Emerging Frontiers in Virtual Learning with Code in Place: An Interview with Professor Mehran Sahami

Allie Lee
Stanford Univeristy

Professor Mehran Sahami, the Associate Chair for Education at Stanford University's Computer Science Department, is widely recognized for his work in machine learning and higher education. In addition to B.S. and M.S. degrees, he earned his Ph.D. in Computer Science at Stanford before returning as faculty in 2007. Professor Sahami's diverse background includes senior roles at Google and Epiphany, and he currently serves as the Robert and Ruth Halperin University Undergraduate Education Fellow and the Association for Computing Machinery Education Board Co-Chair. Professor Sahami has published several research articles on topics that range from student learning to information Web retrieval and coauthored the book *Text Mining: Classification, Clustering, and Applications*. During this interview, Professor Sahami elucidates his perspectives on virtual learning and Code in Place, a tuition-free 5-week Python programming course based on Stanford's introductory programming class, CS106A, that was offered online in Spring 2020 at the onset of the COVID-19 pandemic.

AL: At the start of the pandemic, you led Stanford's CS Department in offering Code in Place. Can you please tell us more about your inspiration for Code in Place related to your greater goals in education?

MS: Sure! Chris Piech (an Associate Computer Science Professor at Stanford) and I were thinking about what we could do with a pandemic going on and where we might be able to help. With people sheltering in place, we aimed to provide an educational opportunity that involved our section leaders, as that human component often makes educational experiences more successful. We both have the goal of using education as a personally and socially transformative mechanism to create more opportunities. But regarding logistics, Chris was actually the driving force behind the scenes. He helped coordinate with the University, recruit section leaders, and get the word out so more students could apply.

AL: Do any moments of Code in Place stand out to you as particularly exciting or inspiring?

MS: Yes, I think there are a number of moments, beginning with the initial response in terms of section leaders. We wanted to involve them before determining how many students we could accept because it was really a matter of the student-section leader ratio. So having that many people—which was more than we expected—volunteering their time was truly inspiring. The number of student applications (approximately 80,000 applications were started on the program website) was also inspiring yet scary, because there was far more interest in Code in Place than we could accommodate; but just seeing the number of people that we could potentially reach was really heartening.

Another element was the program's positivity. If you looked around the Ed Discussion Forum**,** people constantly responded to questions and supported one another—which is not the kind of experience you would expect in online forums, as they tend to be much more negative and sometimes toxic. I spent a fair bit of time, actually, on the Ed Forum reading and responding to students. There was so much energy that motivated everyone to keep investing more and more into Code in Place.

AL: Outside of academia, you served as a Senior Research Scientist at Google before returning to Stanford as faculty. How has your industry background influenced your perspective on this undertaking and as a professor?

MS: With teaching in general, I often focus on relevance outside of academia by asking: What are ongoing trends? What are interesting applications? What are ways we can inspire students with parallels between skills and potential practices? For example, something that we couldn't cover in Code in Place due to time constraints—but did in CS106A—was writing lightweight search engines. In one of my lectures, I pulled up an old Python code from the original version of Google, which was developed as part of the Stanford Digital Libraries Project when I was a graduate student. It is always exciting when students can understand a code's function because not only have they learned a concept, but also its possible applications.

AL: In addition to CS106A, you instruct a wide variety of CS courses ranging from ethics to AI. How have these experiences shaped the direction of Code in Place, and how might Code in Place impact future teachings?

MS: There's always a bi-directional learning experience in the sense that when I teach a class, I can gain insights into other classes that help me reevaluate my instruction. In my ethics class, a recurring topic is the notion of accessibility—who is able to participate and how—and that resonated with Chris and I during Code in Place. We thought that especially during a challenging time where there's also a lot of economic uncertainty, something like Code in Place could potentially be a pathway to different career options.

The flip side was seeing the overwhelming response, students' interests, and ways we could continue making learning more accessible. I've been focusing on ways we can teach material and reach people outside of Stanford with Rob Reich and Jeremy Weinstein, who are both professors here in the Political Science Department. It kind of works both ways.

AL: For me and many other students, Code in Place was our first experience with CS, a field that often appears daunting from the surface. What did you hope that students took away from the course, and what kind of feedback did you receive?

MS: In terms of the role of computing, we understand that for different people, there are different possibilities. Code in Place was really an invitation, an invitation to consider computing as a discipline, a field they might pursue, a new career opportunity, or just knowledge that's useful in life and presents a new way of thinking.

In that sense, sometimes it's weird to categorize CS as a liberal education. But that's often how we view it. You can think about computing as a tool or mind frame that is useful across a variety of disciplines and expands the scope of how problems are solved. For some people, it might mean a whole career. But in other cases, it can be how we analyze text and inform medicine. There are many fields that can benefit from CS and help inform it, and right now we are working to broaden CS so it can better integrate with other disciplines.

One piece of feedback we get from a lot of Stanford students when they take their first CS class is comments of the form: "I never thought I was going to take a CS course and didn't see how this was relevant to me, but after I took it I'm really intrigued and kind of want to learn more about how this field might be useful." And it's not necessarily that they want to be a CS major, but they understand that there's a certain power to CS and a certain way of thinking that can be useful in a variety of ways.

AL: Code in Place brought together around 900 volunteer educators and 10,000 students from around the world. What kind of challenges did you overcome while working at this scale, and is there anything that you would have done differently?

MS: To start with differently, I think that after going through the process once—because this was all new—there is a better sense of what we should've done earlier and how we could've organized more efficiently. A lot of this stuff we were building as we went along, like assembling an engine with the plane in flight. And sometimes that's a pretty scary experience when you don't get things quite right and need to backtrack and change things.

But when you have that many section leaders and learners, you need to understand that there are a lot of diverse starting points and ending points—people who come from different backgrounds and have distinct levels of experience, challenges, and availability. And the things they might want to get from that class vary. So there isn't a one-size-fits-all model, but we wanted to create a broad enough experience where we could bring in a lot of people and assume no prior knowledge. We instrumented tools to accommodate students with different levels of technology. For example, some students' computers weren't powerful enough to install PyCharm, so we created alternatives using web browsers.

By the end of Code in Place, we saw that there were a large number of learners who completed the class (or a large number of assignments in the class) and there were some that for whatever reason—life gets busy, something comes up—weren't able to on the regular timeline. But they still had the opportunity to continue later and use what they learned, even if they didn't get all the way to the end of the class. It's the notion that some amount of knowledge is oftentimes more useful than no knowledge, rather than whether students reach an endpoint since we are learning throughout our entire life. So the challenge was for people who had learned something, but may not have actually finished the class, to feel as though they got something worthwhile out of it.

AL: Code in Place surpassed initial expectations for online courseware with its interactive nature and sense of community. How can platforms like MOOCs (Massive Open Online Courses) learn from the unique structure of Code in Place?

MS: Part of it is the power of human involvement, which takes many forms. One is the feeling of accountability. I think the fact that there were weekly sections and students felt instructors paying attention made it more likely that they would continue. There was also a notion of active community. The fact that Code in Place was not available anytime to anyone—there was actually a cohort of people moving together and helping one another—gave it a timing that people generally tried to keep up with. It's sometimes easy in online learning to believe that the material will always be there and plan to do it later. The problem is, later never comes because life gets busy. But feeling that cadence motivated students and helped to create a whole Code in Place community.

I think the larger lesson for MOOCs in general is that the cohort structure can foster accountability and support. Maybe MOOCs can't achieve the same 10:1 student-teacher ratio at scale, but is there a way to provide support or to leverage previous offerings of the class? Peter Norvig—who is well known in the Artificial Intelligence community and works at Google—had this interesting perspective of viewing online learning not just as a start and an end, but a bus. Sometimes you need to get on and off the bus even though it follows a regular schedule, but it's okay because you can wait for the next time the bus comes around to keep learning.

AL: As demand for remote learning grows, what are obstacles virtual education might encounter? Would you please discuss how CS can help society adapt to these challenging times with learning?

MS: I think the first obstacle is realizing that education isn't just about knowledge transmission. In the early days of MOOCs, there was this attitude that we could simply put content out there and people would learn. But early experiments showed us that education transcends the exchange of information. It's about inspiring and supporting people, providing a personalized experience and accountability so that people will want to keep up and put in the effort. There's all these factors wrapped into the package that we think of as education.

Thinking more broadly in terms of what we can do with technology, we should explore ways that technology might help enhance those aspects at scale. Can we build tools, for example, that make it easier for human beings to provide educational feedback? I've explored this intersection, and Chris Piech has taken a deeper look at it—having humans grade computer programs students submit and pattern-matching places where humans provided feedback to provide that same feedback to other programs with similar issues. The human is still involved, but the work they do can be multiplied greatly to help expand education.

Technology can also be a mediator for traditionally in-person experiences, such as virtual labs where students can experiment with simulations. As more time is forced upon us in this socially distanced world, the more resources will be put into understanding how this works. Developing quality education is hard, and I think it's something that needs to be appreciated. There's sometimes a misunderstanding that if you bring together minds in a classroom and provide them material, everything will work out. But no, pedagogical experiences are carefully crafted to ensure students aren't overwhelmed, but in a place where they're actually learning and able to assimilate knowledge into old scaffoldings. Learning takes a lot of effort and work, and it's something we need to emphasize in addition to knowledge transmission.

AL: You have previously conducted research mapping education and machine learning. Are there any projects that you and your colleagues are developing at this time that the Stanford community can look forward to?

MS: That's a good question. I've been developing more tools to provide analytics for introductory programming classes. They help students see if the time they spent trying to accomplish something was really longer than necessary because there was a conceptual misunderstanding that—had it been clarified earlier—would've helped. This information can also be sent back to the instructor to understand where groups of students struggle and how they might address misconceptions in the future.

More recently, I've focused more of my effort around ethics in technology. So I'm actually in the process of writing a book right now with Rob Reich and Jeremy Weinstein that sits at the crossroads of technology, ethics, and public policy. It evaluates what online privacy, algorithmic decision making and bias, and the power of large online platforming mean for free speech and charts a path forward. That's probably a year away, but we're working on it!