ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #36492

Automated Zoom Chat Analysis Including Chat-Based Polls for an Online Introductory Programming Course

Frank Vahid (Professor)

Frank Vahid is a Professor of Computer Science and Engineering at the University of California, Riverside, since 1994. He is co-founder and Chief Learning Officer of zyBooks, which creates web-native interactive learning content to replace college textbooks and homework serving 500,000 students anually. His research interests include learning methods to improve college student success especially for CS and STEM freshmen and sophomores, and also embedded systems software and hardware. He is also founder of the non-profit CollegeStudentAdvocates.org.

Stanley Zhao

Joe Michael Allen (Student)

Automated Zoom Chat Analysis Including Chat-Based Polls for an Online Introductory Programming Course

ABSTRACT

We describe a tool that automatically derives student participation statistics from a Zoom chat log. We discuss how our 100 student CS1 section, taught online every term the past 7 years to 1,500 students total, evolved to use chat extensively, for real-time questions/answers and also for a rapid dynamic polling approach to teach programming such as "What does this code output?" or "Now you finish the branch expression". The tool counts messages per student to allow for automated participation credit. To provide instructors further insights, it also approximately categorizes messages into poll or non-poll messages and lists the time each poll was conducted. Comparisons with manual analysis show the approximations to be 80-85% accurate, sufficient for insight purposes. We show course evaluation data suggesting that active chat-based polling likely contributed to our online course evaluation scores rising from a respectable 4.2/5 to a well-above average 4.8/5. We hope to make the tool freely available to computer science and other instructors to help them give participation credit for online course participation in the chat and to track their own poll usage and response rates to experiment with their online teaching approach of keeping the class engaged.

1 Introduction

In 2012, we introduced an online section of our introduction to programming course ("CS1"), representing the first online course at our university. Our CS1 has been taught every term since and serves 300-500 students per term, divided into sections of about 100 students including one online section every term since 2012.

At the time the online section was created in 2012, our in-person CS1 had already adopted several active teaching pedagogies [Me93], with students in both in-person and online sections doing extensive weekly online activities with immediate feedback, consisting of about 100 short answer / multiple choice questions (with explanations for right and wrong answers), 15 small coding problems, and 2 programming assignments, all online and auto-graded with immediate score feedback, plus some online reading and watching of short videos and animations. In-person lectures consisted mostly of instructors doing examples, pausing 3-5 times per 80-min period to have students work on small coding problems themselves, often in small groups.

The original online section was run mostly asynchronously. Students had to attend just one discussion hour of their choice per week, mostly with the instructor doing examples and Q&A. Those early online sections had high DFW rates (nearly 50%), as shown in Figure 1.

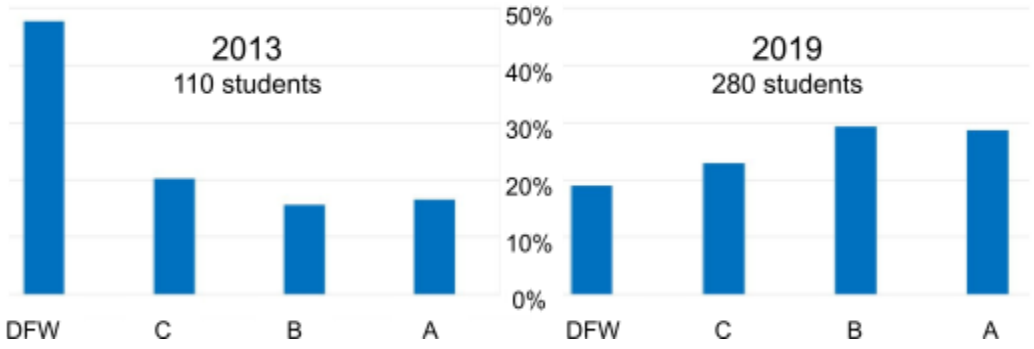

Figure 1: Our CS1 online section evolved to include more required synchronous meeting time, making extensive use of chat, leading to improved DFW rates and grades.

Over time, we introduced more synchronous online meeting times, leading to improvements in DFW rates and grades, and in course evaluations, until the online section's synchronous meeting requirements matched the in-person sections, including required weekly participation in two 80-minute lectures and in one 2-hour lab. Like in-person lecture, the online lecture consisted mostly of live-coding of examples. Figure 1 shows the online section's DFW rate and grades improved until they matched the successful in-person sections, with those DFW rates well below the national average for CS1 classes of 30% [To21][WaLi14]).

As the online section evolved toward more required synchronous participation, the online chat message box became the unexpected star. Because chat participation became a required class component (typically 5% of the course grade), we sought a way to automatically give participation credit to students for typing in the chat, since manual counting was time consuming. Towards that end, in this paper, we describe a tool that automatically counts messages from each user in a Zoom chat log file.

More importantly, we discuss an evolved usage of the chat message box for frequent "polls", which we found kept students engaged online. While live-coding examples, instructors would pause frequently and ask students to respond in the chat to questions like "What does this statement output?" or "Complete the missing part of this expression", giving participation credit for "reasonably participating" with such polls (students don't have to answer every poll). To help instructors gain insight into their polling frequency during class, how many students were responding, and how many non-poll messages still were happening (which gives a sense of class engagement), we extended the tool to strive to approximately detect those items just from the chat log, with no special extra input like markers indicating poll locations. We intend to make the tool freely available to CS instructors (and others) as a free web-based tool.

2 Basic chat during online lecture

As we taught our online course synchronously with the instructor sharing their video and audio with the class, we experimented with different ways of having the students interact back. We found the chat box supported extensive interaction from and among students, without the delays, distractions, and technical barriers often accompanying attempts to use student audio or video, especially in a class of 80+ students.

"Chat" refers to a box present in most video meeting systems, including Zoom. Participants can type in a text input box, and upon hitting "Enter" their text appears in the chat box's stream of messages from all participants. Below is an example of chat messages during our online class. The items in parentheses are notes for this paper's reader and weren't present in the chat. *Everything below from the instructor is verbal, not typed*. The example is abbreviated; for example, the chat below shows two "welcome" messages whereas dozens were actually posted.

(1 min before class, students are present, professor joins)
Mia: Hi Professor!
Jia: How was your weekend professor?
(Instructor verbally "Great! Anyone do anything fun?)
Sam: I went camping at Joshua Tree
Pat: It was so hot last I was there
(Instructor: "OK gang, let's get started")
(Instr shares a coding window, starts doing example...)
Juan: Why no semicolon?
Mia: Not needed at end of for loop
Juan: Oh ya, thanks Mia.
(Instr still doing example)
Taz: Ummm, I'm getting lost...
Jia: Yo, me too
(Instr: "Thanks Taz and Jia. What can I explain?"
Taz: Like, why for rather than if?
(Instr: "Because we want to read 5 values, so we loop 5 times")
Taz: Duh!!!! What was I thinking
(Instr: "It's normal Taz! Loops take getting used to.")
Jun: I was wondering the same thing.
Mia: Loops had me loopy at first
Sam: lmao!!!

Students note in survey and course evaluations the benefits of an online class that uses the chat, including:

- The "friendly banter" before class and throughout class that isn't common in person.
- The ease of asking questions, even "I'm confused". Many say they rarely ask questions in person.
- The answering of questions by classmates.
- The instructor referring to students by name.

Observing the above chat, some instructors might initially believe the chat would distract students, but that is not the case in general, as long as the instructor keeps the chat focused on the current lecture content.

Of course, instructors must create a positive culture that welcomes good use of chat. We find early in a term that students go on tangents in the chat, but prompt friendly instructor reminders to keep chat focused on the current lecture topic leads in just a week or two to the chat staying focused. We also found that a chat's first 15-20 minutes can get inundated with course structure questions like "Is there a quiz this week?", especially by students who show up late. We found that a solution is to consistently let students know "We'll always pause halfway to answer any class structural questions" -- and to remind students of that (in a friendly way) if they do post a class structural question early in class. In other classes, we have seen some instructors turn off chat due to negative or tangential messages, but with the basics above, we have had no problems with chat in 9 years and dozens of online section offerings to over 1500 students total.

3 Using chat for polls

Polls help keep students engaged in class, whether in-person or online, and are especially appropriate for programming classes.

It is well known that instructors should not talk for long stretches, but rather should engage students via back-and-forth and activities [Jo76][Mi96][Fe16]. Thus, many successful instructors use polls in in-person classes. In person, a professor might ask a question like "Does this code output 5 or 6?" and ask for a show of hands. Some professors use "clicker" technologies, and increasingly use web or phone-based polling apps, with results appearing on the lecture screen [ArKr17][Ic21][Ka15][St15][Th21][WhRa10].

For online classes, video conferencing tools like Zoom often have built-in poll support, allowing participants to select from pre-setup choices or to type an answer. Such polling in online classes is known to improve engagement [ShWa09][VaAl20]. We tried using Zoom's built-in polling, and other online polling tools. But, all the tools required at least some setup of each question by the instructor. We found doing setup before class was too rigid, because we intentionally don't over-prepare and we often adjust content based on student questions or think of questions on the fly. Setting up the polls during class slowed the pace too much. Zoom has a yes/no option for quick polls, but yes/no greatly limits the types of questions we can ask to just binary ones.

Instead, we found ourselves using the chat for polling. An instructor might ask "What does this code output?" and then watch answers fly by in the chat box. The instructor can't see every answer nor know the percent who gave each answer, but can get a sense. If seeing these numbers fly by -- 5 5 5 5 5 5 5 6 5 5 5 5 6 5 5 -- the instructor knows most students think the answer is 5, and a few think the answer is 6.

Chat is superb in programming classes for asking short answer questions related to coding. As an instructor is typing a program, they can pause and say "So what should the if statement's expression

be?" followed by students typing. As the expressions fly by in chat, instructors can look for trends or common mistakes and speak to them: "Mia, I see you and some others typed x = 5. That's a common mistake to use = rather than ==, which assigns rather than compares. Thank you for making that mistake so others can avoid it later. We learn more from mistakes than right answers!" To which a common chat message reply from a student like Mia is something like "Glad I could help! LOL". (Incidentally, no student has ever complained about being called out on mistakes, not even on the anonymous end-of-term course evaluations. Most evaluation comments are highly positive about how the chat is used).

Extensive use of chat has become an integral part of our course and many other online courses. Such use is a key contributor to our online class getting high end-of-term evaluations, typically putting the class in the 80th percentile of all classes on campus. Comments like this real one are common: "I strongly would recommend anyone to take this course online rather than in person! Believe it or not, this was actually my most interactive class even though it was online!".

4 Analyzing chat messages for participation

We give course points for participating in synchronous meetings (lecture and lab), typically 5%, per our earlier-mentioned discovery that such participation dramatically improved student performance in our online sections. Those points result in most students attending. We tell students that participating is defined as having a "reasonable" amount of activity in the chat, and internally we (very generously) currently just define that as 3-5 messages during an 80-minute session. In our Spring 2021 CS1, participation scores based on chat averaged 90%, with 80% of students receiving 100% credit. Better definitions are of course possible, such as messages being spread across the session, some messages being meaningful (like real questions or answers, rather than "LOL"), etc.

We developed a web-based tool to analyze Zoom chat logs. Zoom automatically stores a chat log on an instructor's computer, and also makes the log available for download from Zoom's website. The chat log has one message per line, earliest to latest. An excerpt from a real such log appears in Figure 2 (real full student names are replaced below with fake names, first-name only for brevity). While the chat may seem chaotic, it makes sense when hearing what the instructor is saying and seeing what they are doing.

```
15:52:51 From Pat : nice!
15:52:53 From Ali : nice
15:52:57 From Stu : cool
15:52:57 From Jia : Why do I always see “unsigned int I” in the book for the iteration variable
15:53:02 From Kallee : ooo
15:53:07 From Sue : it would make it a period
15:53:12 From Jia: Period after !
15:53:14 From Jim : oop
```

15:53:21 From Stu : if statement?
15:53:23 From Maria : check if space
15:53:30 From Li : cant you earase it by doing pop_back
15:53:33 From Sara : Use the ascii table?
15:53:36 From Jun : if not abc
15:53:38 From Jia : Don't we usually just list all punctuation possibilities
15:53:45 From Cindy : That's a function?

Figure 2: 1 minute of a zoom chat log in our 80-min 100-student CS1 online section Spring 2021. The instructor is live-coding a loop that adds to a vector. Each message in the log has a timestamp, user name, and content.

An instructor can upload one or more log files to our web tool, which runs a server-side Python script to generate a table showing message counts per student for each log file, as in Figure 3.

Student	Wk7Tu	Wk7Th	Wk8Tu	Wk8Th
Ali	22	18	17	14
Cindy	10	11	17	13
Jia	12	13	16	15
Jim	30	35	32	31
Jun	47	4	0	26
Kallee	21	12	24	23
Li	4	7	5	6
Maria	14	19	14	14
Pat	14	13	12	15
Sara	13	22	17	17
Sue	19	0	0	0

Figure 3: Message counts per student auto-derived by our tool from four zoom logs for our 100-student 80-minute CS1 online section Spring 2021 (only 11 students shown, names replaced).

The table is sortable by any column, to find the most or least active students, and downloadable as a csv file, for uploading to a spreadsheet or gradebook.

We found we often wanted different views of a chat log. For example, we wanted to see every post from a particular student. As such the tool provides numerous views of the chat log and statistics, each downloadable as a csv file:

- Num. messages per user: A users table and a message count for each, often used for participation points, as in Figure 3.

- Num. of messages per minute: A table with a row for each minute and that minute's message count, giving a sense of the activity throughout the lecture, suitable for generating a bar chart for visualization.
- Num. of messages and unique users: Three values: Total message count for the lecture, number of participants, and number of participants who posted at least 3 times.
- Messages by user by timestamp: A table of all chat messages (name, timestamp, message content) but primarily sorted by user name, secondary sorted by timestamp, allowing viewing all students for a given student (like all Mia's messages).
- Original chat log: A table of the original chat messages, which are sorted by timestamp.

5. Automatically classifying poll vs. non-poll messages: A burst-detection approach

Because we found frequent polling to be key to student engagement, and because polling is a teaching skill that instructors are learning and developing, we wanted to automatically determine statistics on polls, like how many polls were given, how spread they were, and how many students responded to each. Such info can help instructors improve their use of polls in teaching.

Figure 4 shows a typical poll occurring in chat during lecture. The first number is a simplified timestamp indicating the second at which the message appeared.

```
565 Sam: He missed a parenthesis
567 Mia: Missing ( prof
(Instructor: "Nice catch Sam and Mia. I'll add it.")
(Instructor types "if (x ?? y)" and asks: "OK, so what's ??" thus starting a POLL)
605 Juan: &&
605 Mia: &&
606 Jen: Are the parentheses required?
606 Taz: &&
608 Jia: &&
609 Ali: no clue
609 Ally: ||
610 Mary: &&
610 Joe: &&
611 Ben: &&
612 Juan: Wait, I meant ||
(Instructor: "OK, most of you think it's AND. Actually it's OR, because either variable could be
  true. Looks like Ally got it!"
614 Sal: &&
622 Jia: Go Ally!!!
```

Figure 4: Simplified chat log with poll responses.

The poll response messages (&&, ||, no clue) add to the total messages and messages per minute statistics from our tool. But instructors may want to know how many non-poll messages were in a log, to get a sense of how much normal Q&A and discussion was happening outside of poll responses. Thus we sought a way to automatically classify each message as a poll or non-poll message. A challenge is that the chat log does not have any "markers" indicating a poll's start or end. An instructor could of course type markers like "Start poll" and "End poll", but this slows the pace, and also is imperfect due to students commonly typing questions while others answer the poll (as in Jen's "Are the parentheses required?"), plus people will still answer even after a "Stop poll" message.

A poll is visible as a sudden burst in messages, so our approach seeks to detect bursts to detect polls. The approach counts messages per second, and looks for adjacent seconds whose counts exceed a threshold; those adjacent seconds are deemed to be one poll. However, we found that the count per second often dips and rises during the same poll. To avoid a dip resulting in two detected polls rather than one, we smooth the data by calculating a rolling average: Each second's count is replaced by the average of the raw count for the preceding, current, and following seconds. For one real poll, Figure 5 shows the raw message count per second and the smoothed counts (rolling average). We detect a poll as a group of adjacent counts that exceed a threshold. All messages in those detected seconds are labeled as poll messages, and all other messages are non-poll messages. This approach is approximate, because sometimes other messages appear within the poll messages (like a question, comment, or reaction), and because some poll messages are late (stragglers) and also some messages come before most of the others (quick responders).

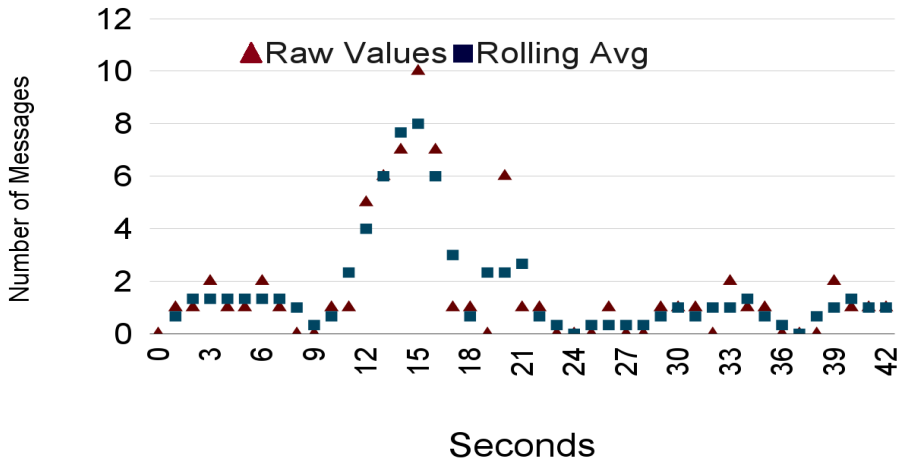

Figure 5: Raw and smoothed (rolling average) messages per second, for just over 2 minutes, showing a poll near the center.

6. Classification accuracy

Message classification

Our goal was to automatically get reasonable approximations of poll and non-poll message counts. To determine accuracy of our approach, we first manually examined all log messages (1981) from an 80-min 100-student class and labeled each as poll or non-poll based on their content. Poll messages are obvious to a human, such as a burst with dozens of "yes" and "no" messages and related messages like "don't know", "idk", "yup", "nope", "thinking...", etc. Figure 6 summarizes the automated poll labeling accuracy compared to manual. False negatives are the main cause of inaccuracy, due to fast poll responders and stragglers. False positives are due to non-poll messages during polls, like questions, or reactions like "lol".

Poll msgs (per manual)	Non-poll msgs (per manual)
True positives: 1212	False positive: 99
False negatives: 247	True negatives: 423
Sensitivity: (1212/1459) = **83%**	Specificity: (423/522) = **81%**
Manual poll/total: (1212+247)/1981 = 74% labeled poll Auto poll/total: (1212+99)/1981 = 66% labeled poll	

Figure 6: Burst-detection heuristic for labeling poll messages is about 80-85% accurate in sensitivity and specificity, sufficient for "general insight" purposes; future improvement surely possible.

7. Using the poll/non-poll labeled messages

With messages approximately labeled as poll/non-poll messages, we update messages per minute with poll/non-poll counts, as in Figure 7. Those counts enable statistics as in Figure 8 for two classes, which compare two classes.

minute	poll	non-poll	Total
0	60	2	62
1	3	46	49
2	0	13	13
3	0	6	6
4	0	30	30
5	13	25	38
6	32	29	61
7	98	12	110
8	116	15	131

9	0	5	5
10	0	11	11

Figure 7: Providing poll and non-poll counts per minute.

Course (100 students each)	Poll Messages	Non-Poll Messages	Total
CS1 Online Spr'19	126	164	290
CS1 Online Spr'21	1311	670	1981

Figure 8: Summary chat statistics for two classes, showing very different use of the chat by the two classes.

Another use of those counts is to create a stacked bar chart of messages per minute, which can help instructors quickly see their class' approximate poll and non-poll activity levels, response rates to polls, and poll frequency. Figure 9 shows such a plot for one 80-min lecture period of our 100-student CS1 online from Spring 2021.

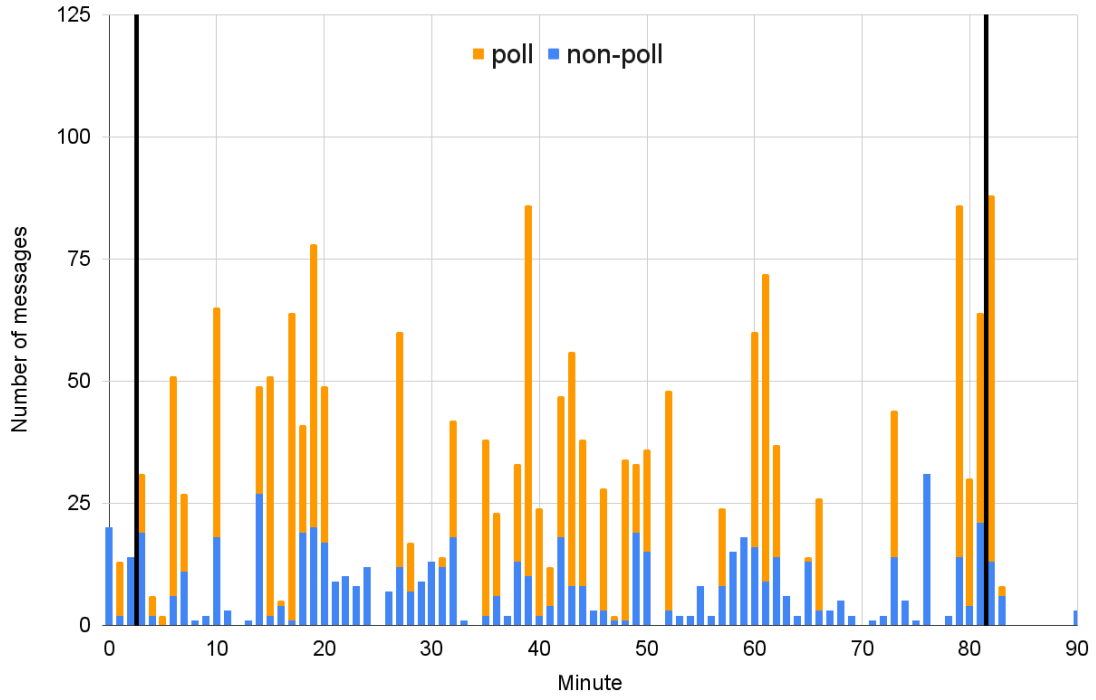

Figure 9: Poll and non-poll counts per minute as a stacked bar chart. Black lines show the start and end of actual class time.

From the chart, an instructor can easily see that this lecture had a healthy number of polls occurring throughout the lecture, with the lull in the middle due to the instructor pausing to take "class structural questions" midway (as usual). One can also see a healthy response rate to those polls, and also a healthy level of non-poll messages occurring throughout. Note that some polls span across two minutes so may appear as two yellow bars, and conversely two polls could be conducted in the same minute so may yield a very tall yellow bar. Note also that students are chatting before and after the 80-min class, which is why the x axis spans more than 80 minutes.

For comparison, Figure 10 shows a chart for the same instructor teaching the same course to a section (also 100 students) in Spring 2019, before adopting the approach of extensively using chat for polls. The instructor still mostly live-coded examples, but would break 3-4 times and have students write code for 2-3 minutes in a google doc, of which the professor would randomly open a few and discuss. (To enable visual comparison with the previous chart, we kept the y axis scale the same.)

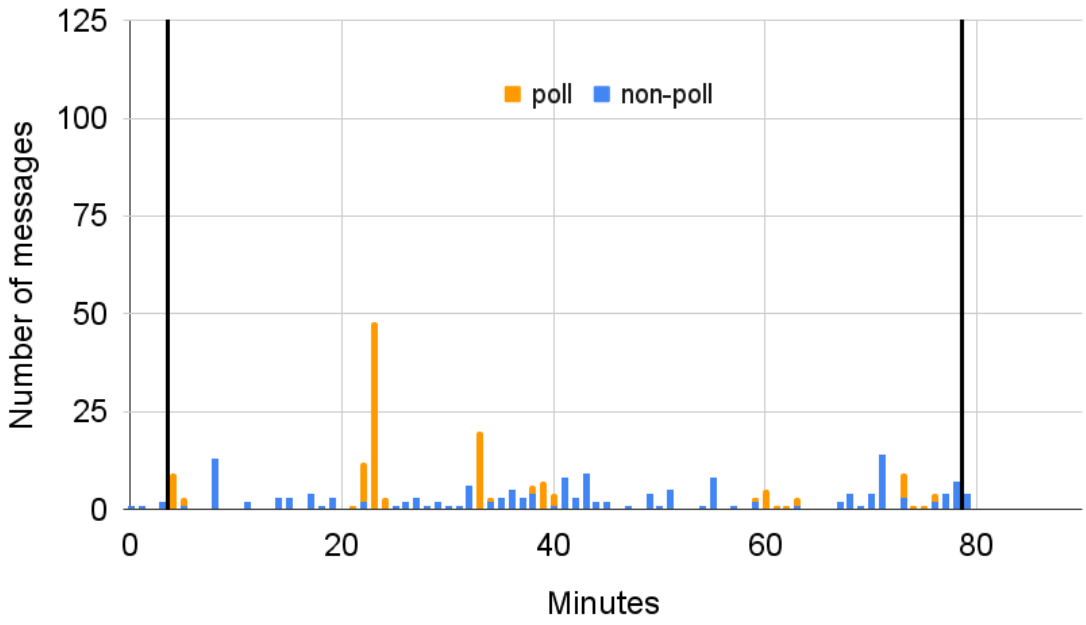

Figure 10: Poll and non-poll counts for the same class but in 2019, before the instructor began using the chat for polls.

One can immediately see that only a couple polls were given, and the response rates were low. One can also see that the *non-poll activity is much lower* than in the previous chart -- suggesting that an active poll-in-chat approach increases overall student participation/engagement as well. One can even see that there is little chat activity before/after class, as the messages span 80 min. Over time, the instructor learned that those few in-class polls seemed more effective, and thus shifted to using them extensively,

leading to a far more engaged class, with more non-poll messages and with healthy discussions before/after class too, as in Figure 9.

Poll counting

In addition to (approximately) classifying every message as poll or non-poll and outputting a table of poll/non-poll messages per minute, our tool also outputs a table of every poll -- a poll's number (poll 1, poll 2, etc.), and its start and end time. With that data, we can generate a plot showing how polls were spread across a class period, as in Figure 11, which shows a blue bar for each poll in the class from Figure 9, showing polls occurring every few minutes.

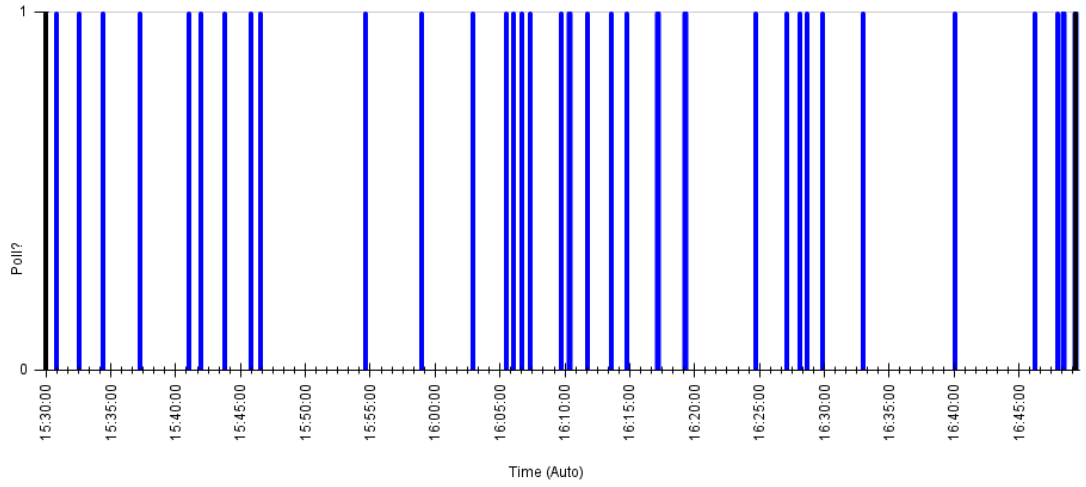

Figure 11: Plot showing each poll's appearance during class.

Compared to our manual analysis, our poll counting was about 89% accurate, counting 34 polls when 38 actually existed. Inaccuracies in poll counts. are due to treating a few rapid back-to-back polls as one larger poll. Accuracy would be higher if polls were less frequent; the above class is on the high-end of poll usage.

In the future, we may update our approach to examine message content to gain further accuracy.

8. Course evaluations

While the purpose of this paper is to introduce the tool, nevertheless it may be interesting to publish our course evaluations before and after actively using polls in lectures. Our tool can help instructors gain quick insight into their use of in-chat polls in a dynamically-run online class. Such poll usage helps engage students. Such engagement can be seen in part by course evaluations. Figure 12 shows the

average course evaluation score from students being 4.1 or 4.2 out of 5 (respectable scores) before adopting an active in-chat poll approach, rising to 4.8 after (considered exceptionally positive, far above average, and staying near 4.8 for every offering in the past few years). The courses were all CS1, taught by the same instructor. Two key changes were made before and after -- active in-class chat, and switching from one large program to many small programs per week. And of course other variables change as well. So it cannot be concluded that the rise in course evaluations was due solely to the in-chat poll approach, but the approach likely contributed. This is consistent with student comments received on those evaluations after the change, such as "class participation was extremely engaging", "one of the few times I actually wanted to go to lecture...because of the interactions", "teaching method is incredible", "I felt safe to type in the chat even if I was not certain my answer was correct".

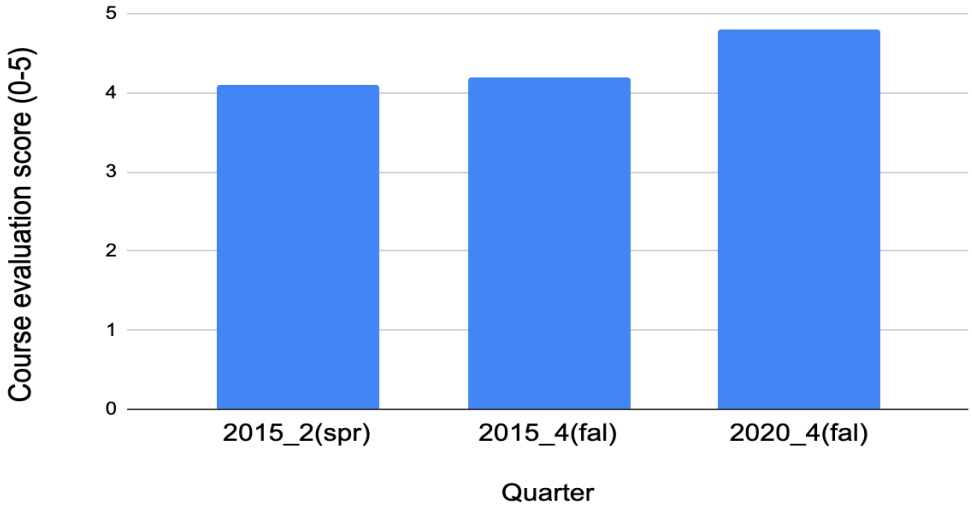

Figure 12: Course evaluation scores before actively using chat for polls (the two 2015 classes), and after (2020) , for the same class and same instructor.

9. Conclusion

Conducting polls during lectures can help engage students. But, for a dynamic class, using tools like zoom's built-in polling tool, is not ideal due to requiring that questions be pre-designed, or due to slowing class to design questions on-the-fly. We found that using the chat box for polls was quite effective, and we encourage such use. To help instructors give participation credit for chat, we developed a tool to automatically count per-student chat messages from zoom chat logs. And, we developed heuristics to automatically detect polls in the chat log and approximately classify messages as poll/non-poll, which can further help instructors gain insight into their poll usage and help train themselves to make best use of polls throughout class. Future work will aim to gain even more accuracy in counting polls and classifying poll messages, perhaps even tabulating responses, and to provide more advanced ways of giving participation credit to students. We intend to make the tool freely available via

the web to educators. We have also implemented a Google Sheets add-on script version of tool, enabling increased privacy plus ability for users to modify the sheet or script for customized calculations.

10. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2111323.

REFERENCES

[ArKr17] Arthurs LA, Kreager BZ. An integrative review of in-class activities that enable active learning in college science classroom settings. International Journal of Science Education. 2017 Oct 13;39(15):2073-91.

[Fe16] Felder RM, Brent R. Teaching and learning STEM: A practical guide. John Wiley & Sons; 2016 Mar 7.

[Ka15] Kappers WM, Cutler SL. Poll Everywhere! Even in the classroom: An investigation into the impact of using PollEverwhere in a large-lecture classroom. Computers in Education Journal. 2015;6(20):21.

[Ic21] iClicker. https://www.iclicker.com/

[Jo76] Johnstone AH, Percival F. Attention breaks in lectures. Education in chemistry. 1976;13(2):49-50.

[Me93] Meyers C, Jones TB. Promoting Active Learning. Strategies for the College Classroom. Jossey-Bass Inc., Publishers, 350 Sansome Street, San Francisco, CA 94104; 1993.

[Mi96] Middendorf J, Kalish A. The “change-up” in lectures. In The national teaching and learning forum 1996 Jan (Vol. 5, No. 2, pp. 1-5).

[ShWa09] Shen R, Wang M, Gao W, Novak D, Tang L. Mobile learning in a large blended computer science classroom: System function, pedagogies, and their impact on learning. IEEE Transactions on Education. 2009 Aug 4;52(4):538-46.

[St15] Stowell JR. Use of clickers vs. mobile devices for classroom polling. Computers & Education. 2015 Mar 1;82:329-34.

[To21] TopHat. https://tophat.com/

[VaAl20] Vahid F, Allen JM. An online course for freshmen? The evolution of a successful online CS1 course. In2020 First-Year Engineering Experience 2020 Jul 26.

[WaLi14] Watson C, Li FW. Failure rates in introductory programming revisited. 2014 conference on Innovation & technology in computer science education 2014 Jun 21 (pp. 39-44).

[WhRa10]] Whitehead C, Ray L. Using the iclicker classroom response system to enhance student involvement and learning. Journal of Education, Informatics and Cybernetics. 2010;2(1):18-23.