ABSTRACT

GAO, ZHIKAI. Automatically Classifying Real-time Course Questionson My Digital Hand . (Under the direction of Collin Lynch.)

Because of rising enrollments, Computer science faculty are facing higher demands for help and support. My Digital Hand (MDH) is an online tool to track office hour data that was developed to help instructors manage office hours and reduce the pressure on teaching staff. Since MDH tracks student questions and office hours, it provides a rich resource for analysis. By analyzing this data to identify patterns and classify help requests, we can better support instructors and make student support more efficient. The main goal of this research is to understand the structure and distribution of questions, comments, and feedback on My Digital Hand, and to develop automatic models that will support the classification of student help tickets. Our analysis and model can help the instructor to plan for future instructional and student support needs and assist MDH for automatic monitoring and feedback.

To achieve this goal, we analyzed the MDH interaction and feedback data from CSC216 in Fall 19 and Spring 19 to identify the types of questions that students were asking in the course. We grouped MDH interactions into five categories by how students describe their questions. We then labeled our data, performed feature engineering, and trained a suite of automatic classifiers to label student help requests. These classifiers relied on time, date, and text features of the students' descriptions and responses. We compared four classifiers: SVM, Logistic Regression, Random Forest and LightGBM. Overall, the LightGBM model performed the best with an accuracy of 0.94. These results suggest that it may be feasible to automatically classify student help requests in MDH to help MDH provide better service. In the future, we could use this classification to predict student's questions as they come in and give feedback to students if their problem description is not clear enough.

Automatically Classifying Real-time Course Questions
on My Digital Hand

by
Zhikai Gao

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2020

APPROVED BY:

Tiffany Barnes	Sarah Heckman

Collin Lynch
Chair of Advisory Committee

BIOGRAPHY

Zhikai Gao is a Computer Science Student with over 6 years of programming experience. He graduated and obtained his Bachelor's degree in Computer Science from Changchun University of Science and Technology in 2018. He is currently pursuing his Master's degree in Computer Science at NC State.

During his undergraduate years, he was a hardworking and efficient student who has a big passion for programming skills. He accomplished remarkable performance in Computer Science courses like C++ programming, data structure, and Algorithms. Namely, he was particularly good at applying some simple or complicated algorithms to solve multiple specific programming problems. For instance, he applied the Edmonds-Krap algorithm, an algorithm on solving the maximum flow problem, to simulate the traffic status under different road conditions.

He also was trained for the International Collegiate Programming Contest(ICPC) for three years. From 2014 till 2017, he spent at least 20 hours every week learning advanced algorithms and solving programming problems. He accumulated extensive experience in C++ programming and algorithm application. Also, he has participated in multiple programming contests and received several awards. He gained the bronze medal for ICPC Asia Regional Contest in 2016.

After He graduated from Changchun University of Science and Technology, he came to the U.S. and started to pursue his Master's degree in NC State. He learned some deep and extensive knowledge in Artificial Intelligence and Data Science, and receive experience in applying algorithms to solve problems with AI techniques. He applied the Monte Carlo Search Tree and Minimax algorithm by Python to develop an AI bot for Light Riders Games. Besides, he also joined the Arg Lab. Under the instruction of Dr. Collin Lynch, he focuses on educational data mining and relevant development.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Motivation

Over the past decade, consistent with the rapid development of computer science (CS), computer science introductory CS courses have become popular with increasing enrollments each semester [Zwe18]. This has created challenges for instructors with increasing demands for individual support, collaborative learning, and automated guidance. With large cohorts of students, the demand for office hours could exceed the time that instructors and staff have for support. In turn, this has led to a greater need to increase the efficiency of office hours and provide additional support opportunities. Instructors have adopted a wide range of support models including virtual office hours [Joh98], peer support [Don11], and ticketing systems for collaboration [Mac13] The latter approach is exemplified by My Digital Hand (MDH) [Smi17], an online support system for office hours which allows students to queue during office hours, post questions in advance, and record the outcome of interactions. MDH has been used in Computer Science courses at North Carolina State University and the University of North Carolina at Chapel Hill. This system assists students in scheduling meetings with TAs and receiving timely feedback, and it assists instructors by tracking all the question and interaction data throughout the whole semester. Using this data, we can identify patterns in students' help requests and automatically partition the questions into different categories.

The goal of this research is to understand the structure and distribution of questions, comments, and feedback on My Digital Hand, and to develop automatic models that will support the classification of student help tickets. Analyzing students' requests on MDH will allow us to achieve a better understanding of the challenges that students face in their courses, and it will help faculty plan for future instructional and student support needs. With real-time automatic classification

we can also help TAs to target their support to students with critical questions, and help them to identify common problems that may be solved with group support or peer assistance. Moreover, a better understanding of how students use the tool can lead to improvements in the design and operation of MDH for future courses. For example, our data analysis methods and results can be used to develop detailed summary statistics that can help instructors view the current course status. Additionally, the results of this research could be used to augment MDH with automatic monitoring and feedback.

The following three research questions guide this work:

1. R1: What data are recorded by MDH and how does that data reflect student and faculty behaviors?

2. R2: What types of questions do students ask on MDH and how are those questions distributed over the course?

3. R3: How can we automatically classify student help requests and tickets to better support effective teaching?

For the first research question R1 on data, we focus on describing and analyzing the MDH ticket data and identifying the behaviors those data patterns represent. For our second research question R2 on questions, we identify the most common pattern and features of student questions. Then we address this question by developing the method to divide those tickets into five categories, labeling all the questions we collected by this method and presenting their distribution over the course. For our last question R3 on request classification, we construct an automatic classifier based on the results of our first two analyses and using accuracy, precision, recall, and F-measure to evaluate it's performance.

The outline of this work is as follows. In the remainder of this chapter, I present background on the current work. Then in chapter 2 I discuss my analysis methods including the data evaluation and classification techniques used. This includes a description of the basic attributes of the MDH interaction data and briefly state how I analyze those attributes. Also, I briefly describe the five categories, feature engineering, and four modeling methods used for classification. Then I present the key results in chapter 3, before I present my overall analysis, conclusions, and future work in chapter 4.

1.2 Background and Related Work

In this section, I present prior research on analyzing student help questions, classifying help tickets, and some details about My Digital Hand. Research on student questions can give us a better understanding of the important features of student questions, and what type of questions students might ask in Computer Science courses. Research on help ticket classification provides some potential modeling methods to apply. It is important to explain My Digital Hand and its features so that we can apply the research to this specific context.

1.2.1 Analysis of Student Help-seeking

Analyzing the value of student questions is a common research goal. Mickey Vellukunnel [Vel17], for example, collected data from students' Piazza posts and analyzed what type of the questions students asked. They developed a categorization method and to separate questions into five categories. By labeling questions and comparing the grades of students asking different types of questions, they conclude that constructive questions can help students earn a better understanding of the course and receive a better grade.

Zhongxu Peddycord-Liu et al. [Liu16] analyzed the impact of national and cultural differences on students' performance in a Massive Open Online Course (MOOC). They found differences in three main aspects: course activity profiles; quiz activity profiles; and most connected forum peer. This research demonstrates the need to provide guidance and improve the experience for international students.

Yiqiao Xu [Wei17] applied the deep learning approaches like LSTM to detect question topics in MOOC posts. He proposed a classification method to divide questions into three types and developed his deep learning model based on these types. His final model can be used to help monitor MOOC student discussions efficiently.

1.2.2 Systems to classify help requests

This section presents research to develop models to classify help requests in other contexts. The requests may not be in educational contexts, but these studies may help identify the methods that can be used for text classification of help tickets.

Mucahit Altintas et al. [Alt14] focused on finding the right method to classify general issue tickets. They analyzed the structure of the ITU Issue Tracking System, including help tickets and labeled data, to build four classifiers on those data. They compared the performance and the speed of those models, and showed that the decision tree method has overall better performance and good speed than the other three methods (SVM, kNN, and Naive Bayes).

Feras Al-Hawari et al. [AH19] also applied SVM methods to develop a help desk system which can automatically classify tickets into thirteen different service types when they came in, based on the content of each post, and sent the ticket to the correct IT staff in an IT support system. The researchers used TF-IDF vectorization as a feature to train a SVM model. Although the final accuracy was only 81 percent, this work still helped the IT support service become more efficient.

Gwan Son et al. [Son14] also applied Naive Bayes and Softmax Regression Neural Networks to develop an automatic classification system on the XSEDE help ticket system. XSEDE is an online system for scientists to share computing resources, data and expertise. Compared to traditional SVM method, their method obtain a much better accuracy performance. They also discovered that the service provider resource (i.e., system name) information could significantly increase the performance to 90 percent.

1.2.3 My Digital Hand

My Digital Hand (MDH), [Smi17], was developed to provide tracking office hour interactions for large computer science courses for three universities in North Carolina. They collected interaction data in the MDH database and focused on analyzing the pattern behind those data. They conclude that students tend to wait more than an hour during busy periods; long interaction time affects the wait time greatly; and only 5 percent of students used 50 percent of office hour resources.

In MDH, students can "raise their virtual hand", creating a ticket when they have a question during office hours. As shown in Fig. 1.1, the question topic, description, and the tried solution are required in the ticket interface. After the student raises a ticket, the teacher can see all the tickets and accept them on the interface. When an interaction begins, the instructor can open or close a ticket to track its status. After the interaction, both the instructor and the student can evaluate the interaction by completing feedback questions in MDH. Feedback questions can be configured by the instructor at the beginning of the semester and can be configured for students or for instructors to be optional or required.

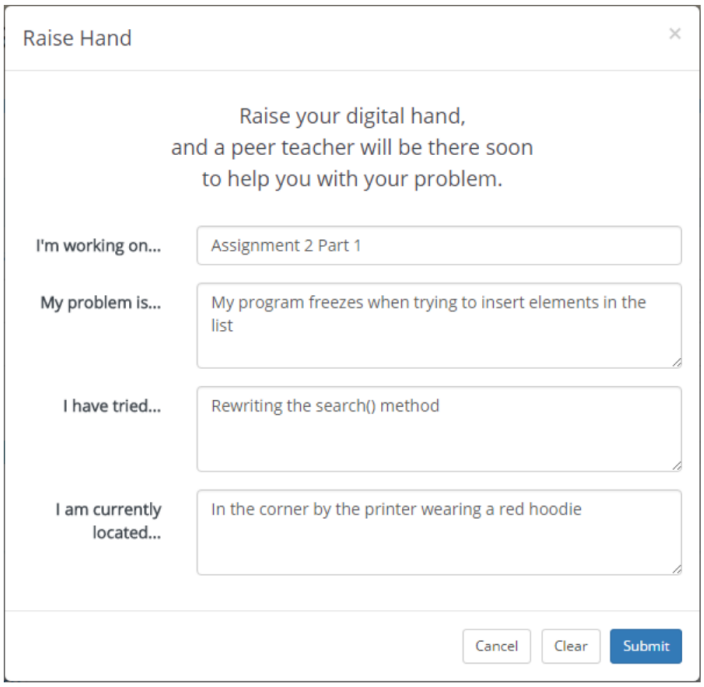


Figure 1.1 MDH interface for raising interaction tickets

Prior research has utilized the data collected from MDH and help instructors improve the efficiency on office hours. Jennifer Campbell et al. [Cam18] conceived that a drop-in help centre could greatly improve the student's experience of office hours. They developed two models based

on this theory: a drop-in centre open to students from all course levels and a drop-in centre open to only students from first-year courses. Then they collected the data from MDH and analyze the wait time, service time, and the busy course load to prove their theory.

CHAPTER

2

METHODS

2.1 Investigative Data Analysis

MDH produces two types of data files: interaction and feedback. By analyzing the patterns in these files, we can find the corresponding behavior of students and teachers to answer the first research question.

2.1.1 Introduction to CSC 216

In this research, we used MDH data from CSC 216 (Programming Concepts - Java) in Fall 2019 and Spring 2019. This course focuses on software engineering practice and skills. The course covers advanced Object Oriented programming, software engineering practices and the software development life cycle. Finite state machines, linear data structures and recursion are also introduced as crucial skills. There is also discussion on the performance and trade-offs of implementations of linear data structures.

There were a total of nine instructors (including the professor and teaching assistants) and 215 students enrolled in the Spring 19 semester. There were fifteen instructors and 263 students enrolled in the Fall 19 semester.

In table 2.1, it shows the number of ticket and number of feedback in each semester. It clearly suggests the data on Fall 19 have a larger quantity and more responsive feedback than on Spring 19. The number of tickets in Fall 19 is nearly twice than in the spring 19. Also, the number of feedback from students is extremely high in Fall 19, nearly all the students gave feedback to the system after the interaction, while in spring 19, only thirty percent of tickets got feedback from students. This

Table 2.1 Number of students, faculties, tickets and feedback for both Spring 19 and Fall 19 semester

	Spring 19	Fall 19
Total tickets	860	1650
Total feedback(teacher)	375 (43.6%)	499 (30.2%)
Total feedback(students)	364 (42.3%)	1597(92.8%)
Number of students	215	263
Number of Faculties	9	15

Table 2.2 Attributes of the interaction data

Attributes	Content Explain	Example
interaction id	The id for each ticket	30072
student id	The id for the student who raised this ticket	1950
teacher id	The id for teacher who deal with the ticket	20810
time raised hand	The timestamp for each tickets that are asked	2019-03-08 19:34:09
time interaction began	The timestamp for each tickets began	2019-03-08 19:38:39
time interaction ended	The timestamp for each tickets ended	2019-03-08 20:01:02
I'm working on	Topic for the question of each ticket	Program 1 Part 1
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

major difference caused by one change in the course policy. In Fall 19, the instructor required the students to give feedback after the interaction. Therefore, this policy makes sure that all the tickets got a response from the student and the data quality is much better than Spring 19. This difference is particularly noticeable given that the number of students was not substantially larger across the semesters (215 vs. 263) but the number of tickets nearly doubled (860 vs. 1650). While the relative feedback went down (43.6% to 30.2%) the rate of the student response more than doubled (42.3% to 92.8%). This suggests that the students were better trained and far more motivated to give feedback even as the TAs continued to struggle with the volume.

2.1.2 Interaction data

The most important of data is the interaction file, which contains all the information about each ticket. The format and contents of the interaction file are shown in Table 2.2 Operationally, a ticket consists of three major parts: participants, time, and the context. The *participants* includes the id of teacher and the student; The *time* describes when the ticket gets raised, opened and closed; And the *context* part maintains the information about the topic, problem statement, students' tried solution,

and suggestion.

The raised time is the timestamp when the students requested help by raising their digital hand; the open time is the timestamp when the teaching staff member opened the interaction and started helping the student; the close time is the timestamp when the teaching staff member closed the interaction. Among the three, We used the open time to analyze the distribution of the tickets and reveal when students are more likely to use office hours. We also calculated the duration of each ticket which indicates how long it was live and thus how long each interaction lasted. In other words, the duration time equals to close time minus start time. We calculated the wait time by open time minus raised time which indicates how long the ticket was open before it was raised.

When filing a ticket, students are required to write down the topic of their question, the detailed description of their question, and any solution they already tried when they raised a ticket. We focused our analysis on determining what content students put in the topic and which words are popular when they try to describe their question.

In addition to the features we could derive directly from the data, we noticed one interesting pattern, which is that many students re-posted the same question within a short time. We therefore added an additional feature to reflect repeat questions when students posted an identical question within thirty minutes after the prior one was closed. We worked to analyze the nature of those questions and exhibit the duration time of the repeated questions.

2.1.3 Feedback Data

The feedback data contains the evaluation from both the students and teachers for each ticket session. Currently, there are three questions. All the feedback data is optional, so around 40 percent of the tickets had corresponding feedback. However, note that the instructor required the students to give feedback in the Fall 19 course. In some semesters, instructors are required to leave feedback as well. The first feedback question is asking the teacher what students were asking about. 2.3 shows the answer to this question. Based on this table, most of the tickets are about an assignment, which corresponds with the interaction data.

Table 2.3 Distribution of answers to the first feedback question and the average wait time and duration time in minutes for corresponding tickets

Spring 19	About an assignment	479	0:21:00	0:09:05
	About a concept or exam question	10	0:04:51	0:04:53
	About a class or logistics question	6	0:14:17	0:06:39
	About something else	3	0:13:12	0:08:27
	Unclear	1	0:00:21	0:02:57
semester	answer	total	avg wait time	avg duration time
Fall 19	About an assignment	366	0:22:19	0:08:30
	About a concept or exam question	3	0:05:23	0:05:52
	About a class or logistics question	6	0:07:19	0:10:47

The second feedback question is asking the member of the teaching staff if the student's question was resolved. Table 2.4 summarizes the answers to this question. Most of the questions did get resolved during the ticket session, which might imply that the office hours are an efficient way to help students.

Table 2.4 Distribution of answers to the second feedback question and the average wait time and duration time in minutes for corresponding tickets.

semester	answer	total	avg wait time	avg duration time
Spring 19	Yes	310	0:22:53	0:08:08
	Partially	62	0:17:25	0:10:15
	No	3	0:19:30	0:12:26
Fall 19	Yes	422	0:20:30	0:08:23
	Partially	66	0:20:34	0:12:06
	No	11	0:21:00	0:12:06

The last feedback question is to students on whether their problem got resolved during the session. 2.5 describes the metrics for this question.

Table 2.5 Distribution of student answers to the third feedback question and the average wait time and duration time in minutes for corresponding tickets

semester	answer	time selected	average waiting time	average interacting time
Spring 19	Yes	329	0:20:37	0:09:10
	Partially	29	0:20:50	0:13:13
	No	6	0:37:30	0:10:54
Fall 19	Yes	1335	0:25:34	0:17:00
	Partially	205	0:30:39	0:53:33
	No	57	0:40:04	0:11:29

2.2 Question Categories

This section focus on the results of analyzing the students' questions, developing a method to divide the questions into five categories, and labeling questions based on that method. By showing the distribution of different categories, we can have a better understanding about what type of questions students are asking. This section provide the methods to address the second research question.

Initially, we noticed that many students like to use keywords for their problems. For example, when they encounter an error when implementing an `add()` function, they often put the text "Add()" in the problem description, while others will elaborate on their problem thoroughly, so the readers can clearly understand the problem. Finally, a few students, perhaps just seeking a

Table 2.6 Explanation and example for all five categories we developed

Category	Explanation	Example
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

quick response, provided irrelevant information in the description. Among these three, the most obvious difference is how much information they provide in the description. Therefore, our initial categorization began with this feature. Then, at the content level, a large number of questions are about failed test cases; typically, these questions might be described as "2 test case fail". It would be difficult for instructors to interpret these without reviewing the code and the test result. So, I think for this course, test-related questions might be a big issue that every student might encounter, and this it may be worth using failed test cases as an individual category. Moreover, a few students copy and paste the error message they encounter in the description to elaborate the bugs they are working on. This kind of behavior is rare in the data, but it may be more useful for instructors, so I analyze it separately.

We identified five different categories based on how students describe their problems. 2.6 explains the five different types.

2.3 Classification and Modeling

In this section, we introduced the methods we use for feature selection and classification modeling. It clearly present the methods to address the third research.

2.3.1 Classification by topic

Classification by topic is relatively easy, since 99.4 percent of the tickets contained only the name of the assignment. We only had to clean the data by emitting blanks and transfer uppercase to

lowercase and then merging common abbreviations. After cleaning, the content should be identical if belong to the same assignment. Then for rest of the unique value, we merge them to the closest ticket's topic based on interaction time.

2.3.2 Classification by description

The classification by description is the main goal of our research and address the last research question. To do this we labeled the all the data manually by the five categories we mentioned previously. And using interaction data from both of the semesters to train a Machine learning model which can automatically classify any new tickets.

The train and test are split by ratio of 0.8 in both of the semester for the training data. We considered three parts for feature selection: text meta features, date related features and text content features. For the text meta features, we calculate the number of words, number of stop words (words that is meaningless), number of punctuation, number of uppercase words, and number of characters. For date related features we identified the features of, the duration time, wait time and separated date information (for instance, day of week, week of year) are extracted.

For text content features, we began by extracting a tf-idf matrix over the content. However, since the tf-idf output is a sparse matrix and we need to find a way to adapt with other dense features; therefore, we built a Naive Bayes model using just the tf-idf sparse features and then use the predictions features.Therefore, five features are generated by this prediction and it represents the probability of this question belongs to each category. For instance, if a ticket has a problem description like "test case fail", the features we generate in this ticket would be [0.1,0.3,0.07,0.03,0.5]. Those features can represent as the probability of this ticket is a useless question is 0.2, and the probabilities of this tickets is a insufficient question, sufficient question, copied error question, or test question are 0.3, 0.07, 0.03, or 0.5 respectively. Finally, we treat those possibilities as new features. Also, similarly, we use a simple word count vector to do the same operation above to generate the the probability of belonging to each category for each ticket and use those probabilities as features.

We omitted the feedback data from our classification for two reasons. First, because our mission is to classify those questions as they came in and feedback would be unhelpful there as it came in when the interaction is closing and the participants fill in the feedback. Second, since the feedback is optional, there are a lot of missing value in our data set.

All the features can be found in 2.7 and 2.8.

After the feature selection, we use four different methods to train the model: Logistic Regression, Random Forest, LightGBM and SVM.

The Logistic Regression method [Pen02] is a typical Machine Learning model for classification tasks. Generally, the idea is to use the sigmoid function to convert the linear regression method from continuous variable prediction (Regression) to discrete variable prediction (Classification). It has been widely used in classification tasks with good results. For instance, Liguo Wang et al. [Wan13] applied the logistic regression method to developed a system called Coding Potential Assessment Tool (CPAT), which can recognize coding and noncoding transcripts from a large pool of candidates.

Table 2.7 Date related feature and text meta features list

Feature name	Explain	value example	range
month	the month when the ticket start	3	1-12
day	the day when the ticket start	24	1-31
hour	the hour when the ticket start	17	0-24
minute	the minute of week when the ticket start	30	0-60
day of week	the day of the week when the ticket start	2	1-7
duration time	end time minus start time	5	positive number
wait time	start time minus raise time	5	positive number
length	number of words in the problem description	12	positive number
character	number of characters in the problem description	12	positive number
stop words	number of stop words in the problem description	2	non negative number
punctuation	number of punctuation in the problem description	0	non negative number
uppercase	number of uppercase words in the problem description	0	non negative number

Table 2.8 Text content features lists

Feature name	Explain	value example	range
prob-tf-useless	the probability of this tickets belong to category "useless" using tf-idf	0.75	0-1
prob-tf-ins	the probability of this tickets belong to category "insufficient" using tf-idf	0.82	0-1
prob-tf-suf	the probability of this tickets belong to category "sufficient" using tf-idf	0.77	0-1
prob-tf-error	the probability of this tickets belong to category "copied error" using tf-idf	0.99	0-1
prob-tf-test	the probability of this tickets belong to category "test" using tf-idf	0.65	0-1
prob-cnt-useless	the probability of this tickets belong to category "useless"using using common word count	0.75	0-1
prob-cnt-ins	the probability of this tickets belong to category "insufficient" using common word count	0.82	0-1
prob-cnt-suf	the probability of this tickets belong to category "sufficient"	0.77	0-1
prob-cnt-error	the probability of this tickets belong to category "copied error" common word count	0.99	0-1
prob-cnt-test	the probability of this tickets belong to category "test" common word count	0.65	0-1

Also, it is very easy to use since many open source libraries like Sklearn [Ped11] contain this method. Therefore, we are using this method as an option.

Random Forests [Bre01] are also very popular in classification tasks. This methods contains multiple decision trees and it uses bootstrapping to generate sample. Then, randomly selecting features for each sample and training a decision tree and using the majority of the output from all those decision trees as the final result. The method also has a wide application in building classification models and the performance is very good. Like the Logistic Regression, this method also has been included in many open source libraries. Therefore, we chose to use this method as a potential model.

Like the previous methods, support vector machine (SVM) is also very popular in building classifiers. SVM [Cri99] is a machine learning method which use some kernel to map the nonlinear data into a linear space and thus easily use a hyper-plane to divide the data into different categories. In chapter one, nearly all the research of the help ticket classification used this method. For instance, Feras Al-Hawari et al. [AH19] applied the SVM method on automatic classification of IT support ticket. Therefore, SVM is using as an option for our classification model.

Finally, just like Random Forest, LightGBM is also a decision tree model. LightGBM is a Gradient Boosting Decision Tree algorithm provided by Microsoft [Ke17]. This method utilizes the Leaf-wise Tree Growth algorithm to optimize the accuracy and applied the max depth of the trees to overcome the over-fitting problem it might cause. Also, it optimizes the speed of training by calculating the gain for each split and uses histogram subtraction. It is known for its outstanding performance and relatively good speed. Thus, many researches applied this method to machine learning tasks. For instance, Dehua Wang et al. [Wan17] applied this method to train a model that can automatic classify the miRNA in breast cancer patients with high accuracy.

Also, Naive Bayes method are used to deal with the spare features of tf-idf. Naive Bayes is a conditional probability model. It utilise the Bayes' theorem to calculate the conditional probability of each class and pick the highest one as output. [Has09] It is a very simple machine learning methods and commonly used for text classification. Therefore, we utilized this method for processing spare feature.

In our modeling process, we were struggling about whether to use the id of students and teachers as features. On one hand, using ID-feature could boost the accuracy since it could detect the specific individual style on language development among different students. On the other hand, the id feature could cause the model over-fitting since in different semesters, the students enrolled will be entirely different. Thus, the model generate with previous student's id as one feature cannot be used in new semester. Therefore, we decided to run models both with and without ID features to test the impact of the ID on the models.

CHAPTER

3

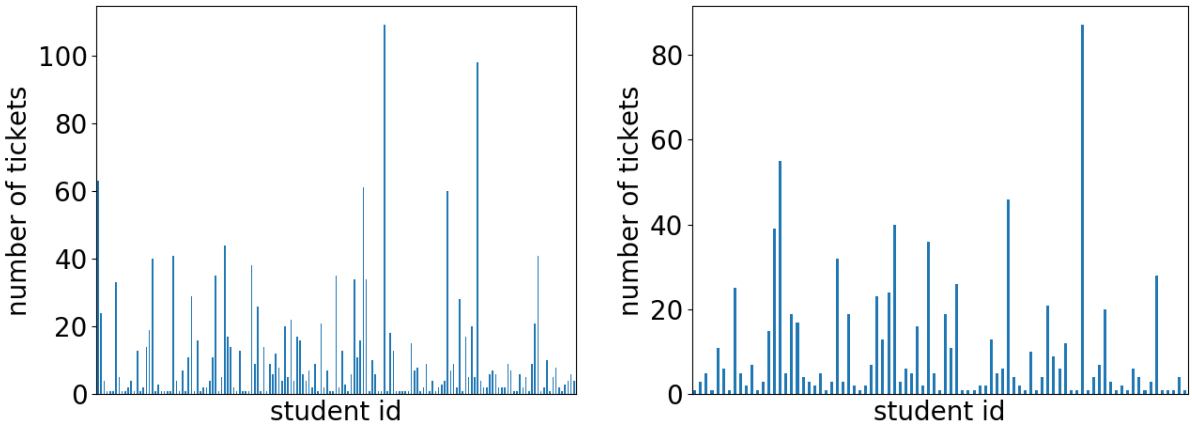
RESULTS

In this chapter, we address all three research questions R1-R3. The answer to the first question R1 on data is given by presenting features of interaction data and the analysis on those features also indicates the participant's behaviors. The percentage of tickets belonging to each of the categories we proposed in chapter 2 also shows which category is the most popular one and thus answer the second research question R2 on the types. The modeling results show the performance of each models. The model with the best accuracy will be our final model which can automatic classify student's questions tickets as the focus of research question R3.

3.1 Data feature analysis results

Fig. 3.1 shows the number of tickets each student raised in Spring 19 and in Fall 19. Both of the semesters have a few students who ask a lot of questions, while the majority of students only asked around 20 questions. There is one student who asked over 100 questions in Fall 19. This could indicate that student was very active in class and encountered a lot of problems.

Fig. 3.2 shows the distribution of the total number of tickets each day. Overall, the Fall semester is busier than the Spring semester. Both semesters only have tickets on weekdays, and this makes sense since there are no weekend office hours offered. In the second week of March, there are no tickets because of spring break, where there were also no office hours. Additionally, Friday is the busiest weekday for Spring while Thursday is the busiest weekday for Fall. The reason for this is because the Spring assignment deadlines are on Fridays, while in Fall, the instructor changed the deadlines to Thursdays. Therefore, we can draw a conclusion that the busiest days in both semesters correspond with the actual deadline for each assignment.



(a) **(b)**

Figure 3.1 Number of tickets per student in (a) Spring 19 and (b) Fall 19

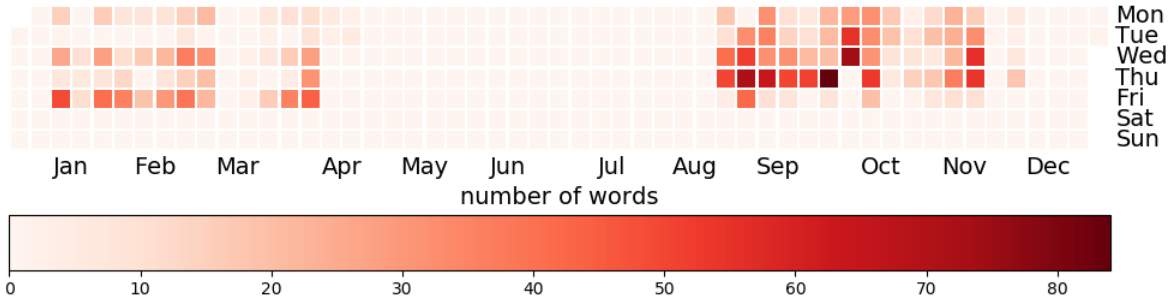
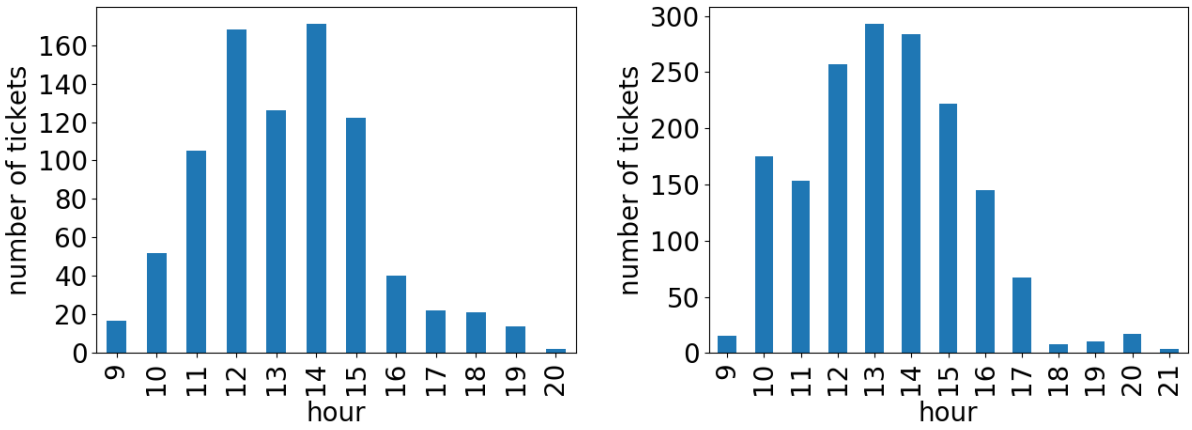


Figure 3.2 Number of tickets per day. Colors varying from white to dark red represent the number of tickets from small to large.

Fig. 3.3 shows the hourly distribution of the tickets in both of the semesters. Most of the interaction happens during the afternoon and the most popular time is around dinner time. A few tickets were handled at night, which was caused by online sessions.



(a) **(b)**

Figure 3.3 Number of tickets per hour in system in (a)Spring 19 and (b) Fall 19

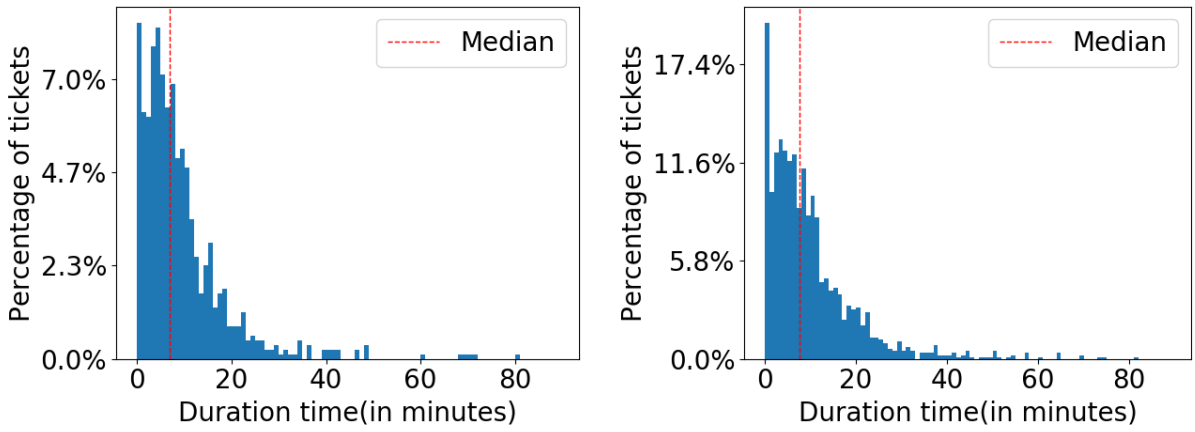
Fig. 3.4 shows the distribution of the duration time for tickets in each semester. Duration time is the time difference between open time and close time. In both semesters, most of the tickets finished within thirty minutes. Also, there is an abnormally and extremely large number of tickets within one minute. However, in reality, answering a question definitely needs more than one minute, especially in this course, since most of the students are asking for debugging, which needs a close examination on the students' code. By looking through those tickets, we found that repeated tickets might be relevant to this problem. Fig. 3.13 shows the distribution of repeated questions and provides evidence for this theory. Based on these observations, we have three hypotheses which might explain this phenomenon. The first one is that the student has multiple questions and those questions are divided into continuous tickets. Such behavior usually results in having multiple repeated tickets where the time difference between the previous ticket's end time and the new ticket's start time should within seconds and the duration time should be roughly the same. The next hypothesis about the reason is that the student didn't show up at the first ticket resulting in a canceled ticket. They then raised another ticket. Such behavior usually results in a very short duration time ticket followed by another regularly repeated question. The final hypothesis is that the teacher forgot to open the ticket in the system, and realized it at the end of the interaction. This kind of behavior has no specific data pattern which makes it hard to recognize through the tracking data in the system. By looking through the data, only five tickets fit the first hypothesis while many repeated tickets fit the second one. Moreover, the final hypothesis is actually pretty common in reality based on discussion with the main instructor. During the busy periods, teachers are too busy to process the requests

Table 3.1 Mean, median, std, min, and max value for duration time (unit: minutes 0

	mean	median	std	min	max
Spring 19	27.13	16.75	30.82	0.03	246.4
Fall 19	27.12	12.95	43.34	0.01	658.77

and it is reasonable that they start the interaction once the student shows up without opening the ticket in the system. Therefore, students not showing up or the teacher forgetting to open the tickets could cause an abnormally large number of tickets within one minute. Also, a few tickets have a really long duration time (exceeding one hour), which happens when the faculty forget to close the ticket at the end of the interaction.

Except for the tickets within one minute, the peak of interaction duration time is around five minutes. Most tickets (75%) were resolved in 15 or fewer minutes.



(a) **(b)**

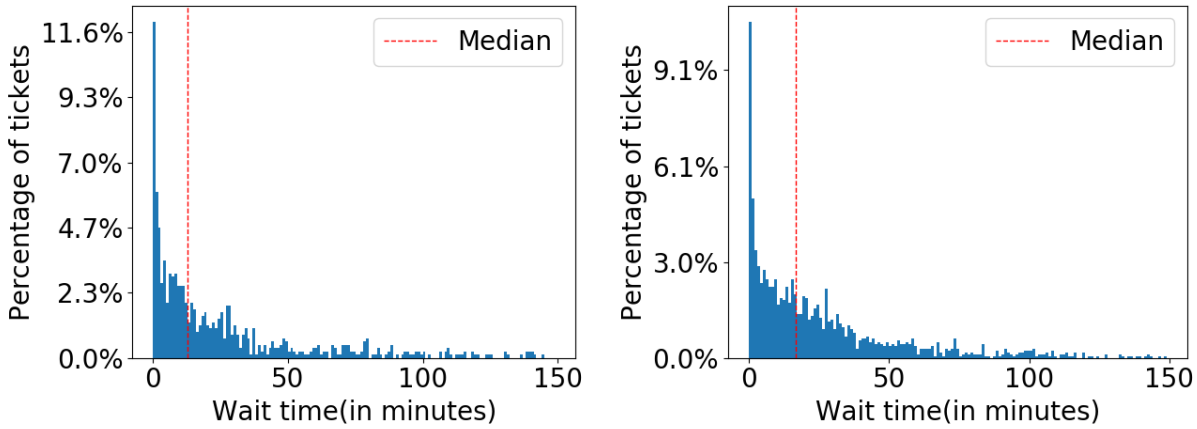
Figure 3.4 Histogram of duration time (time difference between open time and close time) for tickets in (a)Spring 19 and (b)Fall 19

Fig. 3.5 shows the distribution of the wait times in each semester. For both of the semesters, the number of tickets decreases with the increase in the wait time, which is encouraging that most students did not have unreasonably long waits. Although the number of tickets within one minute were also very high, it actually fits the reality. Because many students actually raised tickets during a scheduled office hour and at the office hours location, the teaching staff member could possibly see them right away, such tickets can open instantly. Also, in reality, most of the time the TAs are available during the office hour time unless it is a busy day leading up to a project deadline.

Table 3.2 Mean, median, std, min, and max value for wait time (unit: minutes)

	mean	median	std	min	max
Spring 19	11.53	6.99	57.01	0.03	1652.1
Fall 19	17.304	7.59	130.12	0.03	2145.6

Therefore, if a student want to ask a question during non-busy time, they should be able to meet with an teacher almost immediately. This phenomenon could also indicate that the efficiency of processing questions is actually very good in both semesters since only a limited number of tickets wait a long time. Also, the figure shows that the number of tickets is dropping more quickly in Spring 19 than Fall 19. Since the Fall 19 does have a lot more tickets than Spring 19, Fall 19 is much busier than the Spring 19 and thus the waiting time on Fall 19 should overall longer than the Spring 19.



(a) **(b)**

Figure 3.5 Histogram of wait time for each tickets in (a)Spring 19 and (b) Fall 19 (unit: minutes)

Fig. 3.6 and Fig. 3.7 present the raw text of question topics over both semesters. Most of the students provided the name of the assignment they are working on as the topic. Therefore, the texts are highly similar. Only some small changes like extra blank or common abbreviations divide the tickets that have the same topic. For instances, Project 1 part 1 can be presented as P1P1 or Project 1 Part1.

In order to eliminate these differences we performed some simple text cleaning and alignment. Firstly, we convert all the upper characters into lower characters. Then, we remove all the spaces in the text, so that the extra spaces can be eliminated. Finally, we combine the common abbreviation name of the assignment with the full name of the assignment.For example, "p1p1" is merged into

Table 3.3 Mean, median, std, min, and max value for number for words in problem description

	mean	median	std	min	max
Spring 19	4.04	3.0	3.77	1	48
Fall 19	4.55	3.0	4.38	1	40

"project1part1" and "guidedproject" is merged into "gp". After these cleaning operations for every ticket, the results are in Fig. 3.8 and Fig. 3.9. As you can see, in both semesters, project 1 part 2 and project 2 part 2 are the most popular topics. This may indicate that these two assignments are the most difficult assignments and the instructor should put more resources in these two assignments.

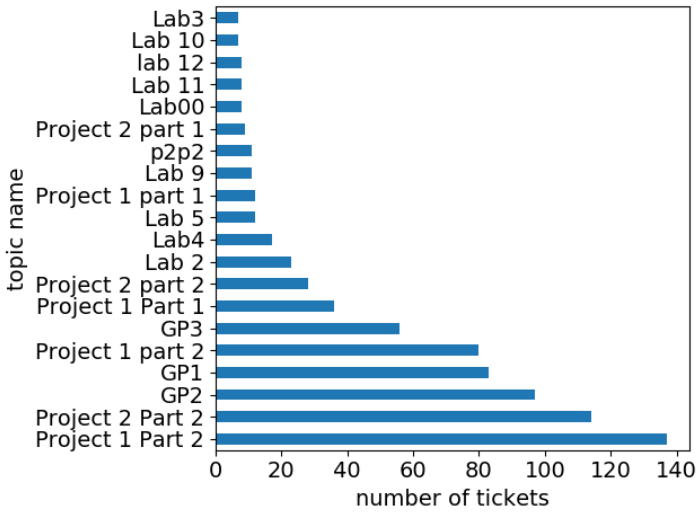


Figure 3.6 number of tickets for top 20 raw text of topic in Spring 19 without binning for common themes.

Fig. 3.10 shows the length of the problem description in the two semesters. The result shows that in both semesters, most of the tickets contain no more than ten words, which implies that students tend to put limited information when they make an office hour reservation. On the other hand, since most of the question is about bug fixing, it is possible that the student just didn't know how to describe their bugs sufficiently. After all, the whole purpose of office hours is that meeting in person can significantly improve the efficiency of communication. Therefore, it is not abnormal that the students tend to describe their questions on MDH in a vaguely and briefly way.

After cleaning the text with the stop words and trimming, we also find which words students tend to use on problem description and attempted solution. Fig. 3.11 and Fig. 3.12 show the most common words appearing in the description and the tried solution on both of the semesters. The red bars represent Spring 19 and blue bars represent Fall 19.

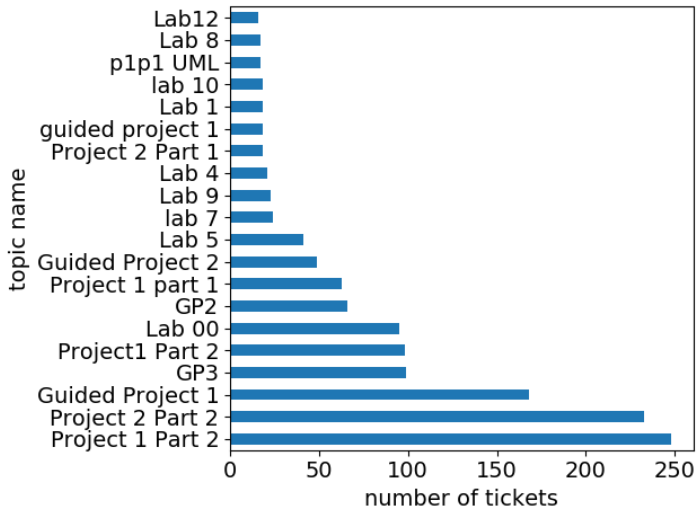


Figure 3.7 number of tickets for top 20 raw text of topic in Fall 19 without binning for common themes.

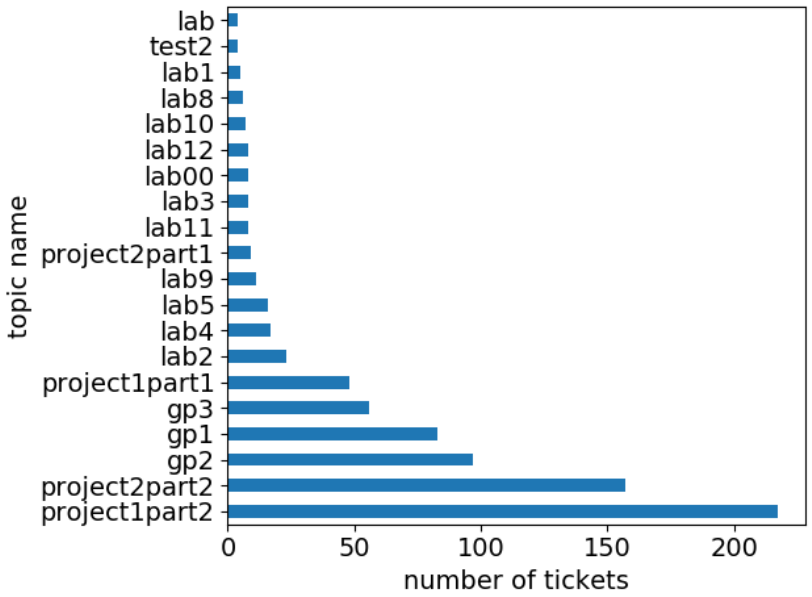


Figure 3.8 number of tickets for top 20 cleaned topic for Spring 19 after binning for common themes

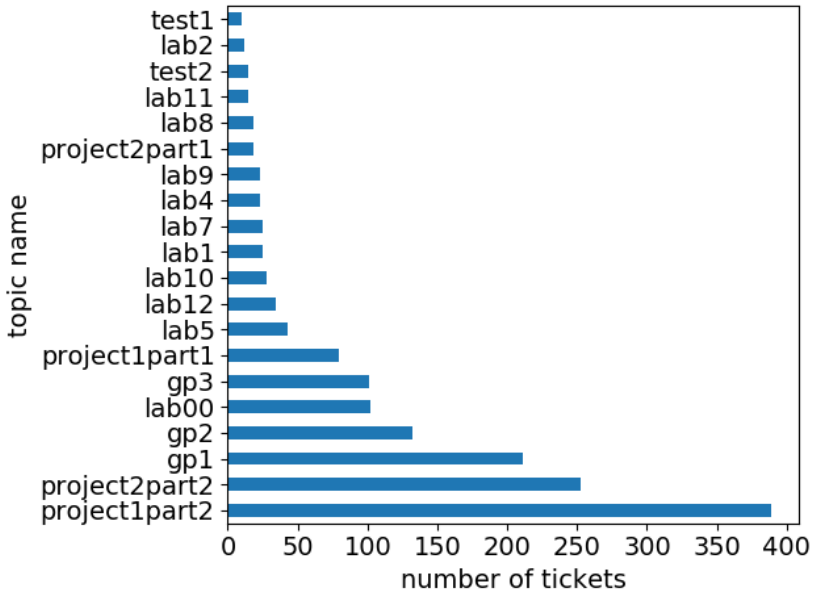


Figure 3.9 number of tickets for top 20 cleaned topic for Fall 19 after binning for common themes

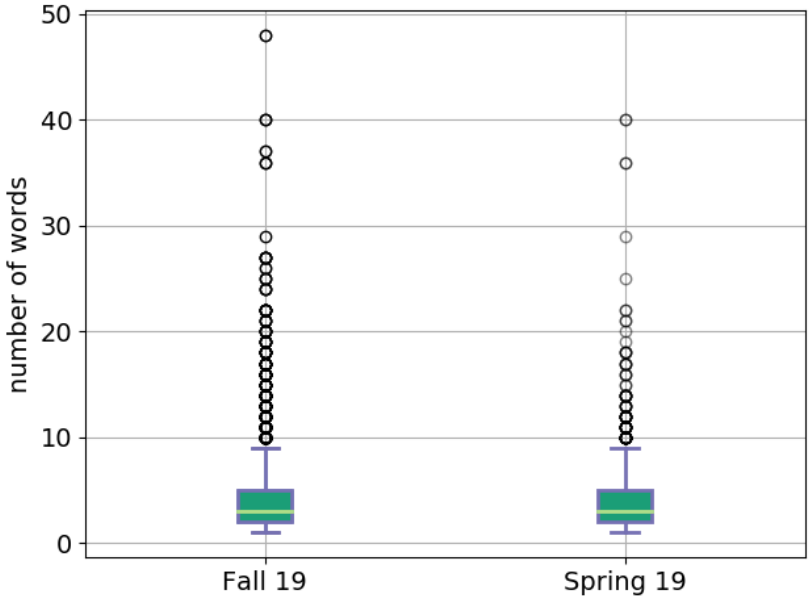


Figure 3.10 Number of words in problem description for each ticket

From the above figure Fig. 3.11, many popular words exist in the problem descriptions on both Spring 19 and Fall 19. The most popular word in the problem description is "test", this could means that the assignments that contains testing tasks like unit and system testing did challenge a lot of students in both of the semester. Also, we can conclude that most of the questions related to bug fixing or test fail because the words like 'fix' and 'error' are highly popular. Similarly, words like "implement" and "constructor " indicate a large number of questions related to coding techniques.

In addition to the common popular words, the number of words that are uniquely popular in different semesters is very limited. As we can see in the graph, only the word "question" is popular in Spring 19 but not popular in Fall 19.

The above figures Fig. 3.12 also present the word choices in tried solution. The most popular word in the tried solution is also "test", which fits an expected task from the problem description. Also, many common words in the tried solution don't provide any information about the solution or any specific method students had tried. For instance, "everything" is a highly common word in the tried solution, but it doesn't describe the solution at all and we cannot know any methods that student might have tried before they asking for help.

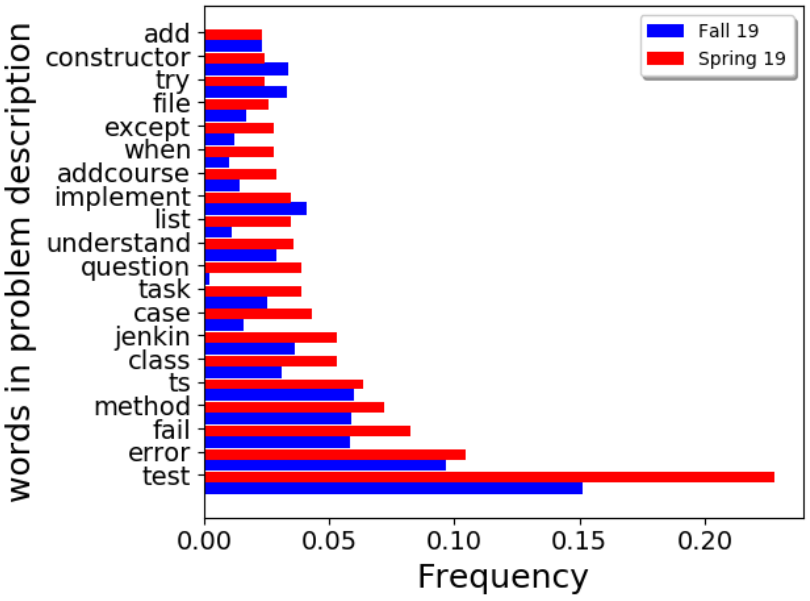


Figure 3.11 Top 20 common words appearing in problem description.The red bars represents Spring 19 and blue bars represents Fall 19.

We also analyzed the nature of the repeated questions. From all the 2,510 questions, there are 249 questions marked as repeated. Fig. 3.13 exhibits the duration time distribution of these repeated questions. Most of these questions are very short in duration time and a large percentage of them are within one minute. Like we discussed before, this could be the evidence for the existence of

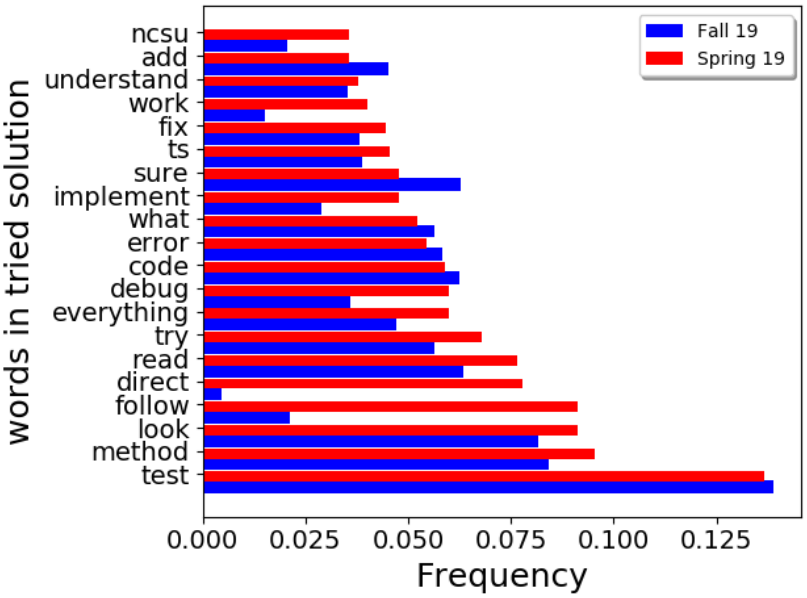


Figure 3.12 Top 20 common words appearing in tried solution.The red bars represent Spring 19 and blue bars represent Fall 19.

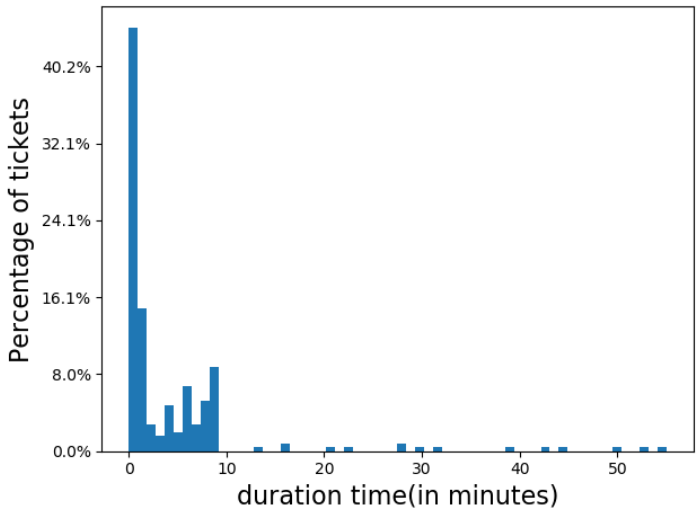


Figure 3.13 Histogram of duration time distribution on repeated questions

behavior that students didn't show up in some of the tickets and raise another ticket to resolve their questions.

We already analyzed the data from both semesters separately and we found out that there is no significant difference of data pattern between two semesters since both semesters have a similar time distribution, similar topic distribution, similar description length, similar popular choices of words, and similar patterns on how to describe the questions. Thus, we can safely draw a conclusion that the pattern of the interaction data is consistent between different semesters.

3.2 Dividing the Category Results

As we discussed in the last chapter, we developed five different categories and labeled all the tickets based on those categories. Fig. 3.14 and Fig. 3.15 show the distribution of labeled tickets on those five categories in Spring 19 and Fall 19. It clearly presents the results to the second research questions and shows what type of questions students are asking.

For both semesters, the insufficient category is the most common category. Over half of the questions belongs to this category. And the test category is the second common type which occupies 20 percent of the tickets. Also, around 15 percents of the questions belong to the sufficient category and 5 percents of the question belong to the useless category. Finally, the copied error is the least common category. Only around ten questions each semester contained the copied error message.

Clearly, the labeling result indicates that students tend to state their questions only with limited information. Also, the questions related to testing problems are very popular. These two conclusions correspond with what we found in analyzing on problem length and common words.

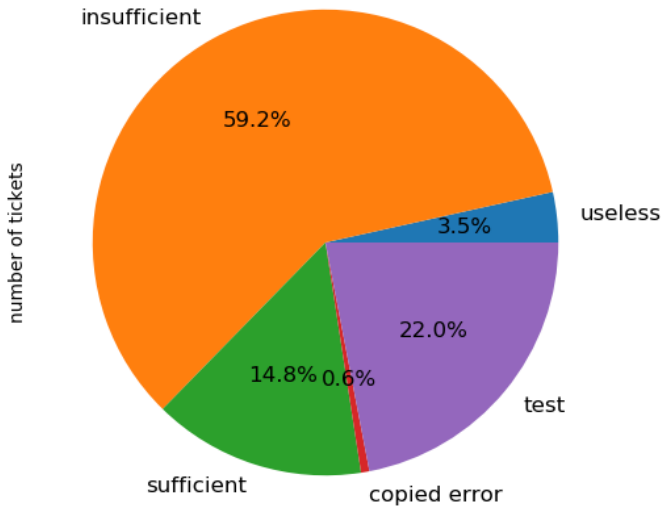


Figure 3.14 distribution of labeled tickets on five categories in Spring 19

Table 3.4 Duration time of each semester

	useless	insufficient	sufficient	copied error	test
Spring 19	18.7 minutes	21.2 minutes	16.9 minutes	10.9 minutes	25.4 minutes
Fall 19	20.1 minutes	19.7 minutes	18.1 minutes	13.2 minutes	26.3 minutes

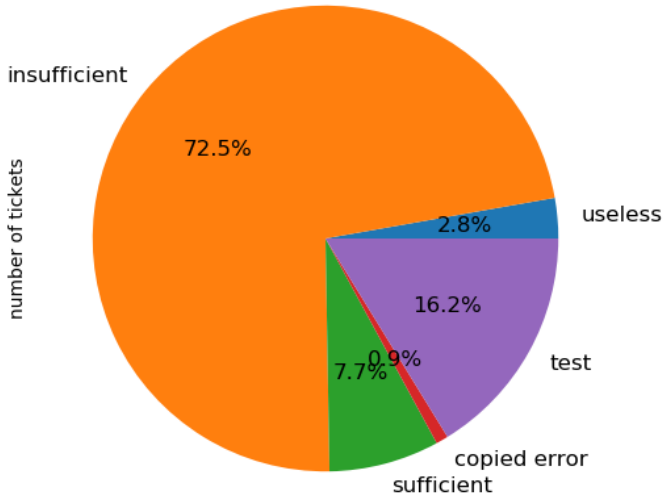


Figure 3.15 distribution of labeled tickets on those five categories in Fall 19

Table 3.4 describes the average duration time of each category across the semesters. The average duration time for each category are similar in both semesters. Useless, insufficient and sufficient categories are around twenty minutes. These three category doesn't have a big difference in duration time, which could mean that how much detail student provide of their questions might not affect the duration time. However, the copied error category has a slightly short average duration time and it might indicate that putting error messages can boost the efficiency of processing tickets. Also, the test category has a slightly long average duration time, which might also indicate that this kind of question need more time to resolve.

3.3 Modeling Results

As stated above, we defined four different models to classify the posts. The model with the best performance should be used as the target model that can automatic classify student's questions. That model is the result of the third research question. We use those four models separately to predict

Table 3.5 Accuracy on each model with ID features

	Logistic Regression	Random Forest	LightGBM	SVM
Accuracy	0.92	0.94	0.97	0.89

the data on the test set and calculate the accuracy. Firstly, let's look at the performance of models with ID-features. The result of models that ID features are in 3.5. Clearly, the LigthGBM model has the best performance with an accuracy of 0.97, while the SVM model has the worst performance with an accuracy of 0.89. Therefore, we are using the LightGBM model as the final target model.

Besides using accuracy, some other measurements are also listed in the following table. Since this is a multi-class classification problem, all measurements are made for each category except for copied error. The reason for not measuring copied error is because data for this category is too limited, only three copied error tickets are in the test set.

We calculated the Precision, Recall, and F-measure of each category over all four models, the results are in3.6. The result shows that no matter which measurement is used to evaluate the performance, the LightGBM is better than others. This may indicate that LightGBM model's performance is relatively balanced. While the Random Forest has the second best performance; the Logistic Regression is the next best model and finally, the SVM model is the worst model. The Precision is overall better than recall. Still, for the Logistic Regression and Random Forest have high Precision and high Recall, which means that these two models are also overall pretty good. However, the SVM model has high precision and low recall, which indicates this model could generate a lot of false negative results. This flaw could cause the bad performance of the SVM model.

Table 3.6 Precision, Recall and F-measure for each category in every modeling method

		Useless	Insufficient	Sufficient	Test	Average
Logistic Regression	Precision	0.922	0.935	0.942	0.987	0.942
	Recall	0.911	0.893	0.901	0.887	0.901
	F-measure	0.916	0.914	0.921	0.934	0.921
Random Forest	Precision	0.974	0.937	0.915	0.998	0.959
	Recall	0.902	0.920	0.933	0.946	0.924
	F-measure	0.937	0.928	0.924	0.971	0.941
SVM	Precision	0.909	0.765	0.911	0.852	0.865
	Recall	0.148	0.783	0.201	0.731	0.446
	F-measure	0.254	0.774	0.329	0.787	0.568
LightGBM	Precision	0.975	0.962	0.992	1.000	0.983
	Recall	0.955	0.991	0.965	0.949	0.965
	F-measure	0.965	0.982	0.978	0.973	0.975

Table 3.7 Accuracy on each model -no id features

	Logistic Regression	Random Forest	LightGBM	SVM
Accuracy	0.83	0.89	0.94	0.82

Table 3.8 Top 10 important features rank on each model (with id features), rank by feature importance on every model

Rank	Logistic Regression	Random Forest	SVM	LightGBM
1	prob-cnt-test	prob-cnt-test	prob-cnt-test	prob-cnt-test
2	prob-tf-test	prob-tf-test	prob-tf-ins	prob-cnt-ins
3	prob-cnt-ins	prob-tf-ins	prob-tf-suf	length
4	prob-tf-ins	prob-cnt-ins	prob-cnt-ins	stop-words
5	prob-cnt-suf	length	studnt-id	character
6	character	character	character	prob-cnt-useless
7	prob-tf-useless	prob-cnt-useless	prob-cnt-useless	duration time
8	prob-cnt-useless	prob-tf-suf	wait-time	prob-tf-ins
9	student-id	prob-cnt-suf	stop-words	prob-cnt-suf
10	prob-tf-suf	student-id	length	punctuation

The above models actually use the student id and teacher id as features. We generated them as a first pass to assess the impact of the IDs on the models. ID-based models however, do not generalize well to new classes and cohorts. Therefore we also trained all the models but omitted the ID features. After deleting those features, our models continue to perform relatively well and have no change in their relative performance as shown in Table 3.7.

This result shows that even without the id features, the LightGBM is still the best model with an accuracy of 0.95, while the Logistic Regression is 0.87, Random Forest is 0.9, and SVM is 0.84. All four models' performance decrease after deleting those features, which indicates that the id feature does help but not greatly. Therefore, while individual students may have common question patterns the differences are not definitive.

We also calculated the feature importance for every include id features models and the 3.8 shows the most important features in each model. The most noticeable thing is the content related features have a really great impact in every model. The most important feature for every model is "prob-cnt-test". Also, Logistic Regression, Random Forest and SVM greatly value the student's id. This means that id features does help those three models a lot and thus they have a greater improvement on the accuracy when including the id features than LightGBM.

CHAPTER

4

CONCLUSION

In this section, we will summarize the answer to each of those research questions. We have successfully analyzed the data retrieved from MDH and identified categories based on the way students describe their questions, and finally, build a model that can automatically classify the questions.

4.0.1 Research question one

The first research questions ask about what data can be tracked in MDH and focus on analyzing participant's behavior. The following data pattern are particular useful and answer this research question.

Firstly, based on the result from analyzing the length of problem description and the distribution of categories, "insufficient" is the most common category and also most of the length of problem description are within ten words. Therefore, we can draw a conclusion that students tend to keep their question description very brief. Many students will enter some related words to their questions instead of describing the whole question thoroughly. For example, if a student cannot pass one test case in that assignment, he might just describe his problem as "test case failed".

Second, by analyzing the result of duration time of tickets, we have already found which day is busy during both of the semesters. On weekends and spring break, there are no tickets at all, while on Thursday in Fall 19 and Friday in Spring 19 are highly busy. Therefore, based on the discussion in chapter 3, the most busy period of office hours is always around the deadline. Thus, the instructor should consider focusing on putting more office hour resources around the corresponding deadline.

Then, the result of common word count shows us the most popular words is the word "test". Also, the popularity of words like "error" or "implementation" indicates that there are a huge number of

questions related to bug fixing. Therefore, most of the questions students are asking is about the bug fix and test case fail on some programming assignments. This could be implied the students need to improve their debugging skills.

Finally, we analyzed the time duration of all the tickets. In our discussion, the unusually high number of tickets within one minute draw our attention and we bought up three possible behaviors that could cause this anomaly. And since we found that half of the repeated tickets are within one minute, we believed that this anomaly could relate to the repeated question. The first behavior is students divide the office hour session into several questions and tickets; the second behaviour is student doesn't show up in the first tickets and then raise other tickets; and the last behaviour is the teacher forgot to open the tickets when the interaction began. In our discussion, we conclude that the second one and the last one is more likely since it fits the pattern we found in the data.

4.0.2 Research question two

For the second research question, we try to answer what type of data students are asking and how are those questions distributed over the course. In order to address this, we come up with a method to divide the questions into different categories. Those categories explain what kind of questions students are asking, those categories are: useless, insufficient, sufficient, copied error, and test.

While considering the categories, the discussion on average duration time indicates that the copied error category might have a small duration time and it could suggest that this kind of behavior does help the efficiency of debugging. However, this conclusion is not very convincing since the number of tickets containing the copied error message is very limited (less than 1 percent) in our data set. Therefore, we can advice the instructor to encourage the students to copy the error message when seeking help for debugging.

Also, the insufficient category is the most common category in both semester. In Spring 19, there are 59.2% of tickets in this category while in Fall 19 it is 70.2%. The around 10% difference could indicate that the students became inpatient when the office hours is very busy and thus tend to state their questions with less detail.

4.0.3 Research question three

For the final research question, our goal is to build an automatic classifier on question's description. We did feature selection and build models with that. We use four different modeling methods(Logistic Regression, Random Forest, SVM and LightGBM) and compare their performance. Our results show that LightGBM has the best performance. Also, by comparing the result on using and not using student id as a feature, we can clearly see that the student id does improve the performance a little, which indicates that the same student tends to ask the similar type of questions. For instance, if a student copied the error message to the problem description, it is highly possible that he will copy the error message again when he encounters another debugging issue.

In the future, we should firstly use the data from new semester to test the performance of our model. This can indicate weather our model can be used in different semesters. Moreover, we could

build a model to predict the interaction time for each ticket when it gets raised. By doing this, it can significantly help the system to calculate the approximate wait time for each student in the line.

Also, we could use our model to predict the student's question type after they raise the tickets in the system and automatically notify the students to refine their statement when their problem description is useless or insufficient.

BIBLIOGRAPHY

[AH19] Al-Hawari, F. & Barham, H. "A machine learning based help desk system for IT service management". *Journal of King Saud University-Computer and Information Sciences* (2019).

[Alt14] Altintas, M. & Tantug, A. C. "Machine Learning Based Ticket Classification in Issue Tracking Systems". 2014.

[Bre01] Breiman, L. "Random Forests". *Mach. Learn.* **45**.1 (2001), 5–32.

[Cam18] Campbell, J. & Craig, M. "Drop-In Help Centres: An Alternative to Office Hours". *Proceedings of the 23rd Western Canadian Conference on Computing Education*. WCCCE '18. Victoria, BC, Canada: Association for Computing Machinery, 2018.

[Cri99] Cristianini, N. & Shawe-Taylor, J. *An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. USA: Cambridge University Press, 1999.

[Don11] Dong, Z. et al. "Build Peer Support Network for CS2 Students". *Proceedings of the 49th Annual Southeast Regional Conference*. ACM-SE '11. Kennesaw, Georgia: Association for Computing Machinery, 2011, 42–47.

[Has09] Hastie, T. et al. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009.

[Joh98] Johnson, J. et al. "Virtual Office Hours Using TechTalk, a Web-Based Mathematical Collaboration Tool". *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, 130–133.

[Ke17] Ke, G. et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". *Advances in Neural Information Processing Systems 30*. Ed. by Guyon, I. et al. Curran Associates, Inc., 2017, pp. 3146–3154.

[Liu16] Liu, Z. et al. "MOOC Learner Behaviors by Country and Culture; an Exploratory Analysis." *International Educational Data Mining Society* (2016).

[Mac13] MacWilliam, T. & Malan, D. J. "Scaling Office Hours: Managing Live Q&A in Large Courses". *J. Comput. Sci. Coll.* **28**.3 (2013), 94–101.

[Ped11] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* **12**.85 (2011), pp. 2825–2830.

[Pen02] Peng, C.-Y. J. et al. "An Introduction to Logistic Regression Analysis and Reporting". *The Journal of Educational Research* **96**.1 (2002), pp. 3–14. eprint: `https://doi.org/10.1080/00220670209598786`.

[Smi17] Smith, A. J. et al. “My Digital Hand: A Tool for Scaling Up One-to-One Peer Teaching in Support of Computer Science Learning”. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’17. Seattle, Washington, USA: Association for Computing Machinery, 2017, 549–554.

[Son14] Son, G. et al. “On Automating XSEDE User Ticket Classification”. *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. XSEDE ’14. Atlanta, GA, USA: Association for Computing Machinery, 2014.

[Vel17] Vellukunnel, M. et al. “Deconstructing the Discussion Forum: Student Questions and Computer Science Learning”. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’17. Seattle, Washington, USA: Association for Computing Machinery, 2017, 603–608.

[Wan17] Wang, D. et al. “LightGBM: An Effective MiRNA Classification Method in Breast Cancer Patients”. *Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics*. ICCBB 2017. Newark, NJ, USA: Association for Computing Machinery, 2017, 7–11.

[Wan13] Wang, L. et al. “CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model”. *Nucleic Acids Research* **41**.6 (2013), e74–e74. eprint: `https://academic.oup.com/nar/article-pdf/41/6/e74/25339876/gkt006.pdf`.

[Wei17] Wei, X. et al. “A Convolution-LSTM-Based Deep Neural Network for Cross-Domain MOOC Forum Post Classification”. *Information* **8** (2017), p. 92.

[Zwe18] Zweben, S. & Bizot, B. “2017 CRA Taulbee Survey”. *Computing Research News* **30**.5 (2018), pp. 1–47.