



Educational Technology: Crash Course Computer Science #39

Crash Course: Computer Science

https://youtube.com/watch?v=zTi3_I5h5PQ

https://nerdfighteria.info/v/zTi3_I5h5PQ

Hi, I'm Carrie Anne, and welcome to CrashCourse Computer Science. One of the most dramatic changes enabled by computing technology has been the creation and widespread availability of information. There are currently 1.3 billion websites on the internet. Wikipedia, alone, has five million English language articles, spanning everything from the Dancing Plague of 1518 to proper toilet paper roll orientation. Every day, Google serves up four billion searches to access this information. And, every minute, 3.5 million videos are viewed on YouTube, and 400 hours of new video get uploaded by users. Lots of these views are people watching *Gangnam Style* and *Despacito*, but another large percentage could be considered educational, like what you're doing right now. This amazing treasure trove of information can be accessed with just a few taps on your smartphone, anywhere, anytime. But, having information available isn't the same as learning from it. To be clear, we here at CrashCourse are big fans of interactive, in-class learning, directed conversations, and hands-on experiences as powerful tools for learning. But, we also believe in the additive power of educational technology, both inside and the classroom. So, today we're going to go a little meta, and talk specifically about how computer science can support learning with educational technology.

[Intro Music]

Technology, from paper and pencil to recent machine-learning based intelligent systems, has been supporting education for millennia, even as early as human drawing cave paintings to record hunting scenes for posterity. Teaching people at a distance has long been a driver of educational technology. For example, around 50 CE. St. Paul was sending epistles that offered lessons on religious teachings for new churches being set up in Asia. Since then, several major waves of technological advances have each promised to revolutionize education, from radio and television to DVDs and laser-discs. In fact, as far back as 1913, Thomas Edison predicted, "books will soon be obsolete in the schools. It is possible to teach every branch of human knowledge with the motion picture. Our school system will be completely changed in the next ten years." Of course, you know that didn't happen, but distributing educational materials in formats like video has become more and more popular.

Before we discuss what educational technology research can do for you, there are some simple things research has shown you can do, while watching an educational video like this one, to significantly increase what you learn and retain. First, video is naturally adjustable, so make sure the pacing is right for you by using the video speed controls. On YouTube, you can do that in the right hand corner of the screen. You should be able to understand the video, and have enough time to reflect on the content. Second, pause. You can learn more if you stop the video at the difficult parts. When you do, ask yourself questions about what you've watched and see if you can answer. Or, ask yourself questions about what might be coming up next, and then play the video to see if you're right. Third, try any examples or exercises that are presented in the video on your own. Even if you aren't a programmer, write pseudo-code on paper and maybe even give coding a try. Active learning techniques like these have been shown to increase learning by a factor of ten. And, if you want more information like this, we've got a whole course on it here.

The idea of video as a way to spread quality education has appealed to a lot of people over the last century. What's just the latest incarnation of this idea came in the form of Massive Open Online Courses, or MOOCs. In fact, the *New York Times* declared 2012, "the Year of the MOOC." A lot of the early forms were just videos of lectures from famous professors, but, for a while, some people thought this might mean the end of universities as we know them. Whether you were worried about this idea or excited by it,

that future also hasn't really come to pass and most of the hype has dissipated. This is probably mostly because when you try to scale up learning using technology to include millions of students simultaneously with small numbers of instructional staff, or even none, you run into a lot of problems. Fortunately, these problems have intrigued computer scientists and, more specifically, educational technologists who are finding ways to solve them. For example, effective learning involves getting times and relevant feedback, but how do you give good feedback when you have millions of learners and only one teacher? For that matter, how does a teacher grade a million assignments? Solving many of these problems means creating hybrid, human-technology systems. A useful, but controversial insight, was that students could be a great resource to give each other feedback. Unfortunately, they're often pretty bad at doing so; they're neither experts in the subject matter nor teachers. However, we can support their efforts with technology, like by using algorithms we can match perfect learning partners together out of potentially millions of groupings. Also, part of the grading can be done with automated systems while humans do the rest. For instance, computer algorithms that grade the writing portions of the SATs have been found to be just as accurate as humans hired to grade them by hand.

Other algorithms are being developed that provide personalized learning experiences, much like Netflix's personalized movie recommendations or Google's personalized search results. To achieve this, the software needs to understand what a learner knows and doesn't know. With that understanding, the software can present the right material at the right time to give each particular learner practice on the things that are hardest for them, rather than what they're already good at. Such systems, most often powered by artificial intelligence, are broadly called Intelligent Tutoring Systems. Let's break down a hypothetical system that follows common conventions. So, imagine a student working on this algebra problem in our hypothetical tutoring software. The correct next step to solve it is to subtract both sides by 7. The knowledge required to do this step can be represented by something called a production rule. These describe procedures as if-then statements. The pseudo-code of a production rule for this step would say, "if there is a constant on the same side as a variable, then subtract that constant from both sides." The cool thing about production rules is that they can also be used to represent common mistakes a student might make. These production rules are called buggy rules. For example, instead of subtracting the constant, the student might mistakenly try to subtract the coefficient. No can do! It's totally possible that multiple competing production rules are triggered after a student completes a step. It may not be entirely clear what misconception has led to a student's answer. So, production rules are combined with an algorithm that selects the most likely one. That way, the student can be given a helpful piece of feedback.

These production rules, and the selection algorithm, combine to form what's called a domain model, which is a formal representation of the knowledge, procedures, and skills of a particular discipline, like algebra. Domain models can be used to assist learners on any individual problem, but they're insufficient for helping learners move through a whole curriculum because they don't track any progress over time. For that, intelligent tutoring systems build and maintain a student model, one that tracks, among other things, what production rules a student has mastered, and where they still need practice. This is exactly what we need to properly personalize the tutor.

That doesn't sound so hard, but it's actually a big challenge to figure out what a student knows and doesn't know based only on their answers to problems. A common technique for figuring this out is Bayesian knowledge tracing. The algorithm treats student knowledge as a set of latent variables, which are variables whose true value is hidden from an outside observer, like our software.



Educational Technology: Crash Course Computer Science #39

Crash Course: Computer Science

https://youtube.com/watch?v=zTi3_I5h5PQ

https://nerdfighteria.info/v/zTi3_I5h5PQ

This is also true in the physical world, where a teacher would not know for certain whether a student knows something completely. Instead, they might probe that knowledge using a test to see if the student gets the right answer. Similarly, Bayesian knowledge tracing updates its estimate of the student's knowledge by observing the correctness of each interaction using that skill. To do this, the software maintains four probabilities. First is the probability that a student has learned how to do a particular skill. For example, the skill of subtracting constants from both sides of an algebraic equations; let's say our student correctly subtracts both sides by 7. Because she got the problem correct, we might assume she knows how to do this step. But, there's also the possibility that the student got it correct by accident, and doesn't actually understand how to solve the problem. This is the probability of guess. Similarly, if the student gets it wrong, you might assume that she doesn't know how to do the step. But, there's also the possibility that she knows it, but made a careless error or other slip-up. This is called the probability of slip. The last probability that Bayesian knowledge tracing calculates is the probability that the student started off the problem not knowing how to do the step, but learned how to do it as a result of working through the problem. This is called the probability of transit.

These four probabilities are used in a set of equations that update the student model, keeping a running assessment for each skill the student is supposed to know. The first equation asks, "what's the probability that the student has learned a particular skill," which takes into account the probability that it was already learned previously and the probability of transit. Like a teacher, our estimate of this probability that it was already learned previously depends on whether we observe a student getting a question correct or incorrect. And, so we have these two equations to pick from. After we compute the right value, we plug it into our first equation, updating the probability that a student has learned a particular skill, which then gets stored in their student model. Although there are other approaches, intelligent tutoring systems often use Bayesian knowledge tracing to support what's called mastery learning, where students practice skills until they're deeply understood. To do this most efficiently, the software selects the best problems to present to the student to achieve mastery, what's called adaptive sequencing, which is one form of personalization.

But, our example is still just dealing with data from one student. Internet-connected educational apps or sites now allow teachers and researchers the ability to collect data from millions of learners. From that data, we can discover things like common pitfalls and where students get frustrated. Beyond student responses to questions, this can be done by looking at how long they pause before entering an answer, where they speed up a video, and how they interact with other students on discussion forums. This field is called educational data mining, and it has the ability to use all those face-palms and "ah ha" moments to help improve personalized learning in the future.

Speaking of the future, educational technologists have often drawn inspiration for their innovations from science fiction. In particular, many researchers were inspired by the future envisioned in the book, *The Diamond Age* by Neal Stephenson. It describes a young girl who learns from a book that has a set of virtual agents, who interact with her in natural language, acting as coaches, teachers, and mentors, who grow and change with her as she grows up. They can detect what she knows and how she's feeling, and give just the right feedback and support to help her learn. Today, there are non-science fiction researchers such as Juston Cassell, crafting pedagogical virtual agents that can exhibit the verbal and bodily behaviors found in conversation among humans, and in doing so, build trust, rapport, and even friendship with their human students. Maybe CrashCourse in 2040 will have a little John Green AI that lives on your iPhone 30.

Educational technology and devices are now moving off of laptop and desktop computers, and onto huge tabletop surfaces, where students can collaborate in groups, and also tiny mobile devices, where students can learn on the go. Virtual reality and augmented reality are also getting people excited and enabling new educational experiences for learners; diving deep under the oceans, exploring outer space, traveling through the human body, or interacting with cultures they might never encounter in their real lives.

If we look far into the future, educational interfaces might disappear entirely, and instead happen through direct brain learning, where people can be uploaded with new skills directly into their brains. This might seem really far fetched, but scientists are making inroads already, such as detecting whether someone knows something just from their brain signals. That leads to an interesting question: if we can download things into our brains, could we also upload the contents of our brains? We'll explore that in our series finale next week about the far future of computing. I'll see you then.

CrashCourse Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows, like Physics Girl, Deep Look, and PBS Space Time. This episode was filmed at The Chad & Stacy Emigholz Studio in Indianapolis, and it was made with the help of all these nice people and our wonderful graphics team, Thought Cafe. Thanks for watching, and try turning it off and back on again.

[Outro Music]