automaTA: Human-Machine Interaction for Answering Context-Specific Questions

Changyoon Lee*, Donghoon Han*, Hyoungwook Jin*, Alice Oh
KAIST, Daejeon, South Korea
{cyoon47, hoonhan.d, jinhw}@kaist.ac.kr, alice.oh@kaist.edu

ABSTRACT

When online learners have questions that are related to a specific task, they often use Q&A boards instead of web search because they are looking for context-specific answers. While lecturers, teaching assistants, and other learners can provide context-specific answers on the Q&A boards, there is often a high response latency which can impede their learning. We present automaTA, a prototype that suggests context-specific answers to online learners' questions by capturing the context of the questions. Our solution is to automate the response generation with a human-machine mixed approach, where humans generate high-quality answers, and the human-generated responses are used to train an automated algorithm to provide context-specific answers. automaTA adopts this approach as a prototype in which it generates automated answers for function-related questions in an online programming course. We conduct two user studies with undergraduate and graduate students with little or no experience with Python and found the potential that automaTA can automatically provide answers to context-specific questions without a human instructor, at scale.

Author Keywords

Context-specific learning; question answering; programming learning; human-machine interaction

INTRODUCTION

In online learning where learners have little real-time feedback and direct interaction with the instructors, getting an answer for a question quickly to deal with issues they face is essential for the learners [3]. Thus, searching and finding answers online for immediate clarification can improve the learning experience. However, when the learners' questions are not about a general concept or knowledge but rather about a specific course task or homework, it is likely that only a few people have asked the same questions before. In other words, the answers they can find through a web search may not be relevant to their questions. In such cases, learners post the questions in the course's question board to receive an answer from the teaching assistant or the instructor. The learners can then get the correct and relevant answers specific to the context, but the high response latency of the instructors can impede learning.

To deal with context-specific questions arising from online education settings quickly and accurately to help online learners, we suggest a human-machine mixed approach that collects question-answer data for each course and automatically suggests appropriate answers to the learners' questions, inspired by Zensors [5]. This approach makes automation possible through machine learning with data created by human workers. When there is a context change and the machine is unable to answer, the suggested approach gives the control back to the human to maintain high accuracy and relevance of the answers while simultaneously training the machine for the new context. We believe that this approach can deal with unanswered, course-specific questions, and train the machine learning model to generate automated answers.

To explore the potential of this approach, we conduct two user studies with automaTA, a prototype that suggests context-specific answers to learners' questions in programming by capturing the context of their questions.

The first study aims to evaluate the usability and functionality, and automaTA received an average satisfaction score of 4.0 (1: Not satisfied, 5: Satisfied). The second study aims to collect question-answer data and train automaTA without any initial data, and evaluates the trained system's usefulness and learners' satisfaction. Participants reported 10 automated answers to be useful and gave a mean score of 4.0 for their satisfaction with the answers (1: Not satisfied, 5: Satisfied).

Our contributions are as follows:

- We show the potential of the human-machine mixed approach for context-specific question answering in an online course, applicable at scale.
- We present a prototype of automaTA, a system which suggests automated answers by training a machine learning model using context-specific answers from human teaching assistants.

* Authors contributed equally.

L@S'19, June 24–25, 2019, Chicago, IL, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

BACKGROUND

Answering Context-Specific Questions

Learners often bring context-specific questions to their instructors. For example, in programming, Jonathan Sillito et al. [6] has shown that many questions need to be provided with their context to get adequately answered. To deal with such questions, instructors first need to figure out the context of the question or infer if it is not explicitly given. Imitating human instructors, we propose the idea of inferring the context of learners' questions first and providing answers by utilizing the questions' context such as code history.

Human-Machine Interaction

Traditional machine learning approaches usually do not perform well without sufficient data. Instead, users' active participation in human-machine mixed approaches can improve machine learning-based systems [1], as it reduces the effort of collecting qualified data to train a model. For example, Zensors by Gierad Laput et al. [5] suggests a human-machine mixed workflow that allows for the construction of an automated system which can handle new or unexpected situations. By turn-taking between human workers and machines, high accuracy can be maintained throughout the training process. We adopted the approach from Zensors by allowing learners to ask human teaching assistants directly when automated answers by the system are not satisfactory. Teaching assistants' answers are then used to train the machine so that the system improves over time to provide better suggestions automatically.

SYSTEM

We implemented a prototype, automaTA, to evaluate the feasibility of a human-machine mixed approach to automate question answering in the context of programming learning. automaTA is a Google Chrome extension that allows students to ask questions and receive automated answers on their programming task window. automaTA adds to the rich features built in Elice, an online Python learning platform [4]. automaTA provides three key functions to support the time-effective and satisfactory question-answering process of self-studying learners.

Inferring the Context of Questions

automaTA has a search bar in which learners can enter free-form questions (Figure 1(a)). When a learner wants to find a function that performs a particular action (e.g., open and read a file), the learner describes the action as far as s/he can and asks on the search bar. Then, automaTA suggests three functions that are most relevant to the question and the programming task the learner is solving. The question does not need to be a complete sentence as automaTA extracts and evaluates keywords from the question. automaTA can also handle sloppy questions, from which learner's intention and context are hard to be inferred, by checking what functions the learner's code lacks, compared to peer codes. For example, suppose a learner asks "how could I get data from a txt?". Although the read() function would be the most appropriate answer, the words used in the question are not directly relevant to the function. automaTA checks and lists the functions used in the learner's current code. If the learner has not used read() while other

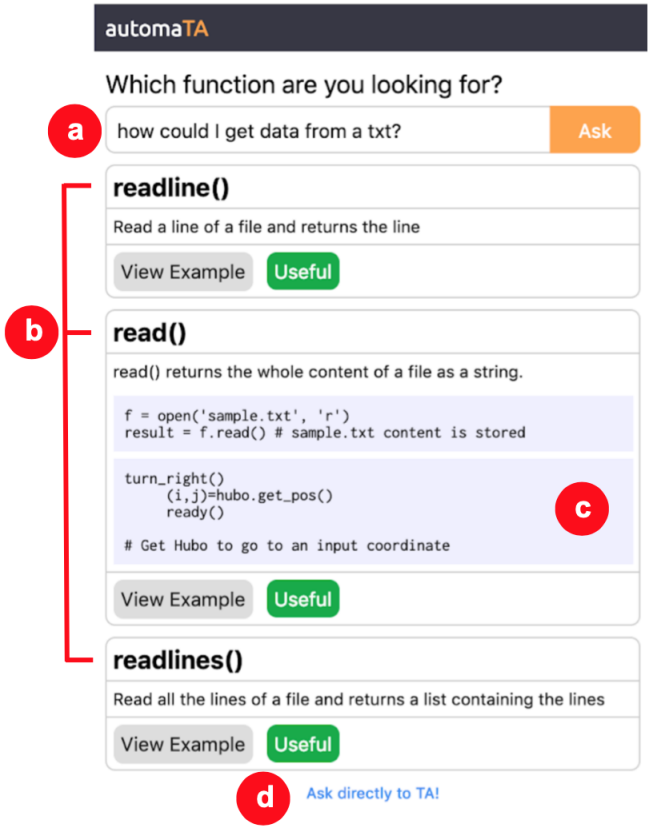

Figure 1. (a) Learners ask for function suggestions by describing the function they are looking for. (b) automaTA suggests three relevant functions for the description. Learners can click the 'Useful' buttons if the suggestion is correct and helpful. (c) Code examples from the official Python documentation and peer code are provided. (d) Learners can ask questions directly to teaching assistants if the automated answers are unsatisfactory.

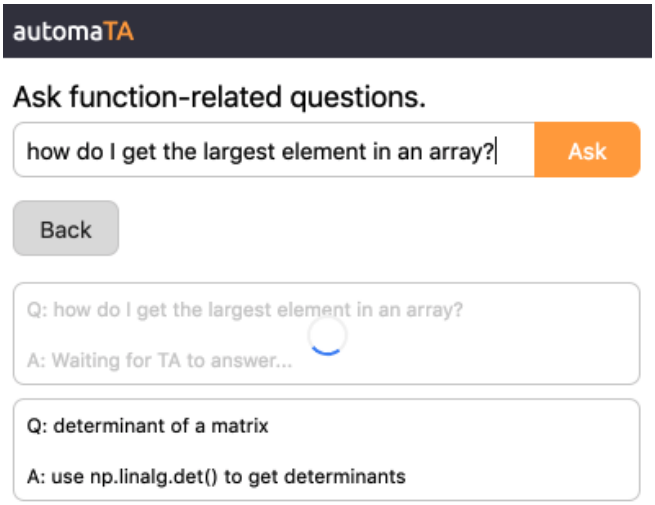

Figure 2. Learners can ask questions directly to teaching assistants. The answers to the questions may not be provided immediately as teaching assistants answer them manually. Questions are stacked from the bottom so that the learners can check and see their previous questions.

previous learners frequently use it in their codes, automaTA suggests using read().

Training Model for Question Answering

Learners can click the 'Useful' button to indicate the usefulness of a particular function suggestion. Clicking this button updates the rank of the function for an input question from a user so that it is more likely to appear for similar questions in the future. The concept of matching users' high-level descriptions to specific functions has been introduced in previous research. The query-feature graph maps a search query describing the user's goal to specific features in a system [2]. In automaTA, we adopted this approach and constructed a context-function graph to generate automated answers for programming beginners' sloppy questions. Our context-function graph captures question context by evaluating the learner's question, code, and peer code. The graph maps learners' context to a finite set of functions used in programming. The 'Useful' button clicks and teaching assistants' question-answering change the mappings dynamically by strengthening associated mappings. With question keywords, code, and peer code information, our context-function graph computes relevance scores for every function and returns the top three relevant functions.

Providing Satisfactory Answers

automaTA provides satisfactory answers by turn-taking between humans and machines. In the case when the automated answers by automaTA are not relevant for a question, learners can ask the same question directly to a human teaching assistant (Figure 2). This workflow ensures that learners can get satisfactory help even when automaTA is not mature enough to give useful answers. When a learner asks a question to a teaching assistant, learner's code and task information are sent to the assistant along with the question. The teaching assistant uses an independent interface to answer the question. The teaching assistant suggests a list of functions and writes an explanation of how those functions can help solve the learner's task. Question and answer data are then used to improve automaTA to give better answers.

EVALUATION

User Study 1

To evaluate the usability and functionality of automaTA, we conducted a user study with five participants. We evaluated the user satisfaction of our system compared to that of online search.

Participants

Participants were undergraduate students who have either only taken introductory Python programming courses or have not taken any. None of the participants major in computer science. At the beginning of the study, the participants were given a short tutorial on Python programming and basic function usages to simulate the usual programming practice environment.

Study Design

The study was a 2-by-2 within-subject study. The participants were assigned to four conditions of different task order and system usage between web search and automaTA. We treated usage order as an independent variable to compensate for the learning effect between the information search tools. We gave our participants two programming tasks—one which uses the cs1robots library, a customized library used in our university's introductory programming course, and another which uses Python built-in functions. The tasks were independent as they do not share functions or algorithms to complete. Participants completed a questionnaire at the end to rate system usability and satisfaction.

Data Collection

We conducted a study beforehand to collect question-answer data. We recruited six programming beginners on campus. We received 33 function-related questions and trained the query-feature graph with these questions. Since the size of the data collected is too small and may not represent the actual distribution of questions and keywords, two authors who have participated as teaching assistants in the introductory programming course added more question-answer data based on their experience.

Result

Participants rated the usefulness of automaTA's function explanation and code examples as 3.6 and 3.0, respectively (1: Not useful, 5: Useful). For the question "how far do you agree that automaTA helps completing tasks more than online search does?", participants rated 3.0 on average (1: Strongly disagree, 5: Strongly agree). The participants' average satisfaction of the system was 4.0 (1: Not satisfied, 5: Satisfied) and three out of five participants answered that they are likely to reuse the system for their learning. One participant commented that, "Once I get well-acquainted with the system, the function suggestion will help a lot."

User Study 2

Since the first study showed the potential that automaTA can give satisfactory answers when trained with enough question-answer data, we ran another study with nine programming beginners to check if data collection can be achieved with our human-machine mixed approach. The first phase of the study aimed to collect question-answer data to train the system with no data, and the second phase of the study aimed to evaluate the quality of answers that are suggested from the data collected in the first phase.

Participants

We recruited undergraduate and graduate students with little to no experience of using the Python programming language. The self-reported score for level of understanding on Python was 1.6 on average (1: Low, 5: High). None of the participants majored in computer science, and six participants out of nine have never taken a Python programming course before.

Study Design

At the beginning of the study, we gave a short tutorial on Python programming and basic NumPy library usage. Then, they were asked to complete five programming tasks within an hour. The programming tasks given to the participants were chosen to be easy to develop an algorithm but challenging to translate the algorithm into a program. These tasks were chosen because we expected the participants to ask questions

related to function usage which automaTA aims to answer automatically. During the tasks, participants were not allowed to perform online searches or refer to external materials, but they were encouraged to use automaTA to ask questions to simulate a context-specific situation where no answer can be found online. We intentionally removed example codes from the system in order to train it with a more diverse question-answer dataset, including function usage questions. Participants completed a questionnaire at the end of each phase to rate system usability and satisfaction.

Phase 1. Three participants were assigned to this phase. Since automaTA did not have any data to start with, participants could not receive automated answers. We received all the questions to answer manually and to train automaTA with the question-answer data. 15 questions were asked and answered, and all question-answer data were used to train the query-feature graph of our system.

Phase 2. Six participants were assigned to this phase. Since we trained the system with question-answer data from the first phase, the participants in the second phase could receive automated answers. Participants could still ask questions to the human TAs through the system when automated answers were not satisfactory or whenever they desired to.

Result

A total of 10 automatically suggested functions were reported to be useful by the participants. A total of 35 questions were asked to the human TAs to answer manually. Participants gave a mean score of 4.0 for their satisfaction with the answers (1: Not satisfied, 5: Satisfied) and commented that the answers were appropriate and immediate. Participants reported a mean score of 3.7 on whether they would be willing to use automaTA again when they learn programming (1: Not willing, 5: Willing).

DISCUSSION

Two studies showed the potential of applying human-machine mixed approach to train and automate a question answering system in an online course. In the first study, the system was able to provide satisfactory answers specific to different tasks. In the second study, the system was able to provide 10 useful suggestions even when there was no relevant data available initially, after being trained with the questions from 3 participants in the first phase of the study.

Limitation and Future Work

However, the current system has some limitations at the moment. Firstly, while our current suggestion algorithm could provide satisfactory answers to the participants, a more sophisticated algorithm for training query-feature graph and for suggesting answers can improve the accuracy of the system. We are planning to deploy our system at scale to evaluate and train our system. Secondly, we suspect some usability concerns in the system that can lead to inaccurate data collection. Whenever the learner finds a good suggestion from the system, the system only receives the information when the learner clicks on the 'Useful' button. However, it is possible that the learner does not click the button. This phenomenon will cause under-training of the system. A more user-friendly and accurate method to capture the system's successful suggestions should be devised. Lastly, we only tested the system and the human-machine mixed approach in the domain of programming learning. Exploration of the approach in other educational domain should be done.

ACKNOWLEDGEMENT

This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921) and by the MSIT (Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW (2016-0-00018), supervised by the IITP (Institute of Information & communications Technology Planning & Evaluation) (2016-0-00018).

REFERENCES

1. Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.
2. Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-feature graphs: bridging user vocabulary and system functionality. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 207–216.
3. Kyong-Jee Kim, Shijuan Liu, and Curtis J Bonk. 2005. Online MBA students' perceptions of online learning: Benefits, challenges, and suggestions. *The Internet and Higher Education* 8, 4 (2005), 335–344.
4. Suin Kim, Jae Won Kim, Jungkook Park, and Alice Oh. 2016. Elice: an online CS education platform to understand how students learn programming. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. ACM, 225–228.
5. Gierad Laput, Walter S Lasecki, Jason Wiese, Robert Xiao, Jeffrey P Bigham, and Chris Harrison. 2015. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1935–1944.
6. Jonathan Sillito, Gail C Murphy, and Kris De Volder. 2008. Asking and answering questions during a programming change task. *IEEE Transactions on Software Engineering* 34, 4 (2008), 434–451.