MIT CSAIL Alliances | Una-May Podcast 2

[MUSIC PLAYING]

Welcome to MIT's Computer Science and Artificial Intelligence Labs Alliance's podcast series. My name is Steve Lewis. I'm the Assistant Director of Global Strategic Alliances for CSAIL at MIT. In this podcast series, I will interview principal researchers at CSAIL to discover what they're working on and how it will impact society. Dr. Una-May O'Reilly's research goal is to improve the application of machine learning so engineers and scientists of complex systems can gain improved insights towards serving others.

With her research group, Any Scale Learning For All, or ALFA, she develops new data-driven analysis of online coding courses, deep learning techniques for program representations, adversarial attacks on machine learning models, model training for adversarial robustness, cyber hunting tools, and cyber arms race models.

Dr. O'Reilly, who holds a PhD from Carleton University in Ottawa, Canada, joined CSAIL in 1996. Her and her group are exploring ways to exploit state of the art artificial intelligence and scalable machine learning for applications in various enterprises including healthcare, online learning, and security.

Una-May, may thank you very much for your time. I'd like to get started by asking you, how is deep learning being applied to code?

Well, all of the computers in the world-- and there are so many of them-- they run on software. And this software is really difficult to write. It's difficult to read. It has the potential for being buggy. It has the potential for being insecure, vulnerable to attacks. And deep learning is being applied to help us understand code better so that we can automatically with our algorithmic techniques look for bugs and vulnerabilities and improve our ability to automatically read and write code.

And that's completely automated?

It won't be completely automated. It's a really, really hard task to understand what code means or to write code. It's, I think, a higher level human function than even language or vision and object recognition. So realistically, we're talking about tools that aid humans in being better programmers and more efficient programmers.

I see. Let's set the record straight. Because there's a lot of confusion about the differences between deep learning, machine learning, deep understanding. Can you just describe or explain for our listeners the difference between, say, deep learning and deep understanding?

Well, that's a great question. There is really the capacity for confusion. Because the meaning of these terms for people who are not engrossed and engulfed in the field must be very confusing. The way we use deep learning in the field is to refer to a set of computer science techniques where we actually apply machine learning to data to arrive at some sort of prediction, or classification, or insights from that data. So deep learning is a way of training and developing models. It's really a method.

Deep understanding is something much more general. I understand you deeply when I go beyond the surface of what you say. And I seek to see what else you mean by what you say. And deep understanding is really about meaning. And the interesting thing is, what we're seeking with deep learning and all these algorithmic techniques is to have a deep understanding of some piece of data, whether it be code, or it be text, or it be images.

We're really seeking that deep understanding. And we're not there yet. Right now, deep learning-- the deep comes from not semantically [LAUGHS] deep, but the deep COMES from the fact that we build these neural networks. And they have very, very high dimensionality. And in a very, very high level way, it really refers to the architecture of the neural networks we're building, not anything about the depth of semantics.

I see. And what are some of the challenges you're facing in this area of research?

Well, it's really hard to understand code. But we need to have a deep understanding of code before we can try and automatically, for example, find a bug, or see whether someone has altered code to make it run maliciously. So for example, my group, we really want to understand how people understand code. [LAUGHS] Because a lot of our methods should follow that.

And one of the most interesting questions right now in code-understanding research is, how much of code understanding is the responsibility of the language system. Remember, we have programming languages. So if I learn a programming language, is it my language system in my brain that's helping me understand code, or is it actually that code is really mathematical? And is it-- do I deeply understand what code does and how to write code because I rely upon mathematical and logic processes within my brain.

And could you tell our listeners a little bit about your group?

So my group is called ALFA, Any Scale Learning For All. And we work on data science using data-driven techniques like deep learning. And we also look at simulation and modeling with evolutionary techniques. And really, one focus we have right now is adversarial intelligence.

So we're very interested in how conflicts require intelligence. And they escalate, because of intelligence. And our first step in decoding was really thinking about coding as leaving vulnerabilities that attackers could actually exploit.

So I will get into adversarial networks. And maybe you could give us a high-level, or our listeners a high-level understanding what that is. I know you could, for example, when you're doing machine learning and deep learning with images, you could inject some noise in there. You could-- it could be off by a few pixels, and it could change the outcome of the result. Can you tell us how the work in your group is using adversarial networks?

Correct. So parallel to an image model, which can understand what images do, you can-- with machine learning, we can develop detectors of malware. But these machine learning detectors of malware are deep learning networks. And they can be attacked just the same way as models that are doing image processing can be attacked.

So my group has looked at the nature of those attacks, whether they take place on models of code that are binary, because you often run a lot of apps with their binary representation, or whether the adversary is actually modifying source code. And we're trying to detect whether the source code has been modified.

Very interesting.

That's malware too, right? Everything's malware.

Yeah, [LAUGHS] way too much of it these days, for sure. And the attackers are getting more sophisticated in hiding it. Can you tell us some of the ways you work to replicate or assist a human code?

So we have a project where we're looking for bugs in smart contracts. And smart contracts are the software programs that run when you're running a digital currency. So we had a project where we actually scraped smart contracts from the digital memory. And we were looking for the vulnerabilities that someone might be able to come in and exploit and steal digital assets.

So that, I mean, this smart contracts dovetails with blockchain, right? So--

We were looking at Solidity. Solidity is a language for writing smart contracts in Ethereum.

I see. Thank you for explaining that. Why is understanding how humans read code so important?

Well, I think there might be people who disagree with me and say, I don't care how a human reads code. I just care that my computer-- I can write an algorithm that understands code. But my belief is that understanding how humans read code tells me what the important aspects of code are that I need to account for when I build a machine learning model.

So I alluded to this a little earlier. It matters whether your language center is being employed to look at a piece of code. So does that mean that you're reading a piece of code, and it's just like reading text? Or conversely, if you're looking at a piece of code and you're thinking about it mathematically, that would intuit to me that there's different kinds of models of code that I need to create to express its meaning.

Interesting. Can you explain more about the difference between-- is there a difference between reading code from a semantic point of view and then reading it from a mathematical point of view?

Yes. Well, we did some experiments to try and understand that. And we collaborated with our colleagues in neuroscience here at MIT. In particular, we worked with a specialist in language. And she understood how language is processed in the brain. So we did fMRI experiments, where we showed subjects pieces of code.

And then in contrast, we showed them a piece of text that said, that described the same actions taking places in the code. And we were able to observe through fMRI with a number of sophisticated methods and experiments, we were able to determine whether the language center was actually being engaged to process the code as much as it was being engaged to process the text.

Interesting. And how--

We literally had a-- we had a survey of about 20 to 30 people who had competence in writing Python. And we pre-tested them to find out their abilities. And then we came up with a whole protocol of contrasting questions that allowed us to tease out the use of language versus mathematical reasoning from those examples while we were watching their brains in the fMRI machine and looking for activations in the right regions.

And is this work published?

Yes, it's published. This work is published in *eLife.*

Great. And what were the conclusions? Were there any conclusions?

Well, we had a small sample. And it was for me as a computer scientist, it was a learning exercise. It was somewhat surprising to me how careful we had to be about every aspect of experimental design, and of course, experimental data and analysis.

And we had great help from our computational neuroscience colleagues on this. We found that the role of language was less than the role of-- in general. And that's a very, very-- I hesitate to put that response out there. But things that we thought would be more language-oriented were not.

I see.

We also did an interesting experiment where we also had data from children looking at visual programming languages. ScratchJr is a language which doesn't have any much of text or mathematics. It's all visual. And so, there's no language there. So it was also useful to look at what was still being activated regardless of whether the code was being presented as more procedural code, like Python versus ScratchJr.

Fascinating. So how can understanding the way humans read code benefit industry or academia?

Well, I think the first thing to recognize is just how important software is to industry. As I said when we started, everything is computation. And software is the currency, the electricity that runs through your system, your computer, to make it do the right things. So we're very interested in things like improving the capacity to write code without bugs. Some bugs are just so hard to find.

Right now, we're thinking about concurrency bugs. Those are bugs that happen because many, many programs are running simultaneously. And they're sharing access to data. And sometimes, there's in a very, very modest amount of time, there's a race condition, and two things grab the data in the wrong order, and the data is corrupted. So there's all sorts of very important aspects of computer software development that good deep learning models of code can help address.

And this goes beyond, say, what a compiler would do in flushing bugs out or errors in--

Yes, compilers are very good at flushing out your bugs in syntax. But they don't help you if you get the meaning of your code wrong. They just they just help you largely if you get the syntax of your code wrong. Like you might call a variable that isn't declared, or call a variable that's out of scope, or have a-- they might be able to find a run-on loop for you.

But these models are going to understand the purpose of the code, the intent of the code. And some models can even summarize it-- code, and say what code is doing. And some of the latest work that's just much more recent is really exciting, where you can describe what you want, and the model generates the code. That's pretty exciting.

So is this-- yeah. I mean, from-- I'm not a coder. But I mean, I would love to write software and be able to just to tell at a high level what I'd like the inputs and the outputs to be and the program to do. Is that where this is headed?

Yeah. Imagine a world where in academics that's how we teach programming. Right now, we force students to acquire a new language. They have to have a mental model of a computer. And if they really want to program the computer, they have to put those two things together, that mental model and the ability to compose in that programming language.

If we could relax programming languages to be much more human-like, and maybe we could understand how to do that if we understand how humans understand code. Then there's a whole capacity to advance programming and software engineering beyond how we see it right now.

That's great. Is there any other insights or of your research that you're excited to share with our listeners?

Well on the same note of code, one thing I've been very interested in with my team is understanding how you learn to code. So one of the things we do is, we look at the data for students who are taking online course in introduction to programming.

And what we're able to do with data science is to actually take each one of those students' interaction sequences as they move through the digital platform. And we can analyze those digital sequences to come up with a narrative of how a student is learning. And I think that's another piece of it. How do we learn to program is another aspect of this whole research into understanding code.

Where could people go to find out more about your research?

Well, they can always Google me. I'm the only Una-May O'Reilly in the planet. And they can Google the ALFA group at MIT-CSAIL.

Great. Well, Una-May, thank you very much for your time. It was very fascinating.

Oh, you're most welcome. It was a pleasure.

[MUSIC PLAYING]

If you're interested in learning more about the CSAIL Alliance program and the latest research at CSAIL, please visit our website at cap.csail.mit.edu. And listen to our podcast series on Spotify, Apple Music, or wherever you listen to your podcasts. Tune in next month for a brand new edition of the CSAIL Alliance's podcast and stay ahead of the curve.