they raised the tickets	Debugging

Rapidly identifying, or even anticipating, students' question topics would allow teaching staff to anticipate the kinds of issues they should be prepared for and may also allow them to set up mini-groups within office hours to deal with problems assignment by assignment, or to separate code questions from conceptual ones. We therefore performed a manual analysis of the topics in our study dataset over all semesters with the goal of determining how students label their topics, and whether it is possible to either anticipate or sort their posts as they come in.

Our preliminary analysis showed that in most cases the students simply entered the name of their current assignment or an abbreviation of it and provided no other details. Moreover, due to the structure of the course deadlines almost every help request in a given session was focused on the same assignment. In the newest version of MDH, the question is now a check box and the instructor can set the assignments. As a consequence we decided to omit this from our classification task and focus on the types of help being sought.

2.3.2 *Classified by Description*

In the description section ("my problem is"), the students can provide a rich summary of their problem including a text description, bug reports, or even code snippets. If it is possible to automatically classify student posts then we can use that approach to triage student questions as they come in, perhaps separating long questions from short. We therefore performed a manual analysis of the description content as well with the goal of identifying useful categories of posts. We also sought to examine how complex the problem descriptions were. In our prior discussions with the teaching staff they reported that many students provide too little information in the description (e.g. a single word such as ”Errors”), provide too much (e.g. a full execution dump and error log), or they simply type gibberish with the simple goal of securing a place in line. All of these strategies are problematic either because they provide too little information to effectively triage posts, or because the dump is too complex and likely out of date before the student reaches the head of the line. In our analysis we analyzed both the length and structure of the students' submissions as well with the goal of understanding whether we can provide automatic scaffolding for useful posts, and automatic triage of the submissions.

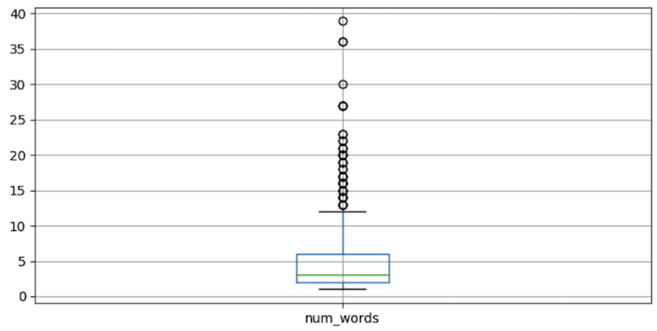


Figure 1: Distribution of the number of words in the questions' description

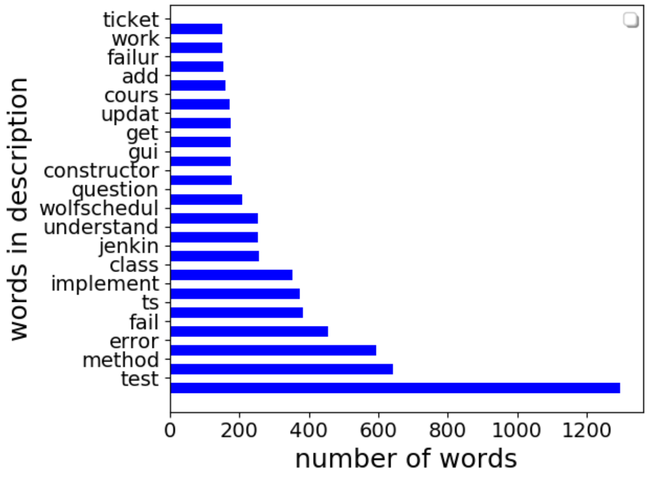


Figure 2: Top 20 Popular words in question description

We examined the length, content, and complexity of the students' problem description including the specific `my problem is` prompt. We also grouped common words to find keywords that are associated with specific types of submissions. We then used this information to inform our understanding of the students' posting behavior and to inform the design of the posting categories. This preliminary analysis was consistent with the experience described by the teaching staff. Figure 1 shows that the average description was less than five words long. When reviewing those short posts we noted that many students preferred to use keywords to indicate their question topics and problems rather than spelling their issues out. For example, when encountering an error in implementing an `add()` function, they often put "add()" as the problem description assuming that the method name, together with the assignment information, provided enough context for the help request.

This led us to focus on the specific terms that students use in their problem description. In this analysis, we grouped words by stemming and ignored stopwords to focus on the primary content information. The top 20 words are shown in Figure 2, top among them being `test`. This is consistent with the design of the assignments where students were provided with tests and required to develop their own. It is also consistent with the teaching staff's observation that many students focus on the tests as a guide for their progress and for where they need help. Upon closer examination of questions using this word we found that most interactions were focused on failed test cases; a typical description for a question of this type was "2 test case fail". It would be difficult for instructors to interpret these without reviewing the code and the test results in more detail but such a review takes time. Another closely-related word that was common in this dataset is `error` which was used primarily when students encounter bugs or other failures. In these cases in particular, the teaching staff noted that some students would simply paste the crash report into the question with little other context. This kind of behavior is rare in the data but was also useful for instructors, we therefore used it as an additional factor.

Based upon this preliminary analysis we defined five categories of help requests based upon the problem descriptions. These categories are shown in Table 3. We then labeled all interactions related to the problem manually. For each question, we also ranked the clarity or comprehensibility of student questions based upon the description provided. As we discuss below, most of the questions provided *insufficient* information to diagnose the problem. However as Figure 1 shows, some students did elaborate on their problem thoroughly as measured by the number of words in the problem description.

2.4 Labeling Process

2.4.1 Code book

To investigate the distribution of the above five categories in our data, we first need to set up a standard to categorize our data and apply it. All seven semesters' data was labeled by one researcher by the following rules:

- Check if there is any text that is clearly an error message copied from the compiler or a test failure. If so, label it as Copied Error. Notice that if the student describes the error message in their own words, then it should also be classified as Sufficient.

- Check if there is any text indicating that this is a test problem, no matter if the description gives you the detail of their test error or not. If you are sure that it is a test problem, label it as Test. If it also qualifies as Copied Error, classified as Copied Error

- If the text does not provide any information about their question and you cannot understand or deduce anything that related to their question, classify as Useless

Table 3: Explanation and example for all five categories we developed

Category	Explanation	Example description(my problem is)
Useless	The description contain nothing related to the question	I would like to check of it
Insufficient	The description contains partial information about the student question, but not enough for instructors to understand the details.	TicketManager getInstanceOf
Sufficient	Contains enough detail on the question for instructors to understand. Usually a very clear sentence.	CourseRecordIOTest, I seem to be failing reading the files, at the moments it is testing the size of the ArrayLists, but I am passing writing the files
Copied Error	Contains a copied error from the compiler.	I got this error: TypeError: barh() got multiple values for argument 'width'
Test	Test case fail related problem	2 test cases failed

- If the text does provide what or where the problem is, but not enough for you to fully understand or identify what is their question, marked as Insufficient
- If the text only contains one or multiple words of the method name associated with a problem, it can help to localize the problem but provides no additional details. It should be classified as Insufficient
- If the text is in a form of "I don't understand xxxx" without further explanation of which part they do not understand or other details, classify as Insufficient
- If the text is in a form of "I don't understand xxxx" with some further explanation, classify as Sufficient
- If the text tells you what their question or describes how they encounter this problem, classify as Sufficient

2.4.2 *Inter rater reliability*

After all data was labeled, we randomly generated a subset of 150 unique questions(30 for each category) and sent it to another researcher to rate. In this subset, we reveal the label of 10 questions for each category as example data and the rater classifies the remaining questions based on those example data and the code book. Then we compare the result with the original labels and calculate Kappa to represent the inter rater reliability. Kappa[4] is widely applied for measuring the agreement between two coders that accounts for chance agreement. Generally a score higher than 0.8 is considered acceptable. In our cases, the final unweighted Kappa value is 0.815 which is acceptable.

2.5 RQ2: Modeling

In addressing RQ2 we drew on our basic categorization developed in RQ1 to train automatic classifiers that can triage posts by topic and content. We used the first six semester data as training data to train our model and the last semester (F20) as the testing set to evaluate our model. To train our classification model, we first extracted training features from the problem descriptions across our dataset. The features included content features such as the keywords described above as well as meta-text features such as length, the number of stop-words (as a general proxy for specificity), the punctuation, and the character case. These meta-text features have the advantage that they are easy to extract automatically and can therefore be used for automated triage. Length, for example, is a suitable proxy for completeness and coherence while punctuation and case shifting are common in error messages. The full list of these features is shown in table 4.

We represented the text features as a tf-idf [19] matrix and basic word count matrix over the content. The word count matrix is simply a 2D Array which describe how many times each term appears in each question text. The tf–idf matrix is the product of two statistics, term frequency and inverse document frequency. The term frequency uses the raw count of a term in a text. The inverse document frequency is a measure of how much information the word provides. Some common words like "is" or "that" do not provide much information but they do usually have a high term frequency. Those words should have less inverse document frequency(idf). We can calculate the value as:

$$idf(t) = ln(\frac{Total\,number\,of\,documents}{Number\,of\,documents\,with\,term\,t\,in\,it}) \quad (1)$$

In our preliminary analysis we found that the matrices performed poorly in classification due to the fact that both were extremely sparse. We therefore opted to compress them so that they can be compatible with the dense feature approaches. To that end we built a Naive Bayes model [9] using the tf-idf sparse features and then use the predictions features. From this model we generated five shallow prediction features which correspond to the probability that the question belongs to each category. We followed this same approach with the word count vector and used those features as probabilities. The final list of extracted features is shown in Table 5.

2.5.1 *Model Training*

We trained our classification models using LightGBM [11], a Gradient Boosting Decision Tree (GBDT) algorithm provided by Microsoft. GBDT is an ensemble model of decision trees trained in sequence. In each iteration, GBDT learns the decision trees by fitting the negative gradients

Table 4: Text meta features list

Feature name	Explanation	value example
length	number of words in the problem description	12
character	number of characters in the problem description	12
stop words	number of stop words in the problem description	2
punctuation	number of punctuation in the problem description	0
uppercase	number of uppercase words in the problem description	0

Table 5: Text content features lists

Feature name	Explanation	value example
prob-tf-useless	the probability of this tickets belong to category ”useless” using tf-idf	0.75
prob-tf-ins	the probability of this tickets belong to category ”insufficient” using tf-idf	0.82
prob-tf-suf	the probability of this tickets belong to category ”sufficient” using tf-idf	0.77
prob-tf-error	the probability of this tickets belong to category ”copied error” using tf-idf	0.99
prob-tf-test	the probability of this tickets belong to category ”test” using tf-idf	0.65
prob-cnt-useless	the probability of this tickets belong to category ”useless”using using common word count	0.75
prob-cnt-ins	the probability of this tickets belong to category ”insufficient” using common word count	0.82
prob-cnt-suf	the probability of this tickets belong to category ”sufficient”	0.77
prob-cnt-error	the probability of this tickets belong to category ”copied error” common word count	0.99
prob-cnt-test	the probability of this tickets belong to category ”test” common word count	0.65

(also known as residual errors). To reduce the complexity of GBDT, LightGBM utilize two novel techniques to improve the algorithm: Gradient-based One-Side Sampling and Exclusive Feature Bundling. This method also utilizes a Leafwise Tree Growth algorithm to optimize the accuracy of the model and it applies a max depth of the trees to overcome the over-fitting problem that it might cause. Further, it optimizes the speed of training by calculating the gain for each split and uses histogram subtraction. LightGBM is known for its outstanding performance and relatively good speed. Thus, many researches applied this method to machine learning tasks.

The implementation code for LightGBM was provided by Microsoft[11] in 2013 and we are utilizing its Python library for modeling process. We applied features and the label of training data by LightGBM to train a model, and fit that model on the testing data to predict each question in those data. By calculating the accuracy of the prediction, we can evaluate the performance of this model. We ran a series of 20 preliminary experiments to explore the space of parameters before we settled on the values listed in Table 6. A list of crucial parameters people generally need to tune to improve classification model performance is also in Table 6. Since our goal is to achieve better Accuracy, we will tuning toward larger max_bin, smaller learning_rate with larger num_iterations, larger num_leaves and larger max_depth each experiment until the accuracy is not improving.

2.5.2 SMOTE

During the modeling process, another issue we faced is that the categories are highly imbalanced. Over half of the questions are in the *Insufficient* category and the *Copied Error* category contained fewer than one percent of questions. To address the problem, we applied SMOTE method which over-samples examples in the minority class. SMOTE [5], first selects one minority class instance at random, create a synthetic instance by choosing one of the k nearest neighbors at random and connecting those two instance to form a line segment in the feature space. We applied this method with k=5 and oversampling the data to generate the training datasets and testing datasets for further model training.

2.6 RQ3: Model stability over semesters

For a trained model to be useful however, it must be stable across semesters or else we suffer from a *cold-start problem* [3]. In order to assess the model stability we ran a series of experiments where we assessed the relative utility of the models by applying a leave-one-out validation strategy on a semester-by-semester basis. Showing that all models perform at a comparable level provides a strong indication that the models themselves are consistent and useful, even early in the semester.

2.7 RQ4: Online Office hour analysis

In Fall 2020, all the office hours were held online, which provided valuable data about online office hours interactions. We are very curious to analyze and see whether the students behavior changed with the move to online office hours and if we should keep some online office hours sessions once we resume in-person instruction.

We first analyzed whether the online session attracted more students to seek help during office hours. For an in-person session, students need to physically find the teaching staff in the office and physically stay in line. For online sessions, students only need to click the link to join the meeting with teaching staff. With online office hours, the friction of physically going to a campus location has been removed. However, online office hours have additional overhead in creating a connection between parties and transitioning between students. To better understand online office hour help-seeking, we calculated the average number of tickets per student and the percentage of students who used office hour in each semester and compared earlier semesters with in-person office hours to the Fall 2020 semester with online office hours.

Table 6: LightGBM common training parameters and the final optimal value after tuning our model

parameter	meaning	final optimal value
num_leaves	number of leaves in one tree	1000
max_depth	Specify the max depth to which tree will grow.	10
max_bin	max number of bins to bucket the feature values.	150
learning_rate	learning rate of gradient boost	0.1
num_iterations	number of boosting iterations to be performed	32
num_class	number of classes. used only for multi-class classification	5

Table 7: Distribution of labeled questions on those five categories in all seven semesters

Useless	Insufficient	Sufficient	Copied Error	Test
3.01%	69.02%	12.04%	0.10%	15.83%

However, online office hours could have an impact on the efficiency of communication. When teaching staff and students meet online, it creates more challenges for teaching staff to indicate problems to the students and to help explain why their code is failing. Screensharing allows the staff to view the students' work, but physical interactions like pointing to a portion of the screen to indicate which button to click is lost. The teaching staff member needs to verbally describe the debugging process and ask students to follow it. Therefore, we calculated the interaction time and the wait time for each ticket. The we compare the distribution of interaction time and wait time of F20 tickets with the rest of tickets. Additional overhead is incurred when connecting to a meeting. There is a lag when a student joins a meeting for their audio to set up to start the conversation.

3. RESULTS

3.1 RQ1: Categorization Results

Table 7 shows the distribution of question categories across our dataset. As the figure shows, the most common category is *Insufficient* which occupies over 69 percent of the questions. The *Test* category coming next at 15 percent. Approximately 12 percent of the questions belong to the *Sufficient* category while 3 percent were rated as *Useless*. Surprisingly, despite comments from the teaching staff, the least common category was *Copied Error* with at most 10-15 questions per semester falling into this group. As our results show, the students tended to use the system primarily as a way of getting in line and typically provided little useful information for the teaching staff. These results also highlight the significance of testing tasks for the assignments and for students' help-seeking given the high proportion of help tickets that are triggered by them.

To assess the stability of these results we also examined the frequencies within each semester. Figure 3 shows this breakdown. We found that the relative distribution is generally similar across semesters while the absolute percentages vary. In more recent semesters the students have authored more *Sufficient* tickets than in prior years suggesting that there has been greater effort by the instructional staff to encourage good communication. Yet the persistence of the other ticket type suggests that automatic classification and triage remain an important feature.

Table 8: Average interaction time(in minutes) and standard deviation of each semester

	Useless	Insufficient	Sufficient
AVG	19.7	21.9	18.3
STD	125.3	237.0	103.6
	Copied Error	Test	
AVG	11.5	24.8	
STD	67.5	208.2	

Table 8 shows the average and standard deviation of interaction time (the difference between when the interaction began and it was closed) of each category across the semesters. For this calculation we did not consider tickets with an interaction time less than 10 seconds in length or which were longer than one hour. Our discussion with teaching staff and the instructors showed that the former were cases that were never seen as the student set a placeholder but fixed their problem before their turn came up or changed their mind, while the latter represents cases where the teaching staff offered help but did not close the ticket, often until well after the tutoring session was over. The *Useless*, *Insufficient* and *Sufficient* categories averaged around twenty minutes in length with no meaningful difference in their interaction times. The *Copied Error* category was slightly shorter on average which may reflect the specificity of the students' problems while the *Test* category had a slightly longer average interaction time. This may indicate that this kind of question is more complex or more substantive relative to the others. Overall these results indicate that the amount of information provided does not necessarily affect the speed with which the issue can be addressed.

3.2 RQ2: Modeling Results

To evaluate the performance of our model, we trained the model using the first six semesters' data and tested it on Fall 20 data.The training dataset applied SMOTE method to oversampling the minority categories and result in each category having the same amount(5037) of questions in training dataset. The model achieved an overall accuracy of 91.8%. We then conducted a more detailed analysis of the performance for precision, recall, and F-score on each question type. The results are in Table 9. As our results show the model is relatively balanced across the categories with the exception of the Test category which had substantially higher precision and lower recall. This indicates that it was far more likely for other categories to be erroneously classi-

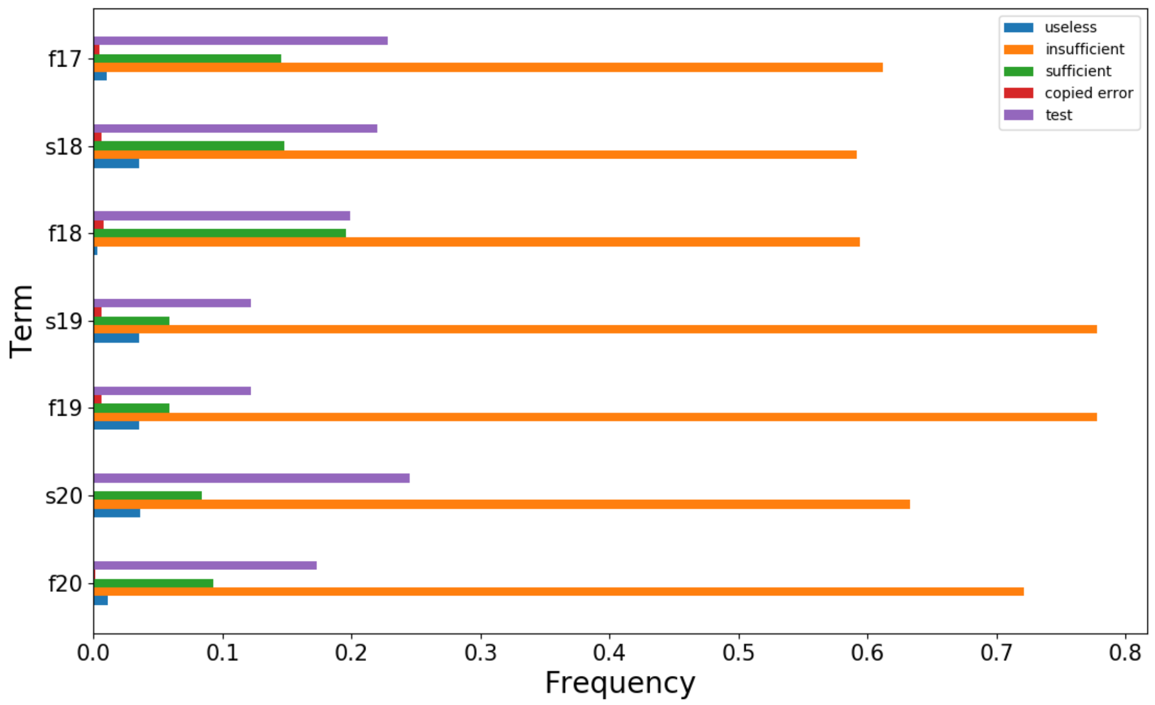


Figure 3: Frequency of each category for each semester

fied as *Tests* when the student submitted other information. One comment that triggered this error was: "Jenkins errors".

Table 9: Precision, Recall and F-measure for each category

	Useless	Insufficient	Sufficient	Test	AVG
Precision	0.902	0.915	0.913	0.967	0.922
Recall	0.901	0.903	0.905	0.877	0.901
F-measure	0.896	0.894	0.901	0.914	0.901

3.3 RQ3:Leave One out Results

Having shown that the model is relatively balanced across categories we then analyzed the relative accuracy of the model on each semester. The results are shown in Table 10. With the exception of Fall 2019, the model achieved an accuracy of at least 0.91 across each semester. Fall 2019 was the largest and busiest semester in our training dataset which may indicate that students were more diverse in their habits or posting behavior but even still we achieved an accuracy of 0.899. In light of these results we believe that our modeling method is sufficiently stable to assist in processing unseen semesters without a cold-start problem.

Table 10: Leave one out accuracy result

left-out semester	F17	S18	F18	S19	F19	S20
accuracy	0.913	0.915	0.908	0.907	0.899	0.912

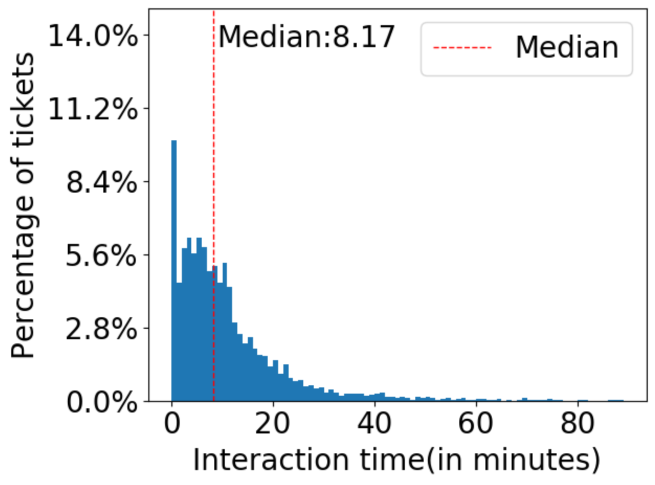
3.4 RQ4: Online Office Hour Analysis Results

Table 11 shows the various summary measures associated with office hours interactions for the semesters studied. The Fall 2020 semester had the highest number of average tickets per student and the largest percentages of students utilizing office hours. Additionally, the average tickets per student in Fall 2020 is nearly twice the average tickets per students in Fall 2019. This suggests that online office hours supports increased student participation in office hours interactions. However, this increment could be explained by other factors. The data shows that in general more students take advantage of the help each year. This is consistent with the increasing class sizes but it is important to note that there are other patterns as well such as regular dips in each spring. Overall it serves to highlight the need for better course management. Secondly, since the course lecture also holds online in F20, the increase of office hour usage supports a general expectation that students are facing additional challenges with online classes however we still believe that online office hours are a practical means to minimize the cost of help-seeking and thus encourage more students.

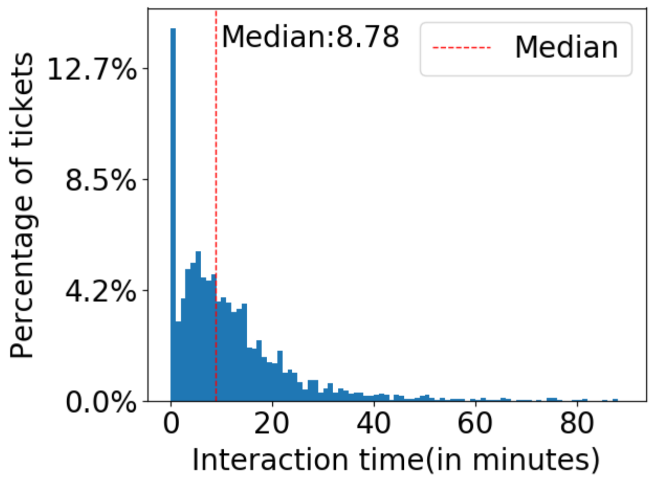
The distribution of interaction times shown in Figure 4 indicates that the teaching staff generally took slightly longer to support students in Fall 2020 than other semesters. The median interaction time of Fall 2020 is 8.78 minutes while other semesters are 8.17 minutes. We believe that this is caused by the inconvenience of remote instruction and debugging. Additionally, the high percentages of interactions within one minute in all semesters are usually caused by teaching staff forgetting to open the tickets when the interaction begins. In Fall 2020, the percentage of short tickets was much higher suggesting the teaching staff were more likely to make such mistakes because they are working with both MDH and the online interaction tool. After notify the student, the teaching staff has to waited in the zoom until

Table 11: Statistic Analysis for each semester

	F17	S18	F18	S19	F19	S20	F20
total tickets	1146	609	1224	860	1650	1401	3452
total students	208	157	259	174	256	191	303
students use office hour	104	63	141	84	158	108	209
average tickets per student	5.51	3.88	4.73	4.94	6.44	7.34	11.40
percentage of students using office hour	50.0%	40.1%	54.4%	48.3%	61.7%	56.5%	69.0%

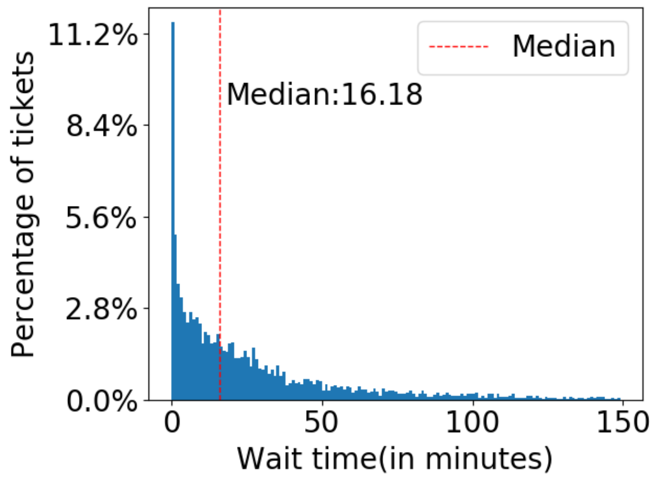


(a)

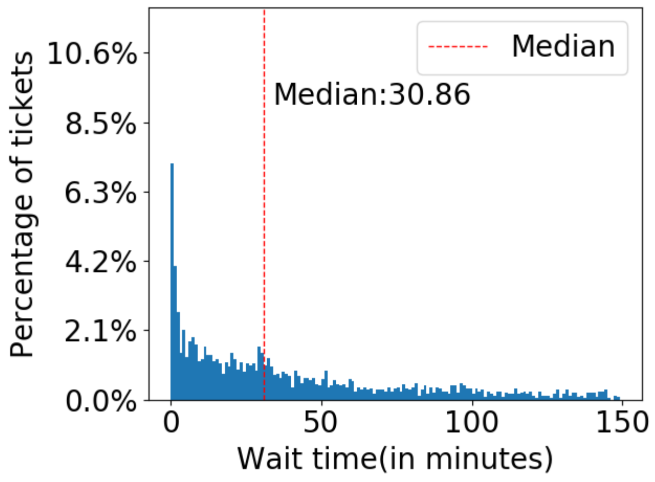


(b)

Figure 4: Histogram of Interaction time (time difference between open time and close time) for tickets in (a) Regular semesters and (b)Fall 20



(a)



(b)

Figure 5: Histogram of wait time (time difference between open time and raised time) for tickets in (a) Regular semesters and (b)Fall 20

the student actually join the zoom meeting to start the interaction. It is very common for the teaching staff just start the interaction in zoom without open ticket in MDH. The wait time in Fall 2020 is much longer than regular semesters, as shown in Figure 5. While we had 37 hours of regular office hour time each week for 17 TAs and 2 instructors, the increased demand in office hour help-seeking did have impact on student wait time and throughput. We additionally observed an increase in the number of canceled tickets. While online office hours lowers barriers for student attendance, additional resources to support demand are needed to ensure timely support. As we transition back to in-person instruction, the course instructor will continue to offer some online office hours to support access to help-seeking.

4. CONCLUSIONS

The results of our analysis in RQ1 show that students' use of the MDH platform does vary substantially from the developers' intentions. Far from using it to write complex help requests many do use it merely to reserve a place in line. Of the five categories, our labeling showed that most of the help tickets submitted lack sufficient detail to be clear what the student is asking about while those that do provide detail are most commonly focused on test cases which constitute a major feature of the class. Contrary to our initial expectations, the students rarely use the system to enter specific error messages, even if they have them. Thus the teaching staff have relatively little to go on when triaging questions. Clearly the proportion of useful information in the tickets increased in more recent years of the course but insufficient detail remains the most common feature. Despite this however, our results also show that there are clear categories of use that we can build upon to assist teaching staff. And our results show that it may be possible to extend the system with minimal automated interventions such as detectors for word counts or grammar that can be used to scaffold, or simply enforce, good posting behavior.

Informed by our analysis, we were able to address our modeling questions, RQ2 and RQ3 by developing accurate and robust classification models that achieved an overall accuracy of 92.6% and individual accuracy of 0.899% to 0.91% . Moreover, the results are robust on a per-category basis. While these results are not perfect, they show that we have the potential to use models of this type for effective triage of student questions as well as to provide scaffolding and immediate guidance for students as they author help tickets. While such guidance has not been evaluated for its' educational impact prior work on self-explanation (e.g. [23]) leads us to conclude that it may help students to diagnose their own challenges.

For RQ4, the comparison between an online session semester and regular semester shows both the strength and weakness of online office hours. The advantages of hosting office hours online is that it can encourage students to utilize the help-seeking resources; However, the large amount of help-seeking requests can be overloaded for teaching staff and the remote debugging through screen sharing is clearly less efficient than face-to-face interactions.

5. LIMITATIONS

There are several limitations to our work that must be acknowledged. While our results span semesters, they are still taken from a single course with a single instructor. As a consequence our results are necessarily dependant on the training that students have received and it is not yet clear whether this stability will be apparent in models created from interaction data for other courses, particularly those that are not as large, do not use the same assignment structure, or rely so heavily on tests.

Additionally, for our analysis toward online office hour, we did not consider the influence of teaching lectures online could raise more challenges for students and thus increase the usage of office hour. Our conclusion of online office hour encourage students to seek help is based on the assumption that there is no significant difference of academic difficulty between F20 and other semesters.

6. FUTURE WORK

This research can support future instructors in course management and the automatic categorization for MDH system. We therefore plan to address these limitations, expand our dataset, and build upon the models that we have obtained. First, we plan to conduct a more robust process of tagging and classifying our tickets with the goal of assessing the stability of our categories with other evaluators and of identifying other important ways of grouping the tickets themselves.

Second, we will extend My Digital Hand to take advantage of these trained models in supporting both the students and instructors. We will support instructors by providing automatic triage approaches that can help to guide their planning. And we will use automated guidance to prompt students to produce better tickets in the first place.

Third, we also plan to investigate other aspects of the office hours that are captured in the MDH data. These include: whether students in the same office hours post similar tickets, thus highlighting the potential of peer feedback; and the presence or absence of serial ticketers; that is students who keep multiple follow-up tickets going to monopolize support. We plan to build models for these features with the goal of understanding how help time is being used and by extension how to better coordinate limited support.

Finally, we plan to apply our models to provide automated scaffolding for students when they provide insufficient comments or errors. Specifically, we will integrate this model to the MDH system and every time a student raise a hand, we will use our model to predict the question category. If their description is insufficient or useless, then we can immediately notify them to revise it. This will help students to better frame their questions, and it will help the teaching staff can be better prepared to answer the students' question. This initial filter can be followed by additional models to suggest debugging steps or common answers based upon their revised question.

7. ACKNOWLEDGEMENTS

This research was supported by NSF #1821475 "Concert: Coordinating Educational Interactions for Student Engagement" Collin F. Lynch, Tiffany Barnes, and Sarah Heckman (Co-PIs).

References

[1] Vincent Aleven and Kenneth R Koedinger. “Limitations of student control: Do students know when they need help?” In: *International conference on intelligent tutoring systems*. Springer. 2000, pp. 292–303.

[2] Computing Research Association et al. *Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006.(2017)*. 2017.

[3] Tiffany Barnes and John C. Stamper. “Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data”. In: *Intelligent Tutoring Systems, 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008, Proceedings*. Ed. by Beverly Park Woolf et al. Vol. 5091. Lecture Notes in Computer Science. Springer, 2008, pp. 373–382. DOI: `10.1007/978-3-540-69132-7\_41`. URL: `https://doi.org/10.1007/978-3-540-69132-7%5C_41`.

[4] Kenneth J. Berry and Jr. Paul W. Mielke. “A Generalization of Cohen’s Kappa Agreement Measure to Interval Measurement and Multiple Raters”. In: *Educational and Psychological Measurement* 48.4 (1988), pp. 921–933. DOI: `10.1177/0013164488484007`.

[5] N. V. Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. ISSN: 1076-9757. DOI: `10.1613/jair.953`.

[6] *csc 216 software development fundamentals ,Engineering Online, NC state university*. June 2020. URL: `https://www.engineeringonline.ncsu.edu/course/csc-216-software-development-fundamentals/`.

[7] Zhijiang Dong, Cen Li, and Roland H. Untch. “Build Peer Support Network for CS2 Students”. In: *Proceedings of the 49th Annual Southeast Regional Conference*. ACM-SE ’11. Kennesaw, Georgia: Association for Computing Machinery, 2011, pp. 42–47. ISBN: 9781450306867. DOI: `10.1145/2016039.2016058`.

[8] Mark Guzdial. “Cutting the Wait for CS Advice”. In: *Commun. ACM* 62.8 (July 2019), pp. 12–13. ISSN: 0001-0782. DOI: `10.1145/3339456`.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

[10] Jeremy Johnson et al. “Virtual Office Hours Using TechTalk, a Web-Based Mathematical Collaboration Tool”. In: *Proceedings of the 6th Annual Conference on the Teaching of Computing and the 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education*. ITiCSE ’98. Dublin City Univ., Ireland: Association for Computing Machinery, 1998, pp. 130–133. ISBN: 1581130007. DOI: `10.1145/282991.283094`.

[11] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3146–3154. URL: `http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf`.

[12] Kevin Lin. “A Berkeley View of Teaching CS at Scale”. In: *arXiv preprint arXiv:2005.07081* (2020).

[13] Tommy MacWilliam and David J. Malan. “Scaling Office Hours: Managing Live Q&A in Large Courses”. In: *J. Comput. Sci. Coll.* 28.3 (Jan. 2013), pp. 94–101. ISSN: 1937-4771.

[14] David J. Malan. “Virtualizing Office Hours in CS 50”. In: *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’09. Paris, France: Association for Computing Machinery, 2009, pp. 303–307. ISBN: 9781605583815. DOI: `10.1145/1562877.1562969`.

[15] Engineering National Academies of Sciences, Medicine, et al. *Assessing and responding to the growth of computer science undergraduate enrollments*. National Academies Press, 2018.

[16] E. Patitsas, M. Craig, and S. Easterbrook. “How CS departments are managing the enrolment boom: Troubling implications for diversity”. In: *2016 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. 2016, pp. 1–2.

[17] Leo Porter et al. “Developing Course-Level Learning Goals for Basic Data Structures in CS2”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE ’18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 858–863. ISBN: 9781450351034. DOI: `10.1145/3159450.3159457`.

[18] Thomas W. Price et al. “Factors Influencing Students’ Help-Seeking Behavior While Programming with Human and Computer Tutors”. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER ’17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 127–135. ISBN: 9781450349680. DOI: `10.1145/3105726.3106179`.

[19] J. Ramos. “Using TF-IDF to Determine Word Relevance in Document Queries”. In: 2003.

[20] Yanyan Ren, Shriram Krishnamurthi, and Kathi Fisler. “What Help Do Students Seek in TA Office Hours?” In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. ICER ’19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 41–49. ISBN: 9781450361859. DOI: `10.1145/3291279.3339418`.

[21] Aaron J. Smith et al. “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. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 549–554. ISBN: 9781450346986. DOI: `10.1145/3017680.3017800`.

[22] Mickey Vellukunnel et al. “Deconstructing the Discussion Forum: Student Questions and Computer Science Learning”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 603–608. ISBN: 9781450346986. DOI: `10.1145/3017680.3017745`.

[23] Arto Vihavainen, Craig S. Miller, and Amber Settle. "Benefits of Self-Explanation in Introductory Programming". In: SIGCSE '15. Kansas City, Missouri, USA: Association for Computing Machinery, 2015, pp. 284–289. ISBN: 9781450329668. DOI: `10.1145/2676723.2677260`.

[24] Yiqiao Xu and Collin F Lynch. "What do you want? Applying deep learning models to detect question topics in MOOC forum posts?" In: