Motivated Explaining and Summary Writing – Does It Help?

Tianyi Li
University of Toronto
tianyidoudou.li@mail.utoronto.ca

Michelle Craig
University of Toronto
mcraig@cs.utoronto.ca

ABSTRACT

Videos and exercise problems are often the core contents of an online education tool, such as in the Programming Course Resource System (PCRS) in University of Toronto. However, the design of questions is more important in terms of encouraging students to actively interact with the learning materials.

Studies suggest that retrieval practice can help students gain more insight into the topics they are studying. In this paper, we design three sets of exercises and assigned them to three separate treatment groups in a controlled environment to analyze whether if programming students who write summaries of what they learned improves the quality of studying.

The sets of exercises analyzed included multiple choice questions and short answer questions, designed to measure the understanding of C signal handling through pre- and post-test questions. The results show that writing summaries had no effect on learning in a mark-on-completion environment.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Computer science education**;

KEYWORDS

signals handling; retrieval practice; C language

ACM Reference Format:
Tianyi Li and Michelle Craig. 2021. Motivated Explaining and Summary Writing – Does It Help?. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Many computer science students struggled during their introductory programming courses, raising concerns to computer science education researchers. The reasons behind this struggle can be very complex to identify and analyze. One would argue that it is because programming languages are built progressively and involve multiple areas of knowledge. On the other hand, the assessment questions may fail to reveal how much students truly understand on the topics and the questions may expect too much from the students.

Nowadays, online education tools have become strongly entrenched in the educational landscape, since they are easily accessible and students can attempt the exercises whenever they are prepared. In addition to the convenience of students, these open online tools have also opened up a new frontier for computer science researchers, as we can track the behaviors and feedback of students in an uninterrupted manner.

In most online educational tools, it is often that there is no feedback for students [3]. The instructional design of online exercises is an essential factor which contributes to students learning qualities [2]. Similarly, using online education tools as a platform to study learning processes has come with its difficulties. Many more research opportunities and topics have presented, yet they also bring in challenges of randomness, validity and generalizability.

In this paper, we present a study which focuses on educational tools which improve student learning gains through retrieval practice (remembering information and writing summaries). In this study, retrieval practice refers to after every videos been watched by the learners, they are asked to summarize the contents of the videos. Previous research suggested that the performance of a retrieval test led to a better learning outcomes comparing to simply re-watching the video or underlining key phrases [2].

2 RELATED WORK

Similar work has been done across disciplines on how retrieval exercises may help or hinder student learning. We have looked through three pieces of related works and provide brief summaries to each of them here; more detailed information on the results of these related works can be found in the references in the last page of this paper.

In the paper regarding how writing and reading summaries of videos affects student learning [2], two studies were conducted by T. Zee et. al regarding the effects of prompting learners to either read or write a summary after every instructional video in a Massive Open Online Course. The results show that learners who wrote or read more summaries of the videos also scored higher on the weekly quizzes. Reading more summaries in the first two weeks of the course also significantly predicts a higher grade on the final quiz, while this is not true for writing more summaries. Surprisingly, writing summaries of videos does not appear to lead to better quiz scores than passively reading provided summaries. In addition, both studies fail to provide evidence for the hypothesis of whether if the students would actually benefit more from writing or reading summaries.

On a separate study [1], an experiment was conducted in a CS1 course with two campuses of the same university, and studied the impact of contexts in a controlled experiment. On campus A, the problems did not contribute to the final marks, and on campus B, the problems would weight for 0.5% of the course grade. The study outcome shows that campus A consistently took fewer attempts than students from campus B, since only more motivated students would want to finish the problems, and the students tended to extract information from the function parts first, rather than the problem description. As a result, the authors did not think that the use of familiar contexts in exercises is harmful, yet they do believe even a small change in the description of a problem could have a significant impact on its difficulty.

The last study that came across us focused on finding which reflective prompts are the most effective at promoting belief revision and guiding online learners to revise prior misconceptions, even when feedback from an instructor was unavailable [3]. Every participants would receive and read through a problem and its solution first, and complete either a "prompts to explain" or "prompts to write thoughts" question. The result of the study shows that the "prompts to explain" questions were significantly more effective than "prompts to write thoughts" questions. However, the effectiveness of prompts depends both on the design of the prompts and its target.

3 METHOD OF STUDY

3.1 Course Selection

We selected the course "CSC209: Software Tools and Systems Programming" from University of Toronto, to conduct our research. CSC209 is a programming course which focuses on software techniques in the Unix environment using language C. Students in this course have learned to use at least 2 prior programming languages (Python and Java), in at least 3 prior computer science courses and are intending to pursue computer science as a degree.

The course is using an online tool called PCRS to give student preparation exercises every week before lectures. The exercises are typically a combination of videos, multiple choices and short answers. 5% of the total grade in the course goes to the completion marks of these preparation exercises from each weeks. The study took place in the preparation exercise for Week 9, on the topic of Signal Handlers.

3.2 Participants

A total of 230 students participated in the study. To validate the equivalence, the students were assigned to three treatment groups based on their enrolling sessions for the course. The three sessions are randomly assigned to be different treatment groups. There are about 80 students in each of the groups.

3.3 Pre- and Post- Tests

We first introduced the basic concepts that the students need to master in the topic through a Youtube video created by the CS department staff. Then, we created the equivalent pre- and post-test questions that will be used to measure student performances.

The paired pre- and post-test questions were also in disguise, such that the students would not recognize and attempt to complete them by remembering answers or patterns. All three treatment groups received the same pre- and post-test questions, while they received a different "Hypotheses Question" ("Signals Follow-up Question Tracing: part 5", Fig. 1) depending on their groups.

Signals Video 1: Introduction to Signals
https://www.youtube.com/watch?v=3KkLUe5ofM0

Follow-up: Question 1
Select all the true statements about signals.

a. A signal can only be sent to a process if that process is a child or parent of the sending process.
b. A signal can be sent by the operating system or a user process.
c. The kill program is used to terminate a process.
d. The kill program is used to send a signal to a process.
e. A programmer can't know when a signal might arrive at a process (unless the programmer deliberately creates a segmentation fault.)
f. A signal can only be sent to a process writing to stdout.

Signals Video 2: Handling Signals
https://www.youtube.com/watch?v=Bx5LqlhEzfM

Signals Follow-up Question Tracing: part 1-5

```
int x = 5;
void handler(int sig) {
    x += 3;
    fprintf(stderr, "inside %d ", x);
}
int main() {
    fprintf(stderr, "start ");
    //                             POSITION A
    struct sigaction act;
    act.sa_handler = handler;
    act.sa_flags = 0;
    sigemptyset(&act.sa_mask);
    sigaction(SIGINT,&act,NULL);
    //                             POSITION B
    x += 2;
    //                             POSITION C
    fprintf(stderr, "outside %d", x);
    return 0;
}
```

1. What does the program print to stderr when it runs without?
2. What does the program print to stderr if the user presses Ctrl+C at the moment when the program is at position A?
3. What does the program print to stderr if the user presses Ctrl+C at the moment when the program is at position B?
4. What does the program print to stderr if the user presses Ctrl+C at the moment when the program is at position C?
5. *Hypotheses Question.*

Follow-up: Question 2
Select all the true statements about signals.

a. The sigaction system call is used to change the code that is executed when a signal arrives at a process.
b. A signal handling function that you write must always have the same signature.
c. If a SIGINT signal arrives before the call to sigaction, it will still execute the handler function that we wrote.
d. The signal handling function that we install will cause the process to terminate, even if it doesn't call exit.
e. The default action of a SIGKILL signal can never be modified.
f. "Installing a signal handler" means writing a function with the appropriate signature and calling sigaction with the function pointer as an argument (a field of struct sigaction *).

Figure 1: Design of the three exercises.

Signals Follow-up Question Tracing: part 5

Group 1. *None.*

Group 2. For each of the 4 previous scenarios, write one sentence that explains why the program generates the output that it does.

Group 3. For each of the 4 previous scenarios, write one sentence that explains why the program generates the output that it does. We aren't going the quality of your explanations, but we believe that writing them will help you learn this material and make the later questions (here and on the exam) easier.

Figure 2: Hypotheses Questions.

Measurements of Data

M1. Measure the average number of attempts until a student answer the entire question correctly.

M2. Measure the average number of attempts until a students answer part of the post-test question correctly (of the 6 choices in the post-test, only a, c, d, f were directly related to the information provided from the 2 previous videos).

Figure 3: Measurements of Data.

3.4 Design

This study was conducted during the preparation exercises for week 9 on "Signal Handling". All three exercises start with a video on the introduction to signals and followed with "Follow-up: Question 1" (Fig. 1), and we used this question as our *pre-test part 1.*

Then, the participants watched the second video on the basic concepts of signal handling followed with four code-tracing short answer questions ("Signals Follow-up Question Tracing : part 1-4", in Fig. 1). The combination of these four questions will be our *pre-test part 2.*

Afterwards, the participants received the hypothesis questions below, designed for their treatment groups. The final question is "Follow-up: Question 2" (Fig. 1), and that will be the *post-test* measurement for this study.

The two "Follow-up" multiple choice questions, each consist of six choices and the participants were required to select all the true statements for the questions. The code-tracing short answer questions are all based on one piece of code (Fig. 1), and the students were asked to fill in the correct outputs for each of the scenarios.

All the instructions of the questions were on the online exercise page. The students were expected to be able to proceed without other directions.

3.5 Hypotheses Questions

For the three sets of exercises, the only variation among them is the design of the hypotheses questions (Fig. 2).

The first group, which was the designated *control* group of this study, did not receive Question 5. Group 2 was the *explain* group, and received the "explain by summary" Question 5. The last group was the *motivate* group, and received a "motivate to write summary" Question 5.

Although, we did not have any biased expectations on the effects of these hypotheses question in the exercises, we were hoping the results would show us that retrieval practices benefit students' learning.

		Control	Explain	Motivate
	# of students	72	80	78
M1	Avg. for pre1	4.8194	5.1125	4.6026
M1	Avg. for post	5.4167	4.4875	4.4231
M2	Avg. for post	3.9167	3.4375	3.4103

Table 1: Average number of attempts until correct for pre-test part 1.

3.6 Method of Study

We decided to use two measurements to look at the equivalence of the treatment groups and to study the resulting student data from the pre- and post-tests questions (Fig. 3).

M1 is valid for both pre- and post-tests questions, while on the other hand, M2 is only valid to measure the post-test question.

4 RESULTS

The multiple choice questions used the "select all that apply" policy, rather than the more common "select the correct answer". The students were allowed to attempt to answer each question as many times as they wanted. The PCRS system gave instant feedback on how many of their inputs were correct, but not which ones.

The short answer questions required the student to fill in the valid outputs to the program. The system responded whether if the input was a compilation error or if the code compiled, and also if any warnings.

All of the student attempts were captures by the PCRS system and were exported as a few .csv files to researchers for further analysis.

Our initial data set contained 8,390 solution attempts from 230 participants. They include both multiple choice and short answer attempts. For the purpose of the two measurements, only the attempts that the students made before they answered the questions correctly will be used. The attempts they made after they succeed the questions were not included in the analysis of this study.

Note: it was not necessary for the students to complete all the questions in their exercises. Therefore, they may have chosen to answer one, a few, or all of the questions prepared for them.

Table 1 shows the average numbers of students attempts until correct for pre-test part 1.

Table 2 shows the average numbers of students attempts until correct for pre-test part 2.

Table 3 shows the analysis of the written summary responses.

4.1 Analysis of Means for Pre- Part 1

The "Avg. for pre1" row in Table 1 shows the mean number of attempts per student to answer pre-test part 1 ("Follow-up: Question 1", in Figure 1) until correct. This is supposed to be the measurement for the equivalence of the three groups. We can see that these numbers are not closed enough to be considered the same.

The "M1: Avg. for post" row shows the mean number of attempts per student to answer post-test part 1 until all the choices were answered correctly. The "M2: Avg. for post" row shows the mean number of attempts per student to answer post-test part 1 until 4 of the choices were answered correctly. A higher number of attempts means that the group found the problem to be more challenging.

		Control	Explain	Motivate
	# of students	71	89	76
M1	Avg. for pre2 q1	4.4225	5.0	5.5395
M1	Avg. for pre2 q2	1.3521	1.6962	1.6711
M1	Avg. for pre2 q3	10.5352	12.6835	15.4211
M1	Avg. for pre2 q4	3.9718	3.5823	8.6184
M1	Avg. for all 5 pre	5.0202	5.6149	7.1705

Table 2: Average number of attempts until correct for pre-test part 2.

First, we notice that students from Group 1 (*control* group) took more attempts to answer the post-question. On the other hand, Group 2 (*explain* group) and 3 (*motivate* group) took fewer attempts in comparison to pre-question part 1. Even though the difference is relatively small, we can still see that the hypothesis questions might be helping the students prepare more for the post-question. It is likely that writing a summary helped them go through the logic repeatedly in a bigger picture, instead of just knowing the correct responses to particular scenarios.

Then, if we take a look at M2 and focus on the correctness of the choices related to the videos the student watched earlier, we can see that the results have improved significantly. The average number of attempts dropped from 5.4167 to 3.9167 in the *control* group and from 4.4 to 3.4 to the *explain* and *motivate* groups for the post-test questions.

4.2 Analysis of Means for Pre- Part 2

The first 4 "M1" rows in Table 2 shows the mean numbers of attempts per student to answer pre-test part 2 ("Signals Follow-up Question Tracing : part 1-4", Fig. 1) until correct. The last "M1" row represents the mean number of attempts per student to answer all 5 pre-test questions.

We see that the data has been very unstable across all 3 groups. However, tracing question 3 appears to be the hardest question among all in all groups; it takes double or triple the amount of attempts compared to the other questions. Also, tracing question 4 has proven difficult specifically for the *motivate* group.

4.3 T-test and P-values

We used Welch Two-Sample T-tests to compare the data between the different treatment groups. Unfortunately, in all the cases, we found no statistically significant differences (p-value < 0.05).

We believe that since the exercises are marked based on completion instead of correction, some of the students may not have taken the questions seriously enough, resulting in inaccurate data.

	Explain	Motivate
# of students	76	74
Avg. # of attempts	1.684	1.284
# of serious attempts	88	71
# of random attempts	40	24

Table 3: Analysis of the Written Summary Responses.

4.4 Analysis of Written Summary Responses

In addition to above, we also studied the student responses from the hypotheses questions. When compared to an *explain* question, the *motivate* question did not seem to encourage more the students to write a summary.

One surprising result is that about 95% of the students answered the questions, which is more than expected, and about 68.8% and 74.7% (for the *explain* and *motivate* groups) attempted serious answers, instead of random place-holders to get participations marks.[1]

5 CONCLUSIONS

We have analyzed a large sum of student attempts to Signal Handling problems, and hoped that by using retrieval practice (writing summaries), the students would benefit in gaining new programming knowledge. However, the current results do not support our expectations suggesting, that there may be other factors which have a larger impact on the difficulty of programming questions.

5.1 Future Work

Since these preparation exercises were marked based on completion instead of correction, some of the students may not have taken the questions seriously, resulting in inaccurate data. We encourage other researchers to repeat similar studies in a more controlled, formal environment, such as quizzes or tests, and gather valid data to further analysis.

REFERENCES

[1] M. Craig, J. Smith, and A. Petersen. Familiar contexts and the difficulty of programming problems. 2017.
[2] T. van der Zee, D. Davis, N. Saab, B. Giesbers, J. Ginn, F. van der Sluis, F. Paas, and W. Admiraal. Evaluating retrieval practice in a mooc: How writing and reading summaries of videos affects student learning. 2018.
[3] J. J. Williams, T. Lombrozo, A. Hsu, B. Huber, and J. Kim. Revising learner misconceptions without feedback: Prompting for reflection on anomalies. 2016.

[1]We considered all the relevant responses as a seriously attempt. We also marked all the random characters, repetition of a previous solutions, copies of a question title, and etc. as place-holders.