[SQUEAKING]

[RUSTLING]

[CLICKING]

JUSTIN REICH: We are trying to make our way through these four as-yet intractable dilemmas. The last time we met, we talked about the curse of the familiar, the ed tech Matthew effect. Today, we're going to be looking at the trap of routine assessment. Anybody want to try to articulate the core puzzle that's at that's at the heart of the trap of routine assessment? What do you got, Dana?

AUDIENCE: Kind of along the lines of, teachers are teaching students a subject, and schools have such an emphasis on final exams and what they're preparing for these standardized tests, but we're also fighting with the idea of what a computer can grade. So if you're trying to standardize some sort of way of grading something, you can only really focus on teaching students things that computers can do.

But if a computer can already do it, then why is that the emphasis of what we're teaching students? So it's kind of the idea that, if you look at the Venn diagram, there's this whole other half of the subject that ultimately we probably should be focusing on. But we are really only talking about that overlapping aspect of what students we're saying should learn because we can assess it.

JUSTIN REICH: Good-- that the technologies we have for assessment constrain what it is that we can assess in any kind of scale. Anybody want to add-- what else seems important to Dana's initial offering there?

AUDIENCE: I think something you brought up in the reading, as well, is that there's a connection to the labor markets. So schools are supposed to help us prepare for the labor market, but we're learning these routine things that are being taken over by automation. How is that actually doing what the school is supposed to function for?

JUSTIN REICH: OK, good. So schools operate in between a couple of different things. One is, what do we need kids to do changes over time, as we develop new technologies. None of you should teach your students-- well, I mean, you shouldn't teach your students to operate a loom on the basis that will be a source of gainful employment for them, although there probably are a small number of people who are very profitable small business loom operators and particular kinds of things, but generally speaking, not a good bet.

And so it makes sense that as technologies come along, we change the kinds of things that we teach to young people. And computers, in various ways, once a decade, over the last 10 years, cause new kinds of problems for us. As computers' capacities expand, what we think about teaching people changes.

At the same time, as computers, technologies expand, there's maybe new or different kinds of ways that we can use them for assessment. And so those two things are interplaying with each other all the time. We have some seats up here. There's plenty of room could use. Nice to see you. Other introductory things that seem important? Omar?

AUDIENCE: Yeah, that's not to say that-- and you mentioned this in the reading-- we're not trying to say, don't teach times tables and stuff like that, where it's like stuff that's-- yes, computers can assess that, but stuff that students need fundamentally to build on for other, more complex tasks that they will learn down the line.

JUSTIN REICH: Good. That makes the whole conversation even more complicated because there's stuff that we need people to be able to do in their lives. There also might be a class of things that's called "stuff that you need to learn in order to be able to do those complex things, even though computers can do those simpler things." So there's some class of things that computers can do that we just don't need humans to ever do anymore.

I think I joked with you all that one of the first times I talked with my students at MIT about large language models, that somebody said, I'm using large language models to do the LaTeX formatting of my papers. And it's like, great. Never think about that again. If no student ever thinks about that again for the rest of their lives, we're probably fine as a civilization.

But single-digit multiplication is not like that. Single-digit multiplication can be trivially done by computers, but it's actually quite important for human beings to learn. So there's some complexity in those kinds of things. Anything else that people want to add to just the initial framing of the trap of routine assessment?

AUDIENCE: I guess just to add on even more, there is this aspect that our society values these routine assessments so that, even though you're trying to change this and branch out and expand what we teach students, they're still always constrained by the market itself. And ultimately, teachers have to fight against that and their own independent wishes. So as much as we think you can just do it, you can't actually.

JUSTIN REICH: Good. So I mean, I think you could go to almost any educator and say something along the lines of, like, are standardized tests a good mechanism for capturing all of the important things that students should know? And the answer is, clearly not. There's almost no educator in the world who believes that to be the case.

There's more debate over the question of, do standardized tests capture some of the important things that we want students to know? And then a follow-on problem that there's then this class of stuff which is pretty important for students to know, but we're not very good at assessing.

And it's hard to prioritize those things in education systems because it's hard to reward students and reward educators for teaching things that we don't know how to measure or don't know how to measure in some kind of systemic way. We don't know how to measure with the kind of fairness that we would want to use for high-stakes assessments for things like graduation exams or college entrance exams or other kinds of things like that.

So all of these are parts of systems that get wrestled with every day by educators at all different kinds of levels. In the United States, teachers are unusually responsible for wrestling with these issues because we don't have something like a national curriculum. We don't even really have something like national entrance exams or something like that, where the whole system has to work together to respond to some of these things. Teachers in the United States have unusual discretion for thinking about these kinds of challenges.

I thought the place that I would start with you all is a little bit towards what Ty suggested. And we're probably going back 20-ish years to one of the last times computers really got educators and labor economists quite upset. So we're in another one of these moments right now with LLMs where people are pretty stressed out. How is this going to change the labor market? How is this going to change what we expect of our students? How should this change school systems and things like that?

And to me, it's always helpful to remember, oh, we've done this before. The introduction of personal computers into most households in the United States, that was a big deal. It was a big change in the-- the introduction of personal computers into workplaces, that was a big deal. The introduction of the internet, broadband internet connection in these spaces, that was a big deal that evoked all of these questions in the past.

And one of the best summary pieces to read from probably-- what was the 2015 summary from this debate is a paper called "Dancing With Robots" by two economists, one who is here, Frank Levy, and another who is at Harvard, who was actually my advisor, Richard Murnane. This was based on another book they wrote called *The New Division of Labor.*

And the original research for *The New Division of Labor* is kind of right when personal computers are really starting to widely spread in workplaces, in people's homes, and things like that. An early bet that economists made was something like-- one of the things economists think about all the time is, what do jobs require? And education economists then start thinking about, OK, what do jobs require, and how do you teach people the things that the jobs that we actually have require?-- that computers are going to complement high-wage workers, and they're going to replace low-wage workers.

That was a pretty early bet that a bunch of economists had had. And there was something about that to Frank Levy and Richard Murnane that didn't smell right. They were like, I think it's going to end up being more complicated than that.

So they thought a lot about the specific kinds of things that people did in jobs and how those jobs were changing. How many of you have ever checked in to get on a flight by talking directly to a human being? OK, so it looks like people-- how many of you, the majority of times you get on a flight, you check in with a machine? OK, so this is actually surprising. I'm surprised that almost everyone in the room who raised their hand said that at least once.

When I first put together this slide, like 15 years ago or something like that, this was a pretty novel experience. So when I was in college, you would not interact with a machine at all to get onto a flight. You would have called the airline or called the travel agent to buy a ticket. You would have a physical ticket. You would take that physical ticket. You would show it to a human being.

Now, the interesting thing about those conversations with human beings was that they were highly standardized. They were highly routine. So post 9/11 it would be, who are you? This is my ID. Where are you going? This is my ticket. Do you have a ticket? Yeah, this is my ticket.

Where are you going? This is where I'm going. Do you have any bags with you? Yes. Did you pack your bags yourself? Have they been in your possession the whole time? Yes. And then off you go.

So the vast majority of times that a person went to a ticketing agent to have a conversation with them, highly structured, highly routined. We know what kinds of questions we're going to get as passengers. We know what kind of inputs we need. We have the conversation successfully. We move on.

And so as computing technology becomes more sophisticated, as user interface becomes more sophisticated, as all these things become less expensive, someone's like, hey, we can just program those conversations into a machine. And so now, overwhelmingly, you have that exact same conversation. You just push the buttons to say, yes, I have a ticket. Yes, this is me. Yes, my bags have been in my possession the whole time.

However, for as long as we've had kiosks, there has always been a number of cases in which the kiosks can't respond to the demands of the moment. So there's something really weird about your ticket. There's some problem that's gone on with your booking that the machine hasn't anticipated or can't understand, or you have some kind of communication issue.

For some reason, a human being can't communicate with the kiosk. Maybe their glasses are broken that day. Maybe they are trying to travel-- they don't speak the language that the kiosk is in. They're so angry at the airline for what they've messed up, they just go banging on the kiosk.

So you've got to think a lot-- there's one person behind this row of kiosks whose job it is to have all the conversations that the kiosk can't have. If you're a labor economist, that's a pretty interesting question. What is that human being doing that we couldn't program into all of those machines?

That was the kind of question that Frank Levy and Richard Murnane were kind of asking. In a world where there's lots of work that we can put on computers, what is it that humans are still good at? In economists terms, what is it that humans still have a comparative advantage over computers around?

Here is another funny example, which I think I wrote about in the book, about how some of these things change. So in something like 1987, my parents bought a cabin in Vermont which had an old boiler in it, like a hot water heater that was attached to a tank. And my mom passed away. I inherited the house.

So sometime around 2007, about 20 years later, this boiler dies, and I'm responsible for it. The original boiler was installed by this guy named Dan Masterson, who lived on Masterson Lane in Bethel, Vermont. And I was trying to find a plumber to replace this hot water heater. And the guy I found was Dan Masterson, who lived on Masterson Lane in Bethel, Vermont.

So the guy had been a plumber for 20 years, and we decided that instead of doing an old-school hot water boiler, we're going to install one of these on-demand systems. So this is a very beautiful picture to me because there's no hot water tank that can freeze. It just like heats up the water exactly when you need it. You can see how nicely all these pipes are. The last building had pipes all over the place, things like that.

So Dan installs this system. Dan, as you can imagine now, is getting towards the end of his career as a plumber. And I say, great. Dan, show me how to use this system. He's like, well, you walk in here, and you open this little door. And when you look at the system, it should say, "Good."

[LAUGHTER]

I was like, all right, Dan, what do I do if it doesn't say "Good." You guys can't see this well. He's like, see this little button that says R? Push that. So that's the Reset button. I was like, all right, I can push the Reset button. Well, what do I do if that doesn't work? He's like, you see this electrical outlet up here? Just unplug it, and plug it back in.

I was like, all right, I will recycle the power in my hot water heater if that doesn't work. And I was like, OK, so it doesn't say, "Good," I've hit R, I've unplugged it, it's still not working. Then what do I do? He says, there's one guy in the upper River Valley of Vermont who has flown to Scandinavia to get the training on how to reprogram these systems. And so if those two things don't work, go find that guy.

And I tried to imagine-- Dan Masterson, probably in 1970, 1975 or something like that, going through his plumbing apprenticeship training. And he learned how to solder pipes together and how to do some basic electrical stuff and things like that. If he went to the same training now, you have to be an apprentice computer programmer to be a plumber.

And not only that, the systems are designed to solve all of the easy problems. If a problem is easy, it'll just solve it itself on reboot. The things that you have to be able to solve as a computer programming plumber are the ill-structured problems, like when the system can't figure out what's wrong with itself and can't correct itself and those kinds of things.

So, I mean, the reason I'm trying to bring these two cases to your attention is we could say, all right, there are some classes of jobs that are going to disappear and be replaced by computers. There used to be-- I mean, the other thing about these people, if you're just thinking about this in a broader social con-- there used to be six people behind this desk.

And they wore uniforms. Maybe they had a college degree. Maybe they had an associate's degree. They probably made a middle-class living. You could make a decent living as a person who checked people in to airlines. And most of those jobs are gone. And even in places where jobs remained, they're being transformed by the capacities computer offer in these systems.

So one of the cool things about the research in *The New Division of Labor* is that they basically go-- the Department of Labor collects all of these names of jobs and the description of jobs. And they do an analysis of those descriptions to see how they're changing over time. And what they find is that you can break jobs down into maybe four categories.

One of those categories would be routine manual jobs-- so that involves jobs that are lifting, moving things around, stuff like that-- routine cognitive work-- so that would be things like filing and stuff like that. That would be things like having standardized conversations to check people into an airline.

Two others are what they call ill-structured problem solving-- so solving any kind of problem where you don't know what inputs you need, you don't know what the outputs look like, you don't know what the correct answer is. You're solving that as you're figuring out the problem-- or complex communication, tasks that require figuring out-- any kind of solving the problem requires talking to people. Some of that could be marketing and sales kind of thing, figuring out what people want. But other than that, it could be a home health aide does a ton of complex communication.

And their basic thesis was that things that are routine are disappearing from the American labor market. This is a huge problem if you want to have a stable society. In the 20th century, there were many, many people who performed routine labor for a reasonable, middle-class wage and had a perfectly suitable life doing it.

Some of the kinds of jobs that are growing are service jobs that don't really require-- they're a little bit of routine manual work, a little bit of routine cognitive work, and they pay the absolute minimum wage possible. That's a little bit less concerning today where you can get jobs doing service work that pay-- you can work at some place in McDonald's and make $15 an hour or $17 an hour, things like that.

But when there was a surplus in the labor market 10 years ago, 12 years ago, all those people would have been paid $7.25 or whatever the federal minimum wage was at the time. But the two kinds of jobs that were growing were things that required non-routine analytical skills and social skills. Some of this was updated by this guy David Autor who's an economist who's here now, as well.

So if you're an educator, the thing you got to be thinking is, like, man, we have a different job in front of us. It's not sufficient to prepare people to do jobs that require routine cognitive work or routine manual work because those jobs are disappearing as a part of the labor market. And even when the same job title exists, like they're still plumbers, but plumbers are increasingly called upon to do ill-structured problem solving and to do complex communication.

This was an update of this, I guess, from 2012. This is just another-- here's a 2-by-2, where jobs that require social skills are here. Jobs that require math skills are here. The jobs that are in green are the ones that are growing. The jobs that are in red are the ones that are disappearing.

And you can see that they're clustered in the things that require social skills are growing, and the things that require fewer social skills are disappearing. Truck drivers is one that people are often, particularly at a time when people were pretty-- this was 2015, 2017. There are 3.5 million truck drivers in the United States. That's about one in every hundred Americans is a truck driver.

And probably 10 years ago was some of the headiest time where computer scientists believed that it was reasonably possible to replace all of those people, that you would be able to develop artificial intelligence systems that would be able to drive trucks. So Uber went to Carnegie Mellon and bought their whole robotics department. They didn't actually pay for the robotics department. You just pay each individual faculty member to go leave and join your company.

And there's still 3.7 million truck drivers in the United States or however many there are because this turned out to be a much harder problem than we thought. But this is the kind of angst that exists in society when these new systems are developed. What would it be like if-- what if every kid in school today who's destined to become a truck driver, we have to find something else for them to do? That's a pretty challenging thing to think about.

This was another 2017 example. And I don't know. I was glad to go back and revisit this because there's lots of things that are going on today about, like, man, LLMs are going to replace all this work that people are doing. But for the last two decades, we've been building software that replaces the work that people are doing. There's all kinds of machine learning based-systems that can do what lawyers do and things like that. You don't necessarily need LLMs to do it, although maybe it will accelerate some of these kinds of things.

So when I was giving this talk to educators seven or eight years ago or something like that, this would be the point where I would say, imagine you could go into your school today, imagine going into any of the Cambridge Public Schools, Somerville Public Schools, whatever it is, and grab every homework assignment that had been given out that day. So go to all the classrooms, collect every single assignment that was given out, and put them in a big pile and start going through them.

Start classifying those homework assignments, like, how many of them require some kind of ill-structured problem solving or expert thinking? How many of them require some kind of complex communication? And how many of those homework assignments ask people to do the kinds of routine work that we no longer need human beings to do? And most educators confronted with that question would be like, I think we're doing a whole lot of the routine stuff. And conceivably, that's something that we should be revisiting as the world is changing.

How many of you encountered the idea anywhere in your study of education of 21st century skills? Is that a thing that people still discuss? Courtney, when you were ever working on products, did people talk about 21st century skills in your--

AUDIENCE: No, I was mostly in ELA and medical ed.

JUSTIN REICH: Yeah, yeah. We might have had some of this stuff. Where did you encounter--

AUDIENCE: So in the nonprofit space in India, for education, there's a lot of nonprofits that are trying to do 21st century skills. It was a buzzword, I think, a while ago--

JUSTIN REICH: Yeah. Once you're 25 years into a century, the idea of 21st century skills becomes-- it felt more urgent in 2004, 2005. Where were you going to say, Kirk?

AUDIENCE: I was just-- examining in general, I think, came out of that.

JUSTIN REICH: Yeah. There was this notion that, OK, now that we have computers saturating our workforce, there's some new set of skills that we're going to have to teach young people how to do. And it was kind of interesting because if you-- I don't know. There were probably like a dozen different organizations that came up with lists of 21st century skills.

For a while, there was this thing called the Five Cs-- communication, collaboration, creativity, critical thinking. There's a fifth one. Maybe it was the Four Cs. I don't know, Four Cs, Five Cs, something-- hmm?

AUDIENCE: Which one?

JUSTIN REICH: The Four Cs, something like that. It was interesting because actually, almost all of those lists at some point cite Murnane and Levy's work. The only people that did original, empirical analysis were these two economists. And then everyone else kind of freestyled on top of that. But I think there was something significant there that you could kind of tell by-- I mean, I think *The New Division of Labor* is 2005 or something like that.

But 20 years ago, you could have a sense that, man, we're going to need many, many more people with far more cognitive skills than we've ever needed before to fill our labor market. We're just going to need more people who can do expert thinking, ill-structured problem-solving, complex communication than we've ever needed before in the history of the American education system. And that's a stressful thing to contemplate. So that's one set of fundamental ideas, a way of thinking about, what do we want people to know?

A kind of analysis that people will be doing in the labor market-- education economists, labor market economists-- in the next five years is something like, well, OK, how do LLMs change this equation? How does generative AI change this equation? Does it simply accelerate trends that have already been existing? Is there some new set of answers to these questions? Are there different classes of skills?

David Autor, who's one of the people I just mentioned, just wrote a-- I don't know what he wrote. I think he wrote a working paper that was then quoted in *The New York Times* or something like that. He's been a person who, for decades, has been saying, oh, computers are pretty bad for the middle class.

A figure like this-- this is a bad figure for the middle class. This is where the middle class existed. These people are disappearing. These are either rich folks getting richer or poor folks-- the service line is poor folks getting poorer.

But he just recently came out with something which is like, actually, LLMs are going to be good for the middle class. Generative AI is going to be good for the middle class because there's going to be a whole class of jobs that people can do with AI support that they wouldn't have been able to do without graduate training or something like that. And now they're going to be able to do it with these systems that are helping them.

And then there's another economist here, Daron Acemoglu, who was like, I don't know, David. In*The New York Times'* article, he's like, oh, that's pretty optimistic. I'm not sure that's going to work out. But these are the kinds of things that people are wrestling with now.

The heart of the trap of routine assessment is that the kinds of things that computers are good at assessing almost exactly overlap with the kinds of things that computers are good at doing. And those are the kinds of things that we don't need people to do anymore. The kinds of things that we can build, tools that are good at assessing human performance are almost always very closely overlapping with the kinds of things that computers are good at and the kinds of things you don't need them to do anymore.

So here are a couple of examples. Here's a framework for mathematical modeling. So one of the big changes in the Common Core State Standards that came about 15 years ago or something like that was that we wanted to teach students not just to do procedural kinds of things in mathematics, but to be able to conduct exercises in mathematical modeling.

And those exercises of mathematical modeling, you could think of them having five steps. First, you have to find some problem in the world. What is an interesting thing to view? Then you have to set that up as a model, whether that's an equation, or a schematic, or a, table or whatever it is. You have to take a problem and then you have to frame it in some kind of way.

Then, probably once you've developed that model, you have to compute some kind of solution inside that model. Then you have to explain your solution in terms of the real world. So was it 27 board feet, or was it 27 kittens, or was it 27 acres?

And then you have to be able to explain why your solution makes sense. Why did you follow a series of reasonable mathematical steps to come up with that solution? Of those roughly five steps, what do we have technologies to evaluate? Of those five things, what have you ever been tested on?

AUDIENCE: I want to say 3.

JUSTIN REICH: Yeah, computing the solution. The only thing of those five steps that we can reliably get computers to do is evaluate whether or not you can compute a solution. If you use mathematics in your profession, which of those five steps will you almost never actually do? Computing the solution.

A computer can't tell you which problems are interesting. A computer can't tell you like, oh, this would be a good problem to think about through this kind of equation, or through this kind of formula, or through this kind of framing. A computer is not that good at taking the outputs and being like, OK, this is what it means for the real world.

And computers are really not particularly good at explaining to you why what just happened mathematically makes sense. But the one thing that you will assuredly do if you're a professional mathematician is have the computer do the math. The computers are significantly more reliable than you are at those kinds of steps.

So if you go and sit down for any mathematics exam that's given in a standardized way at any level, you will spend the vast majority of your time computing. If you're a mathematician, you'll spend almost-- not really a mathematician, but if you are a person who uses mathematics in their work, you will spend almost none of your time computing. You will spend almost all of your time doing those other kinds of things.

That's a real problem. That's a real problem. A useful concept that people who study-- psychometricians, people who study testing, they have a term for how we misunderstand math tests, or tests in general, and it's called the reification fallacy. Reification means the naming of something.

If you call something a math test, people will think it's a math test. But calling something a math test doesn't make it a math test. So for instance, every time you take a math test, you need to know something about language. You probably need to know something about test-taking strategies. There's probably a whole bunch of other cultural things that you need to know to function in a test. And the test is effectively evaluating all those things.

The other thing is a test can-- there's no test that can say that all of eighth grade algebra. That would take you forever. Instead, what we do is we sample out of a domain. We say, here's all the stuff that might fit in eighth grade algebra. And we're going to pick some of this stuff, and some of this stuff, and some of this stuff. A way you could think about this problem is that we oversample computing, and we undersample all of these other kinds of things.

Well, maybe let's do one more that's like this. Here's one of my very favorite paragraphs that's ever been written about computer programming by Hal Abelson, who's here, and some of his colleagues. This is from a computer science textbook. "First we want to establish the idea that computer language is not just a way of getting a computer to perform operations, but rather that it is a novel, formal medium for expressing ideas about methodology." This is probably the key sentence. "Thus, programs must be written for people to read, and only incidentally for machines to execute.

Second, we believe that the essential material to be addressed by a subject of this level is not the syntax of a particular programming language constructs, nor clever algorithms for computing particular functions, nor even the mathematical analysis of algorithms and foundations of computing, but rather the techniques used to control the intellectual complexity of large software systems." Why would it be important for programs to be written for people to read and only incidentally for machines to run?

AUDIENCE: People are the ones who are maintaining the code. If they can't understand it, then how are they going to maintain it?

JUSTIN REICH: People have to be able to maintain it. What else might folks say? Yeah, the way you add new features, the way you collaborate, the way you build large, complex systems is that you build a system that people can read and understand what it's doing. What you can't read and can't understand what it's doing. Now, when you all take introductory computing classes, and you submit your homework assignments, and they're automatically graded, what are the characteristics of your programming that those systems are good at grading?

AUDIENCE: The output of whatever the code is.

JUSTIN REICH: OK, so the assignment presupposes some kind of engineering test and figures out whether or not you pass that test. Did your spell check dictionary detect the misspelled words or not?

AUDIENCE: If it passed a series of randomized test cases from input to output.

JUSTIN REICH: OK, good. So similar kind of thing. We throw a we throw a bunch of possible inputs in this system. Does it all come up with the outputs that it's supposed to come up with? Other kinds of engineering tests. Dana?

AUDIENCE: I was just going to say, sometimes isn't that a time? Does your code run fast enough?

JUSTIN REICH: Good. So there could be a-- that's one kind of efficiency metric. I think there are other kind of efficiency metrics that maybe-- have you ever done things where you only have a certain number of lines or a certain number of characters or something like that to try to-- hmm?

AUDIENCE: That's some--

JUSTIN REICH: Generate some parsimony or something like that.

AUDIENCE: Some classes here do have these style checkers. I'm not too sure what they check, exactly. But they check for certain like stylistic choices you made in your coding. But a lot of them also depend on you to have correct computational code to then give you the style point.

JUSTIN REICH: Yeah, good. And I think the style starts getting at, the programs must be written for people to read. Now, they probably don't check things like, Did you name your variables and your functions something sensible that somebody else would be able to understand later on? or other kinds of things like that. But you could think of those style guides as initial-- we have pretty good tests for the "incidentally, for machines to execute." That's kind of our efficacy, our efficiency evaluations.

In some ways, the style checkers you're describing, also, they might be kind of silly. But another way to think of them is as a reasonable first step towards these kinds of things. What are all the machine-readable parts of style that we can get a computer to evaluate? And how, over time, would we build increasingly sophisticated style checkers, computer assessors that would just do a better and better job year after year of evaluating more and more features of code that have to do with humans understanding and communicating with each other rather than with just machines? What were you going to say?

AUDIENCE: I was just going to add that basically, I've been a part of a course where they also do design checks for the code on top of style metrics. So they're checking, are you writing good, efficient loops? Are you writing redundant code? But even so, I really feel that, within this introductory programming space, they're mostly checking for how well the students are overfitting to the given problem space already. What they want to test for is problem-solving, but that's just really hard to do, I think.

JUSTIN REICH: Good. So that it's very difficult-- so A, in the solution of any given problem, it's very difficult to detect for something that would be generalizable problem-solving skills. And even worse, it's essentially-- so far, it's proven more or less impossible to design assessment machines that can detect something like generalized problem solving skills.

To some extent, anytime we build these things, our most sophisticated assessment tools, work best on particular problems that you've fit them to work on, and other kinds of things like that. And they work less well on more generalizable kinds of things. What are some other domains that we could think of that are-- maybe I'll have you chat with each other for a minute.

Talk about two things. What are just other domains that you can think of that are like this, where computers can test something in the domain, but they may not be able to capture the most important or big, important classes of things in that domain? Think about that for a few minutes. That might be putting a pessimist's hat on to think about that.

And then put on your optimist hat for a minute or two and say, in the next five years, where do you think we're most likely to see the most progress in thwarting the trap of routine assessment? Where are we most likely to develop assessment tools that actually evaluate people on things that we really care about and want human beings to be able to do? Turn to the person next to you or a couple of people and talk about those two things for three or four minutes. Ready? Go.

Let's do the first one first. What are some domains that you could think of where it seemed like, ah, we really are having a hard time using computers to get at the important stuff here?

AUDIENCE: Essay writing or short stories or anything to do with writing and literature.

JUSTIN REICH: Great. And that's probably one of the most studied, most examined domains. And again, it's a place where we have-- at least the baseline in that field is pretty well understood. So some of the very worst assessment-- just the most disappointing kind of assessments we do is people's long-form essay writing on standardized tests.

There was a guy here named Les Perelman who used to joke about the quality of standardized writing assessment by noting that he could pretty reliably score essays from across the room because you just hold them up and see how long they are. And just by glancing them, he could be like, oh, that's 2, that's a 3, that's a 5.

He also wrote pretty early on a pretty fun piece of software. There are all these companies that were creating automated essay graders, and so he got a group of MIT students to make software that generated nonsense text that did really well on these automated essay scores, which was just-- it was pretty fun to read these things, which just were like complete hallucinations of madness. And then some system would be like, 5.

But we actually have developed software tools, so when GRE essays are scored or things like that. So humans go and score them. It's incredibly disappointing how they do that. They read them all very quickly. They score them holistically. The people who do the scoring are not particularly well-paid, not particularly well-trained, as you might imagine.

It's enormously boring to read essays over and over again about, Who is the person in your life that inspired you the most? or whatever other ridiculous questions they ask. But anyway, you can train those people to be reasonably similar in their scoring to one another.

And not only that, but for a given question-- like, if we ask people over and over again, Who is the person that inspired you the most? you have a bunch of humans evaluate it, you feed all of those essays into some kind of machine learning algorithm, we can build machines that are roughly as reliable as those humans. So we can create automated essay scoring tools that do a reasonably good job-- that predict what a human would score an essay about as reliably as two humans would predict what the other person would score the essay, about as reliable as two people are.

Those systems don't work for shorter-answer things. So you have to be in the realm of a few hundred words before these systems become reliable. They're also not very useful for novel questions. If you invent a new essay question you want to give someone, you can't take one of these algorithms off the shelf and run it against the system and have it work, which is why-- I don't know.

I don't know how they do it exactly with essays, but these standardized test companies, some fraction of the tests that you take every time you take it is not scored, and it's just used for development for these kinds of things like that. So you're somehow-- we've convinced you to pay $75 to do voluntary work for giant test-making companies and things like that.

But yeah, that is a domain in which-- people will also disagree. Some people will be like, man, it's great that we have these systems because if you have tools that can automatically score essays, then you can encourage people to write more essays. If we didn't have these machines that could score essays, they wouldn't be on standardized tests. And if wouldn't be on standardized tests, then teachers wouldn't assign them. And then we would teach less writing.

And so as crappy as they-- maybe there are some critiques about their concern-- maybe there are some concerns about their quality, but they're still good for the system. But yeah, clearly not all everything that is important about writing is captured in those things.

AUDIENCE: I feel closely related is language learning. You can very easily assess whether someone knows a lot of vocab, or verb conjugations, or grammar, or whatever by via computer. But that's not really the point of language learning. Language learning is-- a lot of it's about speaking, and listening, and writing, and stuff.

And especially the speaking and listening part, it's probably going to take a long time for computers to do that well, especially when you have variability due to accents and regional stuff. It would be hard to assess whether or not someone really is fluent in a language by simply interacting with a computer.

JUSTIN REICH: Good. And language learning is such an interesting subject in general because it shifts so much in the middle of it. There aren't that many things-- the kinds of things you do when you teach social studies to third graders is not that different from the kinds of things that you do when you teach social studies to 12th graders. It gets more complicated, but it's along the same lines.

But the initial learning of a language is more like a math or a science, and then the later parts of a language is more like a humanities or a social science or something like that, which is cool. In the book, I write a bunch about automated pronunciation detectors, which, to me, I think are a great example of advances in language learning, especially when you think about the classroom contexts in which people learn to pronounce things.

So take yourself back to your middle school Spanish class or whatever, and there's 26 kids in the room, and there's one teacher. And the teacher walks up to you and goes, "Hola, cómo estás?" And then you respond, and maybe the teacher gives you some feedback about the pronunciation. Maybe they're just whipping around the room. But you're getting at best like 1/26 of that teacher's attention for that pronunciation evaluation exercise. It's extraordinarily inefficient.

I remember this a lot because I'm both bad at language learning and bad at pronouncing things. And so I would say stuff, and the teacher would just nod at me like, OK, and go on to the next person. Clearly, I'm not learning this. But I can open up a computer that has an automated pronunciation detection, and I can just mispronounce "por favor" over, and over, and over, and over again, and this thing doesn't care.

It'll listen to me say it as many times as it wants. As they get better, they might be able to-- they can at least give me examples of correct pronunciation. They might be able to fine-tune my pronunciation. That seems like a place where computational assessment might really be a whole lot better than human assessment in a domain, that's pretty important. What was your example, Sabrina?

AUDIENCE: I was going to say, using computers to analyze art. It can tell you about the color compositions, the time period that it's from, what makes it objectively appealing to look at. But it doesn't necessarily go into the artist's background or the emotional message behind it or why it was created in the way that it was. I think, with art, it's usually not the surface level, which I think that the computer is really good at hitting. But maybe the overall meaning message is missed entirely.

JUSTIN REICH: Yeah. I mean, there are AP studio art classes and things like that, where you can submit a portfolio. And I've never thought about it before you brought it up. But to my knowledge, there's no application of software in evaluating a student's art portfolio. But it's a kind of fun exercise to start thinking about, what could you assign an AP studio art student that could be graded by a computer that people would generally think is fair?

Now, clearly, abstract painting or something like that, probably no one will think that's fair. Certain kinds of still life, where the task is, reproduce this bowl of fruit as closely as possible to what the bowl of fruit actually looked like, I would imagine I would trust the computer to do that as well or better than a human would. But I'm not an artist. Maybe there's something about still lifes I don't understand.

AUDIENCE: The only application I can think of-- and I am very biased because I do a lot of photography-- is detecting-- and even then, it's difficult-- detecting if something's been altered in any sort of way after the fact, like Photoshop. And I know Adobe is working on their own thing to detect--

JUSTIN REICH: What would be a thing that you would want a novice photographer to be able to do pretty well that a computer could tell whether or not they could do it well?

AUDIENCE: Whether or not a photo is-- I think the most fundamental thing is whether or not a photo is exposed properly without having to do any digital alterations.

JUSTIN REICH: Yeah, yeah. And again, if you can get a computer that can do that evaluation, then you can have all of the photography students get more formative feedback. Maybe you can get them to have more high-stakes feedback. Maybe we could admit students into arts colleges based on their photography portfolio, or we can just ask them more questions in their portfolio.

We can say, humans, go see whether their intimate portraits evoke the character of the person, and don't waste your time figuring out whether or not they can expose photos correctly because we got machines that can do that. Yeah, that's a neat example. So where were you most optimistic? Where do you feel like there's going to be the most improvement in the next five years, and where are those improvements going to come from?

AUDIENCE: I have a follow-up question.

JUSTIN REICH: Oh, sure.

AUDIENCE: Is it OK?

JUSTIN REICH: Yeah.

AUDIENCE: Sorry. I was just thinking that if the photos' exposure level is detectable by a computer, so you begin to not have humans evaluate that on admission tests anymore, then soon you'd have to drop that from the test, like that portion of eval-- because students would then have access to these kinds of softwares, as well, which means they would never submit a photo that was not exposed properly.

JUSTIN REICH: That is a great description of the trap of routine assessment, that, if computers can detect performance, then computers can probably do that performance. If we can build a machine which tells whether or not a photograph that was taken with the correct exposure, then we can just build cameras that will only take photographs with the correct exposure, and then we don't need to teach humans how to do correct exposures anymore.

And so that is part of this loop that educators are constantly wrestling with, constantly thinking about. And a thing that makes this so hard for educators is it's really hard to predict-- when your old high school friends are going to call you.

[LAUGHTER]

It's really hard-- I won't say any more. It's really hard to predict in what ways these kinds of disciplines will change. Like, how will photography change over the next five years? is a really hard thing for photography educators to guess right now. So I've been thinking about this a lot because I shared this paper that we published with you that was, "Generative AI in K-12-- an MIT Perspective." I didn't share it with you all. I shared it with my students.

And one of the claims that we make in there is, we can be pretty sure that computer programming is going to change, and that copilots are going to be a really central part of computer programming. And so computer science teachers should probably start thinking about how they're going to change their instruction. But I was just at a conference the other day where someone from Cornell Tech stood up and was like-- and the whole premise of the conference was, what guidance should we give to computer science teachers to change their instruction?

And this woman stood up and was like, do not do that. You have no idea how computer programming is going to change. There's no reason to give any new advice to teachers right now because we have no idea what that would be.

And I started thinking more and more like, well, yeah, in part because I've been hearing from people who teach introduction to computer programming that their early efforts to incorporate copilots in instruction were going pretty poorly, that basically, students were using them, and then week 4, week 5 would roll around, and they're like, these students are totally confused. They haven't learned fundamental things they need to learn. And then I talked to other software engineers who were saying things like, every hour you save programming with a copilot will cost you an hour in quality assurance later, that the things-- you solve little problems, but you end up creating bigger ones, for some of the reasons that Hal has on this screen right here.

And so then you start saying, man, maybe we don't know. It's pretty hard to think about how you should change-- some people might say, why are we assessing computer programming at all anymore? The bots are just going to do it for us. And then other people will be like, yeah, I'm not so sure about that. We might actually need people doing this or doing parts of this.

The other thing I think about a lot is that, historically, educators are not very good at this. So if you read, in the 19th century, the 20th century-- in the 19th century, there were a whole bunch of people who were fervent believers in diagramming sentences. Did anybody have to diagram sentences when you were in elementary school? Subject, predicate, adverb, all these kind-- I mean, there are people who went to their graves believing that if we didn't teach young people what the pluperfect was, society was going to collapse.

And they were totally wrong, and we laugh at them today. But there's all kinds of stuff that you and I would sit down with a bunch of curriculum developers and be like, man, it's incredibly important that kids in 2024 learn X. And some fraction of those things are absolutely going to be vital in 2075.

And some of those things that we would passionately defend right now, people in 2075 would be like, can you believe those folks 50 years ago thought that people needed to learn X? Those are like the knuckleheads who believed in the pluperfect in the 19th century or something like that. They are very difficult things to predict. All right, let's talk about more optimistic things. What are other things, Dana, that--

AUDIENCE: We had a really interesting conversation. We were saying, if you go back to what you were talking about earlier in how AI can help teachers inspire-- or AI can inspire teachers with new homework assignments. But then we brought up this weird, like EdTech Matthew effect, where AI only pulls-- specifically, say, ChatGPT is only pulling from existing literature on the subject.

Other than the initial time that it maybe accidentally puts in a random word and starts going off the walls and makes something crazy, it's really only pulling from what is already out there. So in the idea of bringing up the middle class and the teachers with new inspiration, you almost have to pull ahead the elites to then produce the literature that is necessary to then inspire AI with new things that will then trickle down to the middle class for them to use.

So it was almost this interesting idea that AI can be useful, but it needs to have the right literature to back it up. And there's that interesting dilemma of, who is going to create that? And how do you inspire people to push them ahead in that sense to help the little guy?

JUSTIN REICH: Yeah, what kind of training data gets used is enormously important. We were talking about this with one of our guests from WPI who was saying that-- maybe whoever was doing that presentation can explain it better than I can-- but that as LLMs generate more code, and that code has problems with it, then future LLMs will pull in all that buggy code and suggest more bad code.

AUDIENCE: Yeah, that was me.

JUSTIN REICH: Do you want do you want to say more about what the problem is there?

AUDIENCE: So I used it to-- anyway, so it was that LLMs are trained on Copilot on GitHub code. So they essentially get all of the badly written code that Copilot generates. So when students are generating that code, they are essentially going to generate the same bad code, and they're not going to fulfill the needs of what is required from the question.

When you give someone an assignment, the ways to test it is to give them some test cases. But those test cases essentially do not solve the entire problem. So teaching students how to build those test cases that covers the entire input space is more important than for them to be able to write code.

JUSTIN REICH: Good. Yeah. And so that was-- I mean, who knows if what you just said is true, but that's a great claim about the future. Building certain kind of test cases is going to be more important to the future of computer science than being able to write certain lines of codes or knowing other programming principles or things like that.

This is exactly what educators are wrestling with every time there's some new technological change. Do we still teach kids spelling in an age when we have spell check? Do we still teach kids computation in an age where there are computers? These kinds of things come up over and over again. What were you going to say, Avril?

AUDIENCE: Something that maybe could be an optimistic outcome is that software is a language-oriented discipline. And that's kind of why ChatGPT and language-based models are so good at doing it. But there are things that aren't. Analog circuit design is very visual. And right now, signal processing, like training, machine learning models with images and things, is a lot more difficult.

So for at least a few decades, hopefully, analog circuit design and other things that are related to hardware will be good skills, probably, to teach kids and to build decent fields to work in. Also, even though there are copilot things for things like cadence, such as layout for semiconductor chips or whatever, I don't think they're good. I also don't know how they work.

So there's a more strong emphasis on teaching kids the basics of how the things that they are, I guess, catering to with ChatGPT actually work. What I'm trying to say is that I hope that people have this shift towards understanding how things work from the bottom instead of the top, which is good because then you can build things from the ground up.

JUSTIN REICH: Well, great. Yeah, I like that. What were you going to say, Aaron?

AUDIENCE: I don't know. I have a disagreement with the analog circuit design, but that's just purely because anything-- because it's still a visual software that's doing it, so there is some translation that could be considered. I do know a couple of analog circuit design softwares that just translate down to XML. And you could just stick that XML, which is a structured, object-oriented language, into ChatGPT, get new XML back, validate that it's correct, see it, and then it will probably produce a correct circuit in some capacity.

AUDIENCE: But can you-- I mean, where are you starting when you think about designing an amplifier or something?

AUDIENCE: It's starting from-- well, it's the same thing when you started now. Nobody had code when they first started. You would have whatever data set of people has all their amplifier files that are all lined up, whatever it is.

Then you would decide, OK, let's get the textual representation of this in some syntactical language, which XML was just the one that came to my head, but it could be anything that can represent the image in text form, since that's all the software is doing, since it's just literally inputs and outputs of where you connect wire A to wire B or wherever it is, and that can usually be limited.

And then you take that text, train the LLM on it, you run it through it, you get some out. You then say, OK, generate me an amplifier, and it will generate you an amplifier with that XML. You put it back into the visual software, and now look, you now have an amplifier.

AUDIENCE: OK. My new hope is that all of this becomes proprietary, and they don't let--

[LAUGHTER]

JUSTIN REICH: And I think, to me, from an educator's point of view, what's so challenging about these kinds of questions is I know nothing about your domain, and so both of the cases that you just made seem totally plausible to me, and that it's really hard-- people seem to be really bad at predicting the kinds of things that AI are going to be useful for in the future. Which of these problems are solvable and which are they not?

So you could imagine somewhere out there, there are people who run the ABET engineering curriculum certification kinds of things, or people who are department heads, or people who are writing textbooks or things like that. Which of this stuff do we want to keep? Because we'll still need human beings to build-- for the three categories, which are the stuff is like, nah, in the future, just AI will build this.

You'll tell some machine learning system, AI system the kind of circuit that you want, and it'll print it out. And so you don't really know that you need to know that much about it. B, yeah, an AI will probably print out most of these things, but it's still useful for humans to be able to understand this thing because that gives you some kind of insight about how you would build other technology, other things like that. Or, no, there's no way computers are going to be good at this stuff. We still need human beings of primarily to do it.

Those are at least three categories of things, and it's really hard to predict which things move into which categories. And it's very difficult to get education systems to be good at doing new kinds of things, which is why most of the schools that you went to had very little computer science education and stuff like that because, in the 19th century when we designed high schools, there wasn't computer science, and so we didn't put it in the curriculum. And we're a lot better at doing the kinds of things we've done before than doing new kinds of things. All right, good. Were there other observations or thoughts that people had?

AUDIENCE: I think technology can bring about a change in the legal system. So a lot of people-- lawyers spend a lot of time reading stuff and building upon that for a specific case. And that's what determines their career. How big you get depends on how much you've read, how much you have experienced. So if all this knowledge is readily available to them for a specific test case, then probably it's going to make their lives much more easier.

JUSTIN REICH: Yeah. I think some of framework that Levy and Murnane offer also help us think about, what kinds of things will lawyers need to be good at in the future? To me, it strikes me as very unlikely that computers will have much impact on the part of trial lawyer work, which is convincing juries of things.

The people who actually get in front of juries and try cases and things like that, it's very unlikely that we'll have AI machines that generate scripts for them or things like that because that kind of persuasive, complex communication seems to be the domain in which human beings are most specialized at. We probably-- expert thinking seems to be the domain in which-- are people going to write AI software that detects tax loopholes? Absolutely.

Somebody is going to come up with all kinds of new ways that are much faster than humans at finding ways to defeat the tax code. But there probably still will be human beings who are much better at machines, at recognizing, oh, well, this part might be slightly strictly illegal, but because of the way-- or its legality is not tested, but I happen to know, because of everything that I know about the functioning of society, that this is a particularly weak point in the tax code and would be a good place to shelter profits and those kinds of things.

And then there does seem to be some kind of-- the more a legal task veers into the routine category, the more it's, hey, we just need to know how many times in these documents these things are said or something like that. But, I mean, to me, it's useful to recognize that, I mean, people were stressed about that in 2017 with the machine learning technology. So there are still lawyers in the United States. There are still law schools. Our legal system has not profoundly changed from what machine learning offered.

In fact, I think legal education has probably barely changed at all on the basis of these things, except maybe you could take an-- I don't know this for sure. I have no expertise in this. But I assume you could, take an elective in computers in the law or something like that or machine learning and the law now you wouldn't be able to do before. Do you guys have thoughts?

AUDIENCE: Yeah. In terms of when you initially say back about lawyers and not being able to generate scripts, yeah, I would very much agree. There's actually a case where I think two lawyers got heavily sanctioned-- I think it was with the airport-- where they pretty much just tried to generate an entire script, purely through ChatGPT.

JUSTIN REICH: I think it was a filing, and I think the filing included fake citations and stuff like that, which is really not the best.

AUDIENCE: Yeah, and you could definitely see it, where basically, if you give it-- you could basically make a much better search database using-- if you happen to stick in all these thousands of cases where all these lawyers are pretty much having decisions that change, and maybe one that benefits your stance could have been ruled on earlier on in that year. Maybe it's for the specific county, there is something different. I definitely see that in the future-- I don't know-- even if within that five years--

JUSTIN REICH: Yeah, being able to do scans of precedents for a case or something like that.

AUDIENCE: Yeah.

JUSTIN REICH: Maybe one other domain we should talk about before we head out is just, one of the biggest places in which we will see pushes in the K-12 education system with new tools will be in formative feedback for writing. There are many, many people right now who are building systems so that you assign students to write an essay, they turn in the essay to you-- or before they turn in the essay to you, you offer a bunch of formative feedback.

And I'm very interested in the development of these systems. I was talking with someone who's building one at UC Irvine, and I guess I was reading about some others, too. It struck me that one of the things that these systems have the capacity of doing is saying something like, here are 10 or 15 or 100 things that I want you to check for every essay.

Does the paragraph start with a topic sentence? Is there a transition sentence between? Does the paragraph seem like it has evidence? Does the introduction appear to have a thesis? Does the conclusion tie-- you can think about all the things that you've ever been taught to write.

A thing that I'm really interested in is that, my hunch is-- I mean, I thought about these a lot because a lot of what I do is grade writing, give feedback on writing-- is that some of this seems intuitively helpful to me. Oh, it would be great if I could just tell students, hey, before you turn in your assignment, go run it through this thing and make sure that the beginning of every paragraph has a topic sentence, even better than a topic sentence, some kind of mini thesis, something that presents the argument of the paragraph.

But I was thinking, too, probably the most important thing that I do when I give feedback to someone is figure out, of the dozens of things I could comment on, which one I will choose to comment on. One of the things I've learned about as a writing teacher over the years is, it is really not helpful to tell someone everything they can prove in their essay because that's just too much stuff to work on.

I mean, you guys know that I-- I mean, I used to have people turn in printed essays, and I used to make marks on them and things like that. But I found over the years, really what I want to do is be like, these are the two things that you want to work on. And how do I decide what those two things are? I have in my head a model of people's developmental trajectory.

And some of that model is based on all MIT students ever. When I see an essay that looks like this, I'm pretty sure that the next thing they should work on is blank. And then as soon as I've gotten two pieces of writing from someone, then I can be like, well, the thing you were working on last time was this. Why don't you work on it more? Here's the next thing after that to work on or something like that.

That seems like the kind of discernment-- it seems way, way easier to me to program a machine that can find every kind of problem than it is to program a machine that can figure out, what are the next two problems you should be working on as a learner based on who you are, or where I think you're heading, or what's important in my class, or other things like that.

AUDIENCE: I feel like that opens up the idea of breaking out of the trap of routine assessment. If a computer can do those type of things, then maybe teachers can now focus more on, what does revision look like? So a student goes through an essay or writes an essay and then puts it through this machine, and the machine tells you, oh, in these introduction sentences, they could be stronger in X, Y, and Z ways.

And how do you develop those critical thinking skills to take something and feedback and perform it-- or change it in some way? Like you're always saying, anything that you've published was never your first draft, or you said, on day one, of your good skills was being able to just get down a first draft. ChatGPT can do the same for you.

But how do we teach people in a world where AI can give you that baseline? How do we elevate it to a status that is acceptable for a human? I feel like AI can give us a baseline, but humans take that imagination and creativity to that next step. And so you can take the idea of, here's an essay. How would you change it to make it even better?

JUSTIN REICH: Yeah. For these writing software developers, I mean, a big-- well, when I talk to them, I think about, what is the whole system that you're constructing? Not every capacity that you build will end up being useful, not just because it doesn't work, but because using it is not that helpful. You start thinking things like, all right, what if there were these formative feedback systems where I as the instructor can say, OK, start this package of machines or something like that?

Or I select in advance that I want the machine to automatically give you formative feedback on these things. And then on this other class of things, I want you to show me the feedback that you would give to a student, but I will pick from those things, and then there'll be some way for me to add to it. Some of those human-machine combinations seem pretty exciting to me. Good. Well, why don't we wrap there for the day?