Tom Dietterich Oral History Interview, June 10, 2015

Title
“Teaching Machines to Learn”

Date
June 10, 2015

Location
Valley Library, Oregon State University.

Summary
In the interview, Dietterich recounts his upbringing, family background, and early interests as a child, including his fascination with the NASA space program and his first encounters with computing. He then describes his undergraduate studies at Oberlin College, noting his heavy involvement with student government and his academic work in mathematics. He likewise discusses his continuing infatuation with computing - which he pursued both for fun and for employment during his undergraduate years - and his initial exposure to the foundations of machine learning.

From there Dietterich recalls his master's studies at the University of Illinois, including his research on the use of machine learning in plant disease diagnosis and his work in game theory. He notes a summer job that he held at Bell Telephone Labs, describes the circumstances by which he moved to Stanford University to pursue his Ph.D., and details the investigations into applied artificial intelligence that he conducted as a doctoral student.

The session next turns it attention to Dietterich's association with Oregon State University. He recounts his move to OSU as a "trailing spouse" following his wife, who had already obtained a faculty position in the Department of Botany and Plant Pathology. He likewise shares his memories of his initial impressions of OSU, the ragged state of the Computer Science department at the time of his arrival, and the steady improvement that the department has seen in the years since. He also details the technical infrastructure that was available to him during his early years at Oregon State and shares some background on the use of email during its infancy.

Dietterich then discusses several high points of his research career, including a few landmark papers that he authored that helped to advance the field of machine learning. He also describes his association with Arris Pharmaceutical - including his contributions to the company's initiative to synthesize a variety of molecules - and his work on hierarchical reinforcement learning. He specifically reflects on his interest in applying machine learning to ecological problems, discusses his collaboration with the Cornell University Laboratory of Ornithology, and notes his involvement with private sector enterprise projects, including recommendation software packages developed by Strands, Inc. and BigML.

The session concludes with Dietterich's thoughts on the forward march of artificial intelligence and his perspective on popular fears that humans might someday lose control over the technologies that they have created.

Interviewee
Tom Dietterich

Interviewer

Chris Petersen

Website
http://scarc.library.oregonstate.edu/oh150/dietterich/

Transcript

Chris Petersen: Okay Tom, if you would please introduce yourself with your name and today's date and our location?

Thomas Dietterich: Okay, so my name is Tom Dietterich, today is June 10th, 2015 and we're in the Valley Library on the Oregon State University campus.

CP: Terrific. So, we'll talk a lot about your career at OSU and your work in machine learning, but I'd like to begin with some sort of biographical information to start with. Where were you born?

TD: I was born in South Weymouth, Massachusetts, but I don't have—we left Massachusetts when I was three, I'm told, so my earliest recollections are growing up in Iowa. So, I lived in Iowa for ten years in two different towns and then we moved to New Jersey and were there for five years, I think, and then Illinois for two years and then I went to college at Oberlin, Ohio for four years, two years at Urbana–Champaign for a master's degree, five years for a PhD at Stanford and then I came to Oregon State, and I've been here ever since.

CP: What were your parents' backgrounds?

TD: So, my father is a Methodist minister who got his doctorate in theology, so that's why they were in the Boston area, because he was going to Boston University School of Theology. My mother was working as a secretary at MIT, I believe, when they met. Her brother was finishing up his PhD at MIT, so there's education for a few generations in my family, I guess.

CP: And religion too.

TD: A mix of religion and engineering, actually, because my great-grandfather on my father's side was an engineer for the railroads in New York state, so that's the only one. And my grandfather on my mother's side was a teacher of English and an athlete in pole vault. And so, yeah, variety of backgrounds but generally a fairly—I mean both parents had college degrees.

CP: Did you have any siblings?

TD: Yeah, I had three sibs. I am the oldest of the three. Interestingly, all of us went into the computing and electronics industry in one fashion or another, although my sister later has recently started a second career teaching music in middle school. And there's another theme that goes through all this, is music as well, which is pretty common, I guess, for people in sort of the mathematical computer science side of things, also tend to—there's a high frequency of people who are also involved in music.

CP: Why do you think that is?

TD: I don't know.

CP: Just the way their brains work?

TD: Yeah, who knows?

CP: So, I'm interested in—I always ask about community life. You moved around a little bit, it sounds like, but maybe we can talk about in the context of being a minister's son growing up in community life for you?

TD: It probably made it easier when we moved to a new place, because we had some sort of an existing social matrix to fit into. But because he was a Methodist minister as a church pastor when we were in Iowa, and the Methodist church moves their pastors around, as you may know, so that was one of the reasons we moved. But then when we moved to New Jersey, he had taken a positon with the National Council of Churches in Manhattan where he was their representative to the United Nations. So, that was where I learned what NGO meant, because he was an NGO. And so, from that you can infer that my father has an interest not only in theology but also maybe in a practical component of living in the world and

influencing the world. So, I think I inherited some of that desire to make a difference in the world and to be interested in the policy implications of what we do, as well as the technical, puzzle-solving side of things.

CP: It sounds like a household that was rich in ideas.

TD: Yeah.

CP: What were your interests as a boy?

TD: I loved electronics and astronomy. The space program totally captured my attention. One of my earliest memories was I was allowed to watch all the launches on our black and white television, but this—I'd started doing that before I could read, so one morning I got up at six am or whatever it was and just sat watching the television, which had something on it I couldn't read, which what it had on it was a message that the launch had been canceled. And my mom woke up, then she came and explained it to me. So, that shows kind of how early on I was very fascinated by these things.

[0:05:15]

And so, I partly got interested in computing because I was interested in understanding the orbits of spacecraft. One of my hobbies in high school was trying to observe satellites as they went overhead at night and get enough fixes on their location and time that I can predict their return. I was never able to do this. I learned all about spherical trigonometry and all the mathematics of orbits, and if I was willing to assume the orbits were circular, which was probably not too bad an approximation, then I only needed five coordinate fixes, but what I didn't understand was that when an orbit is very big, those five fixes are almost going to be collinear. And so, the estimation error is immense, and that's probably why, either I was always dividing by zero in the computer, or just it was really not statistically possible. So, that was kind of an introduction to the challenges of working with data.

But you know, in those days we didn't have the internet and you couldn't just get a map of the sky or anything. I had to go to a sort of regional library to get star charts that had all the right ascension and declination of all the stars so that I could map between those coordinate systems and earth coordinate systems and observe the center coordinate systems, anyway. So, that was my first introduction. That and because my uncle, my mom's brother who had done his PhD at MIT, was one of the first programmers of the Dartmouth Time-Sharing System, which was—well, he was working for General Electric, which provided the computers for the Dartmouth Time-Sharing System, which was one of the first time-sharing computer systems that was used for a student population, but a non-research thing. And so, one time when we were visiting his house, he actually lived in Maryland and in some sense was telecommuting to Dartmouth to be this—to do systems work there. He taught me how to program in Basic and we wrote a little computer program in Basic using the teletype machine in his living room, which is, those are incredibly noisy machines.

And so, that was when I first learned about programming and it was a very natural and fun thing to do, really gives you a sense of being able to get this computer to do things that you want it to do and to make your programs beautiful. So, there's sort of a—there's both the power of controlling a very complex machine and also the craftsmanship of writing the programs, it's very, you know, is really a lot of fun. I don't get much chance to do that these days now.

CP: About how old were you when this happened?

TD: I don't know. I would guess probably like twelve, thirteen maybe, something like that.

CP: So, that was sort of for your first substantive contact with computers?

TD: Yes, yeah. Definitely. But I mean, I tell students now that when I was applying for college, a very important question for me was what computer do they have, because every campus had one, and the question was, was it a digital equipment corporation system or an IBM system, and was it time-sharing or was it punch cards.

CP: Did math come easily to you, as well?

TD: Yeah. So yeah, I was always doing—I mean, I liked all math and science, pretty much. And I was also the kind of kid who was always taking apart everything. So, who knows what chemicals I got exposed to, taking apart capacitors

and transformers. And I wanted to be able to communicate secretly with my brother by Morse code in our house, so I unwound the wires out of a transformer, which were like hair-thin wires, and then managed to route wires down through the basement from my closet to my brother's closet, which was probably a total of maybe ten feet. But this is where I learned about resistance. I thought "well, we're electrically connected now. Now I put this 6-volt battery at one end and I should be able to turn on the light at the other end," but the resistance of everything wire is much greater than what would allow that to go through. So, there were many lessons learned through practice. And I know when we moved from the first house in Iowa to the second one, my parents were greatly relieved that that house had circuit breakers instead of fuses, because I was always just popping the circuit system.

[0:10:14]

I had a concept I'd learned about, electromagnetism and creating a magnet, and I thought well, a slinky is a big coil thing, so if I could just plug that into the wall, I could get a really good magnet, but I didn't really understand about alternating current or the fact that that would just be a direct short circuit, basically, because the slinky—I don't know if they make slinkies out of metal anymore. In those days, it was high-quality conductor. So yeah, I had that kind of reputation. It's somewhat surprising that I didn't kill myself through electrocution before I reached puberty, but I survived.

CP: Did you go to more than one high school, or were you in one high school?

TD: So, my high school I had done the first two years in Glen Ridge in New Jersey, which was an extraordinary experience. In those days, all funding for schools in New Jersey was completely local, so it was incredibly inequitable. Glen Ridge was a small suburb who, I think a major reason for its existence was its school system, and people moved there and paid very high prices in order to basically—it essentially functioned like a private school. I remember the guidance counselor's office just had the plaques of the Ivy League schools on the wall, and that was kind of the expectation. And something like ninety-five percent of the students went onto college. So, that was absurdly high.

After two years there, we moved. My dad's job with the National Council of Churches ended and we moved to Naperville, Illinois, which is also a very educated community, but of course a much more typical cross-section, a normal prosperous town that had a—there's a Bell Labs, AT&T—well, now AT&T research center there, and Amoco Research. So, it had a lot of educated people. But that high school was just a kind of ordinary good high school. But in both cases, yeah, I had lots of access to things. One of the good things about moving to Illinois was that you could actually take electronics, because the high school had a lot more auto shop and electronics, because ninety-five percent of their students were not going on to college, and so I really enjoyed having access to the sort of practical engineering kinds of things, as well.

CP: So, I'm gathering you were on the college track from the get-go.

TD: Yeah, that was never in doubt.

CP: Did you have a sense of what you wanted to pursue as an undergraduate?

TD: No, when I got to college I was—so I chose Oberlin partly because of its history of the engagement in politics and policy, and I think my first idea for a major was in psychology and I actually took my first statistics course as a social science methods class, which I thought was really interesting. And then I was interested in political science and thought well, maybe I would go into the Foreign Service. I started taking Russian, which I found incredibly difficult. But all along, I was always taking mathematics classes and I was working at the computer center.

And I got banned from the computer center for one semester because a friend of mine and I had broken into the—gotten our way through the security system of the computer, of the time-sharing system, to understand how it worked. But eventually we worked out a thing where I was given a special account so that I could do useful things, but if the system ever crashed and it was clear that I was the user that had caused it, they would know because of the account name. My account name was $crash; that they should ignore it, that it wasn't a real problem. So, they were very tolerant, I think, of us budding hackers.

So, I was also very heavily involved in student politics. I was on the sort of—so Oberlin, like a lot of these elite liberal arts colleges, is really the faculty governs the institution. So, there are something called the General Faculty, which would meet on a regular basis, I think maybe weekly or biweekly, and conduct, approve of all the policy changes and things like

this. And the student government was called the Student Senate. We had some number of representatives who were voting members of that body.

[0:15:31]

And so, I had been involved in the high school, of course, student council type things, so when I got there I immediately got involved in student government and I was a representative on various committees and so on throughout the whole time I was there. I sometimes joke that I probably majored more in student government than I did in my studies. But I did complete a bachelor's degree in mathematics with an honor's project in statistics and operations research. But I had been doing a lot of software engineering and computer programming. In those days, you couldn't major in computer science. There were maybe only three or four undergraduate Computer Science programs in the country.

So, I didn't really know what computer science was. I knew what programming was but I just didn't know what the difference really was. So, I had been, for instance, writing computer programs in COBOL to support the vice president of finance, to help manage the college's investment portfolio, and I was writing programs to help the student services, like billing. I wrote the first revolving, credit-charging program for Oberlin, so that students, since students just had to pay by a certain day, right? Then we figured out how to charge interest and so on.

And so, I had done a lot of programming and my original thought after college was that I would maybe get a master's in computer science and maybe a master's in business administration, because I had this kind of more practical outlook on things. But when I—before I even got to Illinois I was contacted by a faculty member who was looking for a research assistant and he was working in this area that we would now call machine learning. In fact, he's responsible for naming the field and he's got—his name was Richard Michalski. And so, he interviewed me and hired me, so when I arrived I already was involved in this project. I didn't really know anything about it at all; really nothing. I didn't even know what computer science was.

So, when I got to Illinois, I was taking a lot of essentially undergraduate classes to fill-in my missing background, since I had never had any of these things, and started to get a sense of what are the research questions in computer science, the design of algorithms, analyzing algorithms to understand is it impossible to—what is possible to do. You have a computational problem and you might say well, the only algorithms I've been able to develop for this are extremely slow, and as the problem gets big, they get slower and slower. They might take an amount of computer time that grows exponentially with the size of the problem. Is there any way, am I just stupid or is this problem inherently hard? And that's one of the fundamental questions in computer science that we still don't actually know the answer to, but we suspect that the problems are inherently hard. We just can't prove it.

But one of the things I got really interested in was; so the area of machine learning that Professor Michalski was studying was really a question of if I'm given a set of data points—he had a famous example of suppose I'm observing trains going by on the railroad line and there are trains that are eastbound and trains that westbound, and maybe I look at what kind of cars are on each train and what's in each car; can I, if you just told me the number of cars and what was in each car, could I tell you whether it's eastbound or westbound? So, that's the kind of question, is from observational data, can you learn a prediction rule or classification rule?

So, the observational data could be an image and you're trying to say is this an image of the letter A or the letter B, or Z or T or whatever, or it could be as the kinds of things that one of the projects I've done at Oregon State is; it could be an image of an insect living in a freshwater stream in Oregon and we might want to know well, what species is this. So, but it's a step toward—I guess the reason it's called machine learning really dates back to the idea of artificial intelligence and the idea that if we want to create computers that can match human intelligence, people learn from experience. Somehow, how can we make computers learn from experience?

[0:20:40]

And this is sort of the simplest kind of learning you can imagine, which is just can you learn what the difference is between a chair and, I don't know, a television. So, I got interested in that, but then I had taken a class at Oberlin on philosophy of politics, or really political science, and my professor there—this is widely regarded as one of the best classes on campus, so everybody took it—and the faculty member was a guy named Harlan Wilson and he often said that

for students in the class that were taking—who were scientists—he would have them read Marx, who claimed that he was a scientist, and then he'd have us read Thomas Kuhn's *The Structure of Scientific Revolutions* and sort of, you would write a paper that said "well, according to Thomas Kuhn, he's a Marxist Scientist," or "what is the relationship between these two?" Because I don't think Thomas Kuhn would agree, but there is a certain sense in which Kuhn is taking a kind of Marxist analysis of science, in the sense that's he's saying it is not necessarily an objective process; it's wrapped up in a whole paradigm, a way of looking at the world. But by the same token, I don't think that anyone would say that Marx was following any kind of a scientific method. So anyway, it was a very stimulating project, obviously, because I still remember it.

And so, when my advisor was telling me "well, I want to have computers learn from data," I though well, what's known in the philosophy of science about this? Is this even possible? And so, I started reading a lot of work. I spent many days just sitting in the—at the University of Illinois, the library is not one place, it's in like twenty branch libraries, and one of them is the philosophy library, which is actually in the main building but it's like it's its own little closet. And so, I spent many days sitting in there reading lots of contemporary philosophy of science. And I mean, I think as with all philosophy, it gives you an appreciation of the issues but not really any answers. But I've always been interested in kind of the foundational questions of machine learning. So I don't know, I'm not sure what the original question was.

CP: That was very good, a couple sort of asides. I'm interested to hear about this interest in politics and policy and student politics. Were there particular issues that kind of drove you in that direction? Or was it just kind of a more general interest in the world around you?

TD: That's a very good question. I was certainly raised to be a person who participates, so joining organizations, helping make things happen, that's very high value. However, there's another part of me that is maybe more the scientist part who just wants to understand how things work. So, I found it fascinating to just—the college was going through rounds of budget cuts and reallocations, as colleges always are. I wanted to understand how the college budget works, like what were the categories and why were they spending so much on this and not on that.

And so, I got to know the vice president for finance, and I think he was probably quite amused or befuddled by me, but he was willing to spend not many hours, but some number of hours explaining the budget to me and giving me a detailed print out of everything except salary lines of how the money was being spent. And I just pored over that budget, because I was just fascinating by understanding how a complex organization worked.

[0:25:16]

It's interesting, because after that experience as an undergraduate in student politics, I have not ever gotten involved in faculty government at Oregon State. I think I got it out of my system. But I just mention it because I feel like the twin things of just wanting to understand how the world works and to be a witness to things, how things happen in the seats of power, was probably a greater motivation than to have power, to be able to exercise it. And so, sort of like the scientist's or the journalist's kind of view of things, which is I wanted to be a witness to things as opposed to I wanted to control things.

So yeah, I didn't have a strong policy objective. I mean, of course I was raised in the sixties, and so the Vietnam War was a huge factor, but I was just young enough that I was only eligible for the draft one year, essentially, and I had a ridiculously high draft number, so I was not at risk. But I did participate, as the son of my father, who was participating in the marches in Washington in October and November. I can't remember which year that was. '68? And so that was also a part of my upbringing, was driving down New Jersey Turnpike with thousands of other cars all going to D.C. to participate in that. And of course, that was one of the things that attracted me to Oberlin, because of course they had played a key role in the post Kent State situation.

But you know, what were the big, the burning issues that came up in student governments while I was there? Trying to convince Amtrak to put a station stop in O'Leary, Ohio, which would be the closest place for Oberlin, which we succeeded at. I'm trying to remember it. That was our big policy success. I mean, we were very focused on process and making sure that the student voice was being heard in the Educational Programs and Policies Committee, which sort of decided what the majors were and things like that, sort of like the Curriculum Council here. But in retrospect, I can't

remember that there were any—I mean, I think there were popular professors that maybe weren't getting tenure, various kinds of things like that that would happen.

CP: The other question I have is you talked about reading in the philosophy of science; as you got more interested in machine learning and artificial intelligence, I wonder if there was any exposure to science fiction as well? I mean, kind of the culture, HAL 9000, for example, that's sort of the default to that kind of really early artificial intelligence idea.

TD: Well it's interesting, because I read some science fiction but I was much more interested in science. So, I found like Gamow's *One Two Three...Infinity* book a lot more fun than Isaac Asimov, or something. And I loved *2001: A Space Odyssey*, but not for the AI; for the whole concept of space travel and going to the moon and so on. I mean, I was totally obsessed with the space program, as I indicated. I built a thing that I called the Gemini Tracker, which is basically based on a bunch of old telephone relays, a thing that had a map of the world and showed the orbital positon by lighting up little light bulbs in sequence, automatically. And that was kind of my ongoing electrical project.

So yeah, I was much more interested in science fact. The space program totally captured my attention and my enthusiasm. I had a full-scale model—or well, I had a plastic scale model of the Saturn V and all of the Apollo stuff. I rented a television just to watch the landing on the moon, back when I was whatever age I would have been. That was '69, so I was like fifteen. So, I think I was very amusing to my parents.

[0:30:02]

CP: Was there a desire to be an astronaut? Or was it more of the science and engineering side of it?

TD: I loved Herald Masursky, who was a planetary geologist from Flagstaff, Arizona. I thought that was a great job. Boy, studying, understanding how the planets work. Yeah, I didn't think I was cut out to be an astronaut, but Mission Control. Also, maybe being Walter Cronkite. That was also really cool. I liked being surrounded by all kinds of displays.

CP: Alright, back on track, back to Illinois. So, you sort of referenced that you were brought into this research project. I'm interested in the specific work that you did as a master's student.

TD: Well, I joined a project that had been working on doing plant disease diagnosis using machine learning. But that was pretty much done and my advisor was working on what he saw as the next stage, so that—yeah, I don't know how much detail to go into, but there are kind of two ways that you can describe inputs, I guess, to machine learning. One way we sometimes call the flat or feature vector way. So, if we were describing the trains going east versus going west, we might just say "well, there were six cars; three red ones, two blue ones, three cars were carrying coal, two cars were carrying grain," something like this. So, we would have some number of variables, and it would just be a fixed number of variables and we'd describe the train that way. The other way is what he called structural descriptions, and that's the idea that he would say "well, there's an engine and then connected to it right behind it is a car that has grain, and right behind it is a car that has coal, and right behind it has another coal," and so the size of the description would depend on how many cars were in the train. It would actually mimic the structure of the train.

Working with those kinds of structural descriptions, which nowadays I guess we would call relational descriptions, because you'd say there's this behind or immediately behind relation between each pair of cars, and then maybe the contents of relation for each car that says "car three has, its contents is corn" or something. Working with those kinds of relational or structural descriptions is much more computationally expensive, but more expressive. So, there are many things that you can say. You can say "well, maybe it's the trains going west, always the first two cars after the locomotive have coal," or something, which you can't say if you just say "well, there are two cars in the train with coal." So, he was interested in pushing into that area, and so I worked on implementing that.

For my actual thesis project for my masters, I worked on a card game. So, in the early days of artificial intelligence, we did a lot of work in games. And Martin Gardner had this mathematical puzzles column in the *Scientific American*, and he published this description of a game that had been invented by some guy, I don't remember his name right now. It was called Eleusis, and it was basically a game where there's a dealer who invents a rule, and the players have to figure out the rule. And so, it's a kind of game in which it requires that you find a pattern in the—the way it works is the dealer makes up the rule and then he plays I think one or two cards, which are going to form a sequence. And there might be a rule like,

the rule might be something like you need to alternate red and black cards, or odd and even cards or face and non-face cards, or after every face card you must have a card that has a value less than six, and then after a non-face card you can have any value, or something. So, it could be these complicated rules. And so then, players then in their turn, everybody's been dealt some cards, they try to play a card to try to extend the sequence, and then the dealer either tells them they're right or they're wrong, but doesn't tell them what the legal thing would be. So, nowadays we call this banded feedback, but the idea is you only get feedback if you got the right answer. You're not told the right answer. So, sort of like the Socratic style, I guess.

[0:35:07]

And so, I wanted to write a computer program that could play this game. And then to collect data we would have these Eleusis parties where with the other people in the lab we would try to come up with rules to play the game, and then I would collect those rules and the games and feed them to my program and see how well it could do. So, that was my master's thesis. I'm not sure that it had any lasting value, but it was a lot of fun. And I did have some correspondence with Martin Gardner about it after we solved it, because he thought it would be impossible to solve.

And I'm not sure, I mean it's a kind of open-ended game where it is never solved because, at least my program basically—I analyzed a whole bunch of these games and found that there was sort of three general styles of rules that came up, and so I built sort of separate algorithms to match against each of those styles. But if you could come up with another one that violated those, I would find it. Although one thing that's kind of interesting is I might find something that was pretty close and you might still be able to win the game, because you never, the players—no one ever actually has to articulate the true rule. Instead, if a player thinks they've got it then they announce that they would like to become "the prophet," and basically now when each player plays, they first prophesize what "God," the dealer, will say, whether it's correct or not, and if they can survive as prophet until the end of the game, then they win. So, you can see there's an interesting kind of philosophical, philosophy of science aspect to it. We never know the truth, but we can be good at predicting.

CP: What was the sort of, I guess, technical environment in computers at that time? Was it a timeshare situation still, or did it—

TD: It was a timesharing situation, yeah. I was writing in Pascal on a CDC 6600, which was really a big scientific computing machine. One thing that's kind of—I mean, Illinois is still one of the top departments, but they were really known for designing and building computers, not for software or algorithms. It's kind of ironic. I took kind of software and artificial intelligence classes, intellectual engineering, so they kind of got things swapped, whereas in the Computer Science department we had classes on computer hardware, computer architecture, lots of low-level stuff, circuits, so really they need to relabel the departments, or merge them or something. So, most of the coursework I took was actually about computer architecture, and at that time, Illinois was working on the ILLIAC IV, which was a big parallel computer, big in that day; I guess it had like thirty-two processors, distributed memory parallel computer that was built for NASA. I never got to work on it, but the professors teaching classes were designing it, and so that was really interesting.

CP: But how did the timeshare work?

TD: You sat at a terminal. I think by that time we had video terminals. We may also have had these things called DEC Writers, which were kind of like dot matrix, or really it was a spinning wheel that would be hit with a hammer to—they call them Daisy wheels or something, I can't remember now. So, it was printing on paper, but it was interactive.

CP: Did you do any teaching?

TD: Not at Illinois, no. I was only there two years.

CP: It sounds like you had a summer job at Bell Telephone Labs as well, was that—

TD: Right, so after my second year at Illinois, I decided to move on to a different school, partly because it was clear that if I was going to work in artificial intelligence, Illinois was not the place to do that. They only had a couple of people there and the advice I had been given was "well, maybe you would do better elsewhere." And I had—I'll get back to the summer thing in a second—so while I was working with Michalski at Illinois, we were visited by Donald Michie, M-I-C-H-I-E, and Donald Michie was one of the sort of, kind of early researchers in artificial intelligence in the UK. He'd

actually worked at Bletchley Park with Turing, et al. during the war, and then he had been an academic at the University of Edinburgh, as well as having a consulting company on the side, where he was, at that time, doing some applied artificial intelligence work.

[0:40:26]

And so, he came for a sabbatical and then I guess he had known my advisor in the past. But then Michie was at Stanford doing a sabbatical there and teaching, teaching a class in, I'm not sure what he called it, machine intelligence or something. Michie was most famous for actually editing a wonderful volume of a series of books called *Machine Intelligence Series*. He would run a summer workshop in the UK where people were given a half day to describe some AI program they were building, and there would be a very lively discussion about it. These ended before I came along, so I never got to see it, but then each of those discussions was turned into a paper and a chapter in this book, and they were influential, very well-written, because he and his wife would edit them very carefully. So, it was a very high quality thing.

Anyway, so Donald Michie was teaching at Stanford and he was very interested in chess, computers playing chess. He would tell stories about having played chess with Alan Turing, and of course Turing was very interested in computers playing chess, too. So, in those days people were looking at the end game in chess when you have very few pieces on the board, like king and rook versus king and pawn or king and knight versus king and rook. These end games, it was possible using a computer program to play the game backwards. So, you could start with all situations in which one side had lost and then you could make all possible reverse moves back some level, say twenty or thirty or forty moves. And so, you could figure out, for each of those positions, who could win, what was the optimal way to play that position out to win or to lose, so you could know what we could say is the value of each of those positions.

And so, the idea was if you had a computer program that was trying to play chess, when it got to one of these board positions that had been previously enumerated by reverse play of the game, you could know how to finish the game perfectly, which was very valuable because the plays toward the end are very much like puzzles, I guess. I mean, I'm not much of a chess player, but playing them exactly right, it becomes much more like an algorithm and much less like looking for it, searching for it, especially if it's twenty moves into the future to the end. People can't really see that far. So, chess masters would also study these end games.

Anyway, so when you do this reverse enumeration you end up with gigantic tables of all these board positions and then who will win and therefore what is the best move to make. But Donald Michie was interested in saying "well, if we applied machine learning to it, could we boil this huge table of board positions down into some simple rule that a person could learn and memorize?" And so, he had built some of these tables and he was teaching a class at Stanford where the students were trying to apply existing machine learning algorithms to this problem. And one of the algorithms he was trying to apply was this software that I had inherited and was improving at Illinois called Induce 2 and they were having troubles getting it to work. It was written in Pascal, I think, and they just couldn't get it to compile, or I don't know.

And so, Michie called up my advisor and said "I'd like to fly Tom out to make it work," because this is how Donald Michie does things. When a problem comes up, he solves it. And so, he flew me out to Stanford and I worked with his TA and we got it running, but he also introduced me then to the faculty member who was ultimately to be my advisor at Stanford, Bruce Buchanan, and also gave me a lot of advice. I mean, he actually had been trying to recruit me to come work with him in Edinburgh, but I was more interested—and serve with his company and stuff—but at that point I decided to do a PhD.

[0:45:10]

I talked it over with my grandmother one day and she said "oh, go for the doctorate!" She had absolutely no doubt. And because I had also fallen in love with the ideas. So, very soon after starting graduate school, I got diverted off of the "go get an MBA and be successful in industry" and into the "oh, these ideas are fascinating, I need to learn more." And I always tell my students, "if you're getting a PhD it's because you like ideas more than money." And that was certainly true for me.

So, looping back then, the faculty at Illinois had recommended me for a summer internship at Bell Labs in Naperville, Illinois, which they did even knowing that I was leaving. I mean, when they told me they had nominated me for this, I said

"well, you know I'm leaving, I'm not coming back for the PhD" and they said "that's okay, Bell Labs knows that if they want to get good people, they ask us. And you will be a good representative for us." So, I went ahead and did it.

And so, I was working for the summer at a place that was really just a few miles from where my parents lived, and so it was kind of coincidently very handy. And that was fun because the Unix system, which had been written at Bell Labs, was just a kind of emerging academic thing at that time, and one of the advantages of working at Bell Labs was I could read the source code. So, this is just like being a witness to power; I could see how it all worked, every line of it, all four thousand lines. It was a very small program. And so, that wasn't my summer job, but after hours I was poring over the code just to understand how it all worked. But I worked on some fairly tedious kind of side projects that they needed doing.

In those days—I'm totally out of touch now with how digital switching works, but in those days they had a whole bunch of different switching systems that they sold to the regional phone companies that had names like number two, number three, number four, number five ESS, for Electronic Switching System. And so, some of them were designed for smaller, say rural applications, and basically they needed to do software updates on these things. They called it microcode distribution I think, or something. But it was basically just a software update. And the software update, since there wasn't the internet, would involve making a phone call over a modem to the device and then uploading or downloading the code into the device. And I wrote programs to make those phone calls and make sure that transmission was successful and retry it if it wasn't and that kind of thing, so very low-level stuff. So yeah, that's all I did while I was there.

But one thing I was really impressed with at Bell Labs was that although I was only there three months, I think I was reorganized three times, because they were constantly changing how the groups were organized and I was in a group that was something like Software Development Systems Group, which really worked on software tools for other programmers to work on. And that was eye-opening too, because I'd never really seen industrial software engineering and all the tooling that that requires, and testing and everything. So, I had a very experienced guy who was my mentor and officemate, and so that was fun to see how all that worked.

CP: Well, you went on to Stanford and it sounds like it was a good environment for you.

TD: Oh yeah, well I think Stanford was at that time, and probably still is, the best department in the country in computer science. Not the biggest, by any means, but they had lots of excellent faculty, and therefore they attracted lots of terrific students, and so it was a great environment in terms of just the other students there. So, I was joining—Stanford was also one of the big centers for artificial intelligence. There was a Stanford AI lab, which was run by John McCarthy, who was one of the, probably is responsible for the name of the field artificial intelligence. I was in a more applied AI research group that was led by Ed Feigenbaum, called the Heuristic Programming Project in those days, later known as the Knowledge Systems Laboratory, and then there was a third group that did vision and robotics that was led by Tom Binford.

[0:50:11]

And so, I don't know, the idea of working on machine learning actually was very fringe in those days. Now it's probably the number one topic in artificial intelligence, but in those days it was a very fringe topic, because the main focus was on, in those days—this was the days of what were called expert systems, and it was really the realization that, unlike in something like chess where there's a small set of rules that you can write down and then it's purely a problem of computation or reasoning to play the game, in the real world, the real world is a very messy place and there isn't a simple set of rules. For instance, it's not like a physician can learn twelve axioms of group theory or something and then they diagnose all disease. No, there are thousands and thousands of things they need to know to do disease diagnosis.

And so, Ed Feigenbaum was interested in building computer systems that could do things like disease diagnosis, and this really meant interviewing expert physicians and somehow capturing their knowledge in a form that a computer could compute with. And so, this was called knowledge engineering and knowledge capture and knowledge acquisition. And so, there was no notion that the computer was autonomously learning anything. It was being programmed, but maybe in a much higher-level programming language, where you could write down kinds of disease diagnosis rules as rules.

So, I was kind of lost, actually, I think, in that. My PhD thesis was a great training experience, but I never did anything on it afterwards, because it was really, it didn't really—it was a dead end. It didn't really go anywhere.

CP: How did you find living on the west coast?

TD: Oh, I loved the west coast. I was astonished by the west coast, actually, because I had lived, basically had been moving back and forth along the 40th parallel, and based on my life experience, the weather was the same everywhere. It was hot and sticky in the summer and it was cold and freezing in the winter time. And the idea that in the west, that you could travel just a couple miles and have a completely different climate experience, that totally blew my mind.

The other thing that really I was astonished by was the idea that if you make a wrong turn, it could be a long time before you get a chance to recover from that. You know, when you grow up in Iowa, if Iowa was designed by René Descartes, essentially, it's a giant piece of graph paper. There's a road every mile, north and south, and so if you miss a turn, you just turn on the next one, just one mile away. But even here in Oregon, it still cracks me up that there'll be a sign that says "Ocean Beaches" and that's the one chance to get to the ocean, and if you miss that, it's another hour before you have another chance. So, the west is weird. But yeah, so I fell in love with the west.

CP: Well, you finished up at Stanford in '84.

TD: Well yeah, very end of '84, right.

CP: And then you came to OSU in '85.

TD: Right.

CP: Was there something in between?

TD: No, no. Straight from one to the other.

CP: Well, take me through how did you –

TD: Well, so I fell in love. While I was at Stanford, I was singing in the Stanford University Chorus and I met my future wife Carol Rivin there. She was an alto, I'm a tenor. And she was doing a postdoc in molecular biology, basically plant genetics in a group at Stanford. And so, that was just her postdoc and so she was going into the market, and so she interviewed for jobs and did a bunch of interviews and ended up accepting a position here at OSU in Botany and plant pathology, and I wasn't done. And given the pleasant aspect of the quality of life and the fun I was having as a graduate student at Stanford, who knows if I would have ever gotten around to graduating if I hadn't had the impetus that Carol had moved away. So, I tried to finish up as quickly as I could, which took nine months more, because she started in April of '84 and I didn't finish my degree until December of '84.

So, I was a trailing spouse. I had to find a job up here. I was really fortunate that right at that moment there were lots of jobs. So, OSU and University of Oregon and the Oregon Graduate Institute were all advertising faculty positions. And so, I visited all of them and I got job offers from U of O and OSU, and so I chose OSU and now I'm just completing thirty years of service here, so it's been a great experience.

[0:55:37]

CP: What do you remember about some of your initial impressions of the university or the town?

TD: Well, of course coming from the Bay Area, Carol and I thought this place was so dead and quiet. I mean, in 1985 there were not a whole lot of restaurants worth going to, and certainly there wasn't any good Chinese food at all, which in the Bay Area was sort of your default meal. So, it was very quiet. But we got married in April of '85, we had children in June of '87 and suddenly we discovered that we were in Paradise; that Corvallis was a wonderful place to raise a family. And the town sort of embraces you. There's so many services; the library, there's really nice daycare options, the parks, so it was just wonderful. So yeah, so then our perspective completely changed.

CP: How about the state of the department at that time?

TD: Well, the university was kind of in a mess, and Computer Science was in a terrible mess. When I arrived, Computer Science did not have a recurring budget, so every year they had to argue for their budget to get it. And when I arrived, the year I arrived, the department lost, I believe, five faculty, because there was this boom happening in industry and most of those faculty left and went to join a computer science research lab that Tektronix was opening. And I remember my first week, one of my colleagues from electrical engineering said "so, how long are you going to stay?"

And frankly, at that time I was planning to stay maybe three years. I was hoping to move on to some other place, because I was more ambitious than—I mean, OSU is still very poorly ranked and at that time the College of Engineering, well we weren't in the College of Engineering, because Computer Science had been a bud off of Mathematics, which was a common way that Computer Science departments got started. And so, we were in the College of Science, we were teaching a huge amount of service courses, which was good, I think, and has always been important, but we had no control over how many majors we had. And so, Computer Science goes through these boom and bust cycles and when we would have these booms, we would just have overwhelming numbers of students.

That said, I mean the department was very nice to me and really protected me from that, so I was never given a teaching assignment with these truly immense classes, but I was teaching two classes my first quarter and also wrote two grant proposals. So, I really worked hard the first quarter, so I got off to a pretty good start and the dean actually gave me a little reward with some additional money, saying "this is wonderful, keep up the good work," for my research. But yeah, very few of the faculty were engaged in research at that time, at least at a significant level. So, there was hardly anybody who had any funding, government funding.

So, things of course have transformed over that time. There was a change in department head to Walter Rudd, and as part of his dowry to become department chair, he did get a reoccurring budget, and we also were able to hire some more faculty, and we continued to hire better and better people, partly, I think, because I was really in the wave of the first people who had gotten PhDs in computer science, as opposed to having been sort of retrained from having been mathematicians or physicists or electrical engineers. And so, there was kind of this fanning out from the original, the key institutions like MIT and Carnegie Mellon and Stanford or Berkeley, to start populating all these other departments.

[1:00:12]

So once I was here, then we hired a guy from Carnegie Mellon, a guy from Berkeley and really started to build up a more professional, ambitious faculty. And then also I think there was a steady improvement in management. So, Walter Rudd was a big step forward, and then I think Mike Quinn took over from him, and then after that we decided to merge with electrical engineering and we had Terri Fiez, who is just an incredible department head, and I guess we are now getting a new head who just said yes yesterday or the day before or something, who also looks very good. And similarly, at the level of deans and the support for the institution from the institution, that has also improved monotonically over time.

So in particular, I don't remember who was president at the time, I'm very bad with these names, but there was a provost who was later president of Penn State and got into trouble with a—

CP: Spanier.

TD: Yeah, Graham Spanier. He did a terrific job for Computer Science. One of the things he did was he had assigned a sort of specialist to come interview the faculty and talk about where is this department going in the future, and he decided it really made sense to move us into the College of Engineering. And there was—the faculty were not totally behind this, I think. They were very worried because coming out of Mathematics, we view ourselves—the field of computer science, for a long time, in trying to prove to the world that we aren't just programmers; that we work on fundamental questions, and actually this has encouraged the field to become extremely mathematical in its approach to problems. We want to not only come up with a solution; we want to prove that it is the best possible solution, or prove that it is at least within some margin the best possible solution. And there was a sense that we were going to mix with those engineering types who are just much more industry-oriented and have sort of a different culture.

But I think the fact of the matter is that we are training software engineers in the hope that they will build reliable systems that work, as opposed to crashing all the time. And I think we can't say that we've succeeded as a field in that, but we are, we do engineering, that is our primary thing. Of course, like other fields of engineering, we also do want to understand the fundamentals, and so we didn't have to give up our mathematical roots by moving into engineering. Certainly Berkeley and MIT have been ECS departments for ages and they don't seem to have had a problem with this. So, anyway, it's worked out really well.

So, I think one of the things I find frustrating is that there's this mania for ranking departments, and the ranking of Oregon State has not really changed that much in the whole time I've been here, and yet the quality of our education and our research has just vastly improved. We're probably a thousand times better than we were. We produce really well-trained undergraduate engineers who get positions at top companies, and our PhD graduates also go to the top places, but the thing is, all the departments have gotten much better, and this is not reflected at all in rankings. So, I would like to see academic institutions scored on some absolute measures of quality, as well as relative measures, because there can only be twenty-five top twenty-five programs, but really as a country we should want that all two hundred of our programs are at that level of quality. And I think we're not there but certainly there are many, many more good places now, because virtually all of the major private land grant universities are now staffed by people who have had excellent training themselves.

So, it makes it a lot tougher to get grants, because everybody's doing such a good job and the pot of money hasn't grown as much as the number of people. But that's a complaint about success.

[1:05:10]

CP: Was there much in the way of a technical infrastructure awaiting you when you arrived, or was that part of the grant process for you?

TD: Well, so the department gave me I think something like a hundred thousand dollars start-up, and so I bought the first laser printer for the department and then I bought a bunch of specialized computers that were—in those days there were computers that were designed to execute just one programming language, the Lisp programming language, which was at that time the dominant language for artificial intelligence programming. So, I think I had, at some point, four or five Lisp machines connected up in a little Ethernet.

I mean, another thing to realize was a big limiting factor in having a good Computer Science department was being on the ARPANET, and so there was like this club of the top schools that were on the ARPANET, and that was only maybe a dozen. UCLA and USC were involved because they helped design the network, and MIT, Berkeley, Stanford and so on. And if you weren't on the ARPANET, you just were not in—you were left out. But right about the time I moved here, the NSF decided to make a major investment in something they called NSFNET, to network all the Computer Science departments in the country. And for a long time, NSFNET was still essentially like a dial-up connection, and we still had —and was mostly for email. You could do some file transfers.

We also had a dial-up connection to HP through something that was called UUCP, Unix to Unix Copy, I think it was. And so, if you had an email message that was urgent to get out, you could launch that and it would call to HP, which would bounce the message through three places at HP and eventually out to some other university. But then, I think, I don't know when really, I mean the internet was just being rolled out right at that time. Essentially, the internet protocols, I think, were designed in '79 or something like this and was first rolled out on top of the ARPANET and then other networks were connected to it. And so, once OSU was actually networked, suddenly that really leveled the playing field across the country for departments like ours to be able to get access to technical reports and software and all the kinds of things that now we'd assume are available with a voice query.

CP: So, email was part of your life at this point in the mid-eighties?

TD: Oh yeah. Well, because I had been using email right as a graduate student. It was already an everyday tool. Yeah, I remember as a graduate student, once you were like third or fourth year, the department would let you take a terminal and a modem home. So, I had a 300 Baud modem in my apartment or something and a computer there so I could dial in to

campus. And one of the early signs of internet addiction was if you were eating breakfast while reading your email. This would be 1980 or '81 or something.

And so, I remember a very common topic at parties was "email is so amazing, how come this hasn't swept the whole country? Because this is such a great technology." And it really took twenty years, which is the typical kind of delay, I guess, with most new technologies between the time it's sort of fully functioning in a laboratory setting and the time it sweeps the society. I mean, I think the big thing that all of academic computer science completely missed was, to us computers existed to compute, to take inputs and produce an output like an answer to a problem. We didn't think of computers as a medium that would be a communication mechanism between people. But that's actually what computers are used for today; that they are just like these switches of this digital medium that connects everybody, people to people.

And so, once web browsers came up in '93 and email and the World Wide Web, then there was just this amazing explosion of things that was a complete shock and surprise to us, because we thought well, people want to use email like we use email, but the idea of bulletin boards and mailing lists and all this stuff, it was pretty weird.

[1:10:09]

That said, there were several designs for email and for mailing lists that were much better than what we've ended up with, but unfortunately died along the way, and some mistakes probably can never be undone. Xerox had a much better design for email, I think, than we have now. But oh well. They also had a better design for networking that had more internet addresses, because they were using 40 bits instead of 32 bits I think, or something like that. And now that we've run out of addresses, they've been vindicated. But nobody uses XNS, I guess, was their network protocol that they were proposing.

CP: Well, in preparing for this interview, it seems to me that there are sort of three kind of large themes to the last thirty years at OSU; sort of the more academic research that you've done as a faculty member here, you're very heavy involvement in developing the field of machine learning as a field, and then private sector work as well. I'm wondering if we can kind of, as best as we can in the time that we have—and I'm trying to do some major justice to thirty years of work—touch on all three of these themes a little bit.

TD: Okay. So, in the academic work, I guess the...how to organize this...maybe I should just talk about a couple of landmarks.

CP: Sure.

TD: One of the ways that the field of machine learning has developed is by defining what we call settings. So, the simplest machine learning setting is the one I was saying before, where I give you a vector of variables that describe some objects or things—these trains going east and west—and then a label or output that says east or west. And the goal of the learning algorithm is to learn to look at the variables and predict whether you're going east and west, or we can think of that and mathematically that is a function that maps from these inputs to this output. And so, that is the simplest setting for machine learning. It's called the supervised learning or supervised classification setting. But then, I think you can view some of what's happened in my career as discovering new settings and then developing an algorithm, maybe like the first algorithm that tries to solve a problem in that setting.

And the first one that came up, well there were two that I sort of explored simultaneously. One was when the input—and really I worked on this on my master's thesis—the input is a sequence of objects in which the ordering might be important and the output is a sequence of labels, say. So, an important early problem of this type was mapping from a word spelled in English, so a sequence of letters, mapping that to a sequence of phonemes that could drive a speech synthesis device. So, basically trying to teach a computer to pronounce English from its written form, which is of course quite challenging because our spelling is such a nightmare.

And this problem had been explored by Terry Sejnowski, who is one of the leaders in neural network algorithms for machine learning, and he had built a system called NETtalk. Digital Equipment Corporation, in those days, sold a speech synthesis box that was mainly designed for applications for the blind. So, you could build a machine that could read out loud. And Kurzweil is famous, I don't know if you—he's sort of an industrialist of artificial intelligence—for building something called the Kurzweil Reading Machine that combined optical character recognition—well scanning, OCR and

then text-to-speech to read out loud to the blind. So anyway, in the past people had built these systems by sort of manually programming them, and one of the key ideas of machine learning is that instead of trying to manually program a computer to do these things by writing out a whole bunch of rules for how to pronounce English, for example, we could just start with a dictionary of words, say a thousand words where we had the spelling and the corresponding pronunciation and then just try to use machinery to learn a function that maps from the input strain to the output.

[1:15:08]

And the thing is, instead of just trains going east and west where you only have two possible outputs, here you have combinatorally many outputs if there are something like, I can't remember, sixty phonemes in English, or fifty-four, I think, and every word has, say, five of them, then that's fifty-four to the fifth power possible pronunciations. And we would say that's a very large output space. So, this is now known as structured prediction because the input has a structure as a sequence of items and the output has a structure as a sequence of items, and we need a map between them.

And so, I had a student working on this problem and we developed a solution for it that was based on something called error correcting codes. So, Bella Bose, who's my colleague and currently the interim head of EECS, is an expert in what's called error control coding, which is designed for adding redundancy to a message that you're going to send over a very noisy channel, like in the old days with modems or radio, in general, where errors are going to happen when things are coded. Or even when they write music onto a CD, there are errors in the writing, but they write it in a way that's redundant so that as long as the error rate is small enough, you can reconstruct the original signal perfectly. So, in talking with Bella, I thought well you know, a machine learning algorithm is a bit like that, because given the inputs, it only gets the output right with some probability.

So, we thought of a way that when we have these problems that involve particularly many, many things, we could actually encode each output as a string of bits and we could try to predict each bit separately, and there would be some error probability of predicting each bit, but we would use a code that was an error correcting code and we could then recover from those prediction errors. And this is probably one of my first really exciting papers that we had a lot of fun with. So, that's one setting.

Then, on my first sabbatical, Carol and I went to Berkeley and I got involved half-time, working for a start-up pharmaceutical company called Arris Pharmaceutical. And they had a problem, their goal was to—very often you don't fully understand the biology of—you're trying to design a drug to do something. For instance, they were very interested in trying to build an artificial erythropoietin. So, if you're undergoing kidney dialysis, a lot of your red blood cells get damaged, and there's a hormone, EPO, which we now know mostly in the press because it's used for doping in sports, but it was originally designed as a drug that kidney patients could take to speed up the development of red blood cells, but it wasn't known how it worked. They had isolated the protein in nature, and it's a big protein, and the question was maybe there could be a small molecule, because most successful drug molecules are not proteins. If you take a protein orally, your stomach just digests it and it doesn't work. This is why you have to inject insulin, you can't take it orally because insulin is a big protein.

So, and at that time, Genentech had, or was very soon to announce, that they had cloned EPO and were able to produce it, but people would have to inject it. And so, one of the ideas of Arris was we were going to make a small molecule EPO mimic. And the basic idea of the way they were going to do it was to create a large number of small molecules that had various shapes and then test them to see whether they—you can sort of think of it as the lock and the key. If you think of the thing that EPO binds to that turns on the biological process in the cell, can we think of that as like a lock and we're trying to figure out what shape the key is? And the troubles with big protein, we didn't know the shape of the EPO protein. We didn't really know how it works. Now we know it actually grabs two of the receptor molecules and brings them together, which probably could never be done by a small molecule.

But we didn't know that at the time, and so we were building these so-called libraries of small molecules and then they would be tested to see how well they bound and then the ones that bound well we would use machine learning to look at the binding and to look at the shape of the molecule and how well it bound and try to infer what is the shape that will bind best. And this is sometimes known as structure activity relationships or 3D qualitative structure-activity relationships.

[1:20:13]

And the interesting problem is that you do not know the small molecules. You don't actually know their shape either. They can adopt one of a small number of different confirmations, it's called, as the bonds rotate. And so, when they bind, one of those configurations is presumably the one that's binding successfully. So, what we really had was we had say four possible shapes for this molecule and it binds well and twelve possible shapes for this molecule, it binds well, four for this molecule, it doesn't bind well. And so the question was, for the ones that are binding well, one of those shapes must fit. For the ones that aren't binding, none of their shapes fit. And so, this was known as the—we called this the multiple instance problem, that instead of having one input to one output, we'd have multiple inputs to one output. And we were trying to infer which one in this bag of things is the one that's actually working.

I mean, the analogy I was given was, suppose I give you a keychain that's got a bunch of keys on it and I tell you one of those keys opens this lock, I give you another keychain, it has a bunch of keys, I say one of these also opens this lock, I give you another keychain and I say none of these fit. Can you, just by examining the keys, try to figure out which one I'm talking about? You're not allowed to try them individually, because in the chemical case we just give that whole keychain to the thing and one of them fits, and we can measure that one of them fits, we just don't know which one it was.

So, we wrote a paper. After I got back from that sabbatical, I continued working for them for a couple of years, basically trying to finish up a paper on this problem and come up with an algorithm that actually worked. We eventually did develop a nice algorithm called Compass and we got a patent for that, but I published a paper with a couple of co-authors from the company. That was my second big paper. So, that's an example of another kind of setting.

And then, I guess my third big technical contribution would be a paper that I published in 2000 on something called hierarchical reinforcement learning. So, reinforcement learning is you had another setting. In reinforcement learning, the idea is more that I'm going to take a sequence of actions to try to control something, like to try to—if we think about it as maybe I'm—what's a good example...Actually, a good example maybe right now is I have two problems that I'm working on right now that are related. So, if we think about we have some landscapes that are being invaded by an invasive species and we'd like to somehow try to control that invasion, maybe even wipe it out, and if we could—and so we have certain actions we could do. So, we've been studying an organism called the tamarix, which is a tree that's invading sort of river systems in the west. It's a big problem in New Mexico and eastern Colorado right now. And so, it's traveling through these river systems and it makes lots of seeds; like most invasive species. It's a very successful organism.

But the trouble is, we don't have a budget big enough to, say, in one year go out and just kill every tamarix tree in North America. Instead, we have small budgets and the question is well, each year we could go out and try to kill some of these trees and we could also try to plant native species as competition against these, and the question is where in the landscape should we do it? It's sort of like we're playing a game against nature where every year we can observe where the plants are, we get our budget and then we have to decide how to spend it in terms of performing actions. After we perform our actions, then nature gets to make its move in terms of the trees reproduce, some trees die, some trees spread, new trees sprout, and then it's another year and we get to take another action.

And so, the question is, over the long term, can we eventually, by taking actions strategically, even if they don't look like they're achieving the goal in the short term, in the long term, can they achieve the goal? And so, these problems are sometimes called stochastic dynamic optimization problems. They're stochastic because we think that the success of our actions and the behavior of nature is somewhat unpredictable. We can go and try to—and we do a kill this tree action, but maybe we only succeed seventy percent of the time, and thirty percent of the time the tree recovers from it. Or we plant a native tree but maybe it only survives eighty percent of the time. And we can't predict exactly where the seeds that get dumped in the river, where they will go. So, that's also stochastic.

[1:25:16]

And so these, in machine learning we call them reinforcement learning problems, which betrays a history back to animal psychology studies where you provide rewards for the animals running mazes and things like this. But the main idea is that we get delayed reinforcement. So, we take actions now whose benefit might not be observed for many years. And particularly in these kind of policy making things, convincing people to spend a lot of money now for a benefit that may not happen for a while is challenging. You really need to show, with high confidence, that it will succeed.

So, I was very interested in this general kind of program. I got interested in the late nineties, and one question was can we somehow break these big problems down hierarchically into small problems, just the same way we write computer programs by breaking them into subroutines and sub-subroutines and so on. And so, I developed, in a very clean mathematical setting that couldn't possibly work with tamarix, but a theory of hierarchical reinforcement learning that showed that it was possible, and demonstrate some interesting interactions between your ability to introduce extractions where you could focus just on a subset of the variables inside the subroutines and still get an optimal solution, overall. So, that was probably my third big sort of hardcore academic contribution.

CP: It's been interesting to learn about the sort ecological applications of a lot of your work, and the university actually, I think early in the mid-2000s, developed this set of six strategic initiatives it was going to fund. One of them was ecosystem informatics. I gather you were involved with that?

TD: Yeah, so right about the time I published this paper I had maybe a mid-career crisis in some sense, because I felt that I had established that I could do academic research at the highest levels and make contributions there, publish papers, get lots of citations, but I didn't feel like I was having an impact on important problems facing society. And so, I came back from my sabbatical in '99 determined to kind of reorient my research program. I had already been collaborating a little bit with ecologists on campus, because they're so thick around here that if you just choose a collaborator at random, you're likely to run into one.

But I decided that since OSU is really a world leader in ecology, and arguably number one or number two in the US, and certainly just the top place for this, that one of the pieces of advice one of my advisors had always given was "collaborate with the smartest people you can find on campus." So, I started doing more social networking and meeting people and I was very fortunate to—we were interested in trying to get funding for a graduate training program. NSF has these IGERT programs where they will give you, say, twenty PhD fellowships for a targeted area. The program has changed a bit now, but—and so, a group of us in Computer Science were brainstorming about what that might be. We found out that there was a group in geosciences led by Julia Jones who was also trying to think of something, and so we joined forces.

And Julia is a very centrally networked person on campus, in terms of knowing a lot of the people in the Andrews LTER and a lot of the data, and so kind of drew us in and introduced us to all these people. And so we, working with Ed Waymire and Enrique Thomann, and I'm blanking on some other names, Mark Harmon in Forestry, Julia was our initial P.I. team, and it took us two tries, but we got that funded. And then part of our program was then brought in; I think we had something like eight or a dozen students, PhD students that recruited into that, and we taught a year-long core course in a team-taught setting, teaching these students who were from mathematics, Computer Science and various parts of ecology about each other's fields, which also meant that there were typically three faculty members in the room as well; me, Julia and Mark or something like this. And so, the faculty were learning from each other as well.

[1:30:19]

Now, I think this is, from an institutional point of view, this does not look like it's a good idea. You have three full professors teaching, say, twelve students. Why is this a good investment of institutional funds? Well, because it then spawned all kinds of subsequent research. So, speaking of delayed rewards, it really was an education for us on the math and computing side, about what were the important questions that ecologists were interested in, and what are OSU's unique advantages in working on those, because of our field study sites and the expertise we have.

And so then, soon thereafter, Desiree Tullos in Biological and Ecological Engineering decided to P.I. a summer undergraduate program in ecosystem informatics, and so I don't know that I was a co-P.I. in that, but I was a participating instructor in that. And so, that's still going and we just on Monday. We have this year's group of, I think we have ten students joining us for the summer. And that's a lot of fun too. All funded from NSF and coming from schools all over the country, including OSU of course. And again, we're still bringing together computer scientists, mathematicians, now statisticians, engineers and biologists.

And part of the reasoning behind that, I think, is that we don't think that computer scientists need to also become outstanding ecologists and vice versa, but we think that there's important research to be done, and in these cross-disciplinary areas, and people need to understand each other's vocabulary, conceptual frameworks, problem definitions, what counts as a research contribution, how would you have a real impact in the world, all these questions. And so, that's

kind of the theory behind this, is that you have our original idea, was maybe we'd have pairs of PhD students that would collaborate and help each other out. In the class they did class projects that involved those kinds of collaborations. A couple of those worked, but mostly students just became sophisticated about these other fields and then could integrate that work into their dissertation work.

So yeah, so then after that then we made a proposal to the provost's initiative for some new faculty slots in this area to hire one person in Computer Science, two in Math and one in Forest Ecology, and those have been really great hires, too. And then that, in turn, led to me being invited to participate in a ten million dollar grant from NSF on computational sustainability, we called it, that was joint with Cornell, which is probably the other top school in ecology in the US, and there's an awful lot of flow back and forth. It'll be interesting to know what fraction of the faculty in ecology here spent at least some time at Cornell.

But so, the NSF has this program in computer science called Expeditions in Computer Science, whose goal is to encourage the CS discipline to branch out in new ways, for instance in an interdisciplinary project or to take on some sort of high-risk, high-payoff kinds of things. And so, NSF was very enthusiastic about a proposal that we did jointly with Cornell, and so that, in turn, brought more millions of dollars to campus to fund graduate students and postdocs. And then, following out that I've gotten two more grants, so this is all really turned into one of the things I'm passionate about, is working in this area.

CP: Yeah. And the Cornell project was the birds project, is that right?

TD: Well, the first work with Cornell actually funded work on birds, so because the Cornell lab of ornithology, we had already been doing some work, because when Matt Betts was hired as one of the provost hires, he does ornithology as well as ecology. And he was very interested in whether we could use microphones in the fields to identify bird species, or even to identify individual birds and sort of track them during the whole reproductive season. And so, we were starting to get recordings of data from that. "We" in this case is not me, but my colleagues, particularly Xiaoli Fern and Raviv Raich, because Xiaoli was another one of the faculty who was hired under this. And so, they started working on that and we were thinking boy, you know it would be really good if we could collaborate with the Cornell lab of ornithology, because they already have some recordings like this and they are the best place in the world for ornithology.

[1:35:28]

So, one of the advantages of being invited to join this thing with Cornell was we got to collaborate with the Cornell lab of ornithology. And one of their big projects is this citizen science project called eBird where bird enthusiasts, bird watchers, can collect data for eBird, uploading their checklists to eBird.org, now with an iPhone app and Android, I believe. And so, one of the things we tried to do with that data was to start working on species distribution models and migration models and then yes, as the five year expedition grant was ending, we got a three-year grant to work on building a content scale bird migration model.

But some of the other things that were funded by the expedition grant were beginning to work on tamarix and beginning to work on wildfire management in eastern Oregon, which is another one of these sequential decision problems. And then I also got a subsequent grant to continue that work, which is currently operating. So yes, many things came out of that.

CP: We talked a bit about your earlier private sector involvement with the pharmaceutical company. I want to ask you a bit about a couple Corvallis companies. One was MyStrands, which is a company we heard a lot about, for a little while, at least.

TD: Right, and now it's Strands and it's still privately held.

CP: And then BigML is what you're involved in now?

TD: And BigML is the other one, and it's also still privately held. And then I was involved also in Smart Desktop, which we put in Seattle. So, the story with Strands and with BigML; both grew out of my sabbatical in Spain. When I was there, I met a PhD student named Francisco Martin who was an amazingly bright and busy guy, and while I was there he actually dropped out of the PhD program to start a company called iSOCO, for Intelligent Software Components, I believe, which was an early web services and software development company and was very successful in Spain. So,

he did that for a few years. They had invited me to participate but I had my head down. I was working on hierarchical reinforcement learning. But then after that company, he finished his PhD and then he decided to come to the United States to do a postdoc with me. So, he contacted me and said "would this be possible?" He was actually able to fund himself because he'd been successful with his company.

So, he came and we started working on some problems that had to do with kind of improvements to the internet. These are sort of early ideas that we were collaborating with some folks at MIT and USC. But after only about six months here, he and another colleague here, Jon Herlocker, who is very interested in recommender systems, decided they want to create a company to do music recommendations. The iTunes had just come out, Starbucks was just starting to sell music in their stores and the iPod was out, and so the idea that there would be this kind of big data produced by people using their iPods, which you could see what they actually listened to and we could use their actual behavior to make recommendations to them, instead of just asking them to rate things like the way it's done with movies. And so, they asked me to participate in this, so I was not like the—I did not provide the ideas for this exactly, although when it came time then to actually design algorithms to do the recommendations, I made some contributions there. And so, I would typically go down and visit maybe once every other week for a couple of hours and talk to the engineers and talk with Francisco.

And so, then he was running Strands for several years and it morphed into a couple of other things. It was doing some stuff with sports and really, in some sense, it was ahead of its time, both in music recommendations, because music recommendations—eventually Apple bought the patents that Strands had, and of course now they're providing more and more recommendations, including more they just announced with their new service. And of course with Spotify there's recommendations, too.

[1:40:18]

And then when—but then after Strands, when Francisco moved on from Strands, he wanted to work on BigML, but I—anyway, two other ways Strands was ahead of its time; it was clear that there was going to be sort of wearable computing, like smart shoes or, well we didn't anticipate Fitbit, but things like phones that could track where you did your run and stuff like this. And so, Strands got into that, too. It wasn't really related to recommendation. It was just that there were a lot of runners in the company and they were excited about it. And then they also got into financial products and recommendations for financial products. And now that's probably their main business line. So, they have several big customers and they're doing quite well there, as far as I know.

But I think in all those cases they were really ahead of things, and maybe a little too early. But with BigML I think we're right on time, because with this huge ability to get all kinds of data from point of sale terminals, from the web, companies want to become more data-driven; essentially use more of the scientific method to make decisions about how they should be doing things. And of course big companies like Verizon or something, they can afford to have a big staff of statisticians, who are now called data scientists, but who can apply machine learning and statistical methods to help them make decisions.

And so, BigML hopes to help them, but I think it looks like—you know, whenever you start a company, you're not exactly sure who your customer was going to end up being, but I think it looks like there are millions of small companies that can't afford to have any real staff statisticians, but can we make it possible for, say, a sole proprietor or a small restaurant or something to be able to take their spreadsheet and drop it into our website and get out useful things like predicting how many customers they'll have each night of the week as a function of the weather and sports events and whatever seems to be relevant?

So, that's the kind of thing we built. And we built a tool that actually can work for massive data sets, but I think we're finding that most of the business is actually with much more modest-sized things. So, a lot of the focus has been on making it super easy to use, completely intuitive, very nice visualizations and interaction and things like this. And the real brains behind it on the technical side is a guy named Adam Ashenfelter who did a master's thesis with me, working actually on that structured inputs to structured outputs problem for his master's thesis. And so, we're really fortunate that he likes Corvallis and stuck around.

CP: Well, as we sort of wrap up here a little bit, I'd like to ask you a question that I'm guessing you probably have thought about a lot and you probably get asked about a lot too, and that's just sort of the future implications of machine learning.

There's an anxiety there for a lot of people about artificial intelligence, and we're now losing to computers on *Jeopardy*, for instance, and there's this fear that perhaps we might lose control of the technology at some point. What are your thoughts on that?

TD: Yeah, I think...well let's see, where to start. I guess the first thing is on the question of the advancing of the technology. I think there are really two kinds of problems that artificial intelligence faces. One are problems that are really like these mappings from inputs to outputs, so seeing big advances in the ability to recognize faces, recognize objects, recognize handwriting. I mean Google, now you can take a picture of a sign and it will find the text in the sign and translate it into English for you. I mean, this is fantastic, right? Microsoft is very soon going to be releasing a Skype with real-time translation between languages, I guess at the sentence level. You'll speak a sentence and then it will speak it in Portuguese or whatever.

So, I think the thing to realize about these is that these tasks do not require deep reasoning; it's all about finding patterns in data, and one thing about finding patterns in data is that the central limit theorem applies, for a more technical audience, which is to say that the accuracy of the predictions of the system will improve as basically one over the square root of the number of data points you have. So, if you have a million times more data than you had before, your error rates vanish and you can go from where we had a thirty-percent word error rate ten years ago for speech down to maybe something like five percent word error rate right now. So, speech with your phone recognizing it has gone from being completely intolerable to being pretty reliable. It's just extraordinary how much. But I think that's because we have this one-over-root-n kind of behavior, where the exponential increases that we're getting from big data and from computing, we see an immediate benefit.

[1:46:03]

But the other kind of thing that the AI systems need to do is to actually reason about, say, the future. So, like with my tamarix problem or a wildfire, you need to reason, say a hundred years into the future. And the problem with that is just like the problem of playing chess, is that if you look at the current situation now, you have many choices of actions you might do. In chess they estimate roughly thirty or forty actions you could take at each step, and for each of those forty actions your opponent has forty responses and then you have forty responses to his forty responses. And so, you have this exponential growth and you cannot possibly, in the amount—well, so the point is that computers can search maybe six or seven moves ahead by applying lots of tricks of cleverness, and if computer power increases by a factor of ten, that goes from seven moves ahead to eight moves ahead. So, if you have an improvement of six orders of magnitude, that maybe gets us six moves deeper into the future, so we can look sixty years further ahead in planning our Tamarix stuff.

So, unlike when we have these—when we're looking for patterns in data where X measure growth leads us to very rapid improvement, when it comes to reasoning, exponential growth gives us very slow progress into the future. So, the vision that some people have that we're going to be—that computers will very rapidly become super intelligent, like vastly, essentially omnisciently intelligent compared to people, I think those are completely unrealistic, because I think it's physically impossible. That's not to say that if a computer can see eight moves ahead and we can only see seven moves ahead that it doesn't beat the pants off of us. After all, we're maybe not all that much smarter than chimps, but we're killing them off because we're just occupying habitat. So, computers do not have to be super intelligent or omniscient or anything. We don't have to have the technological singularity to have a problem with the technology. And nowadays if we build software and something goes wrong, we can Control-C, or we can reboot our phone, or whatever we have to do to stop it, but what if this computer is driving our car and something goes wrong? Are we going to have a kill switch? And if we are, then is it safe to use it?

So, and of course we've always—well, not always, but we have in the past put computers in charge of high risk things. I mean, computers flew the Saturn V rocket, and they certainly land aircraft now. And so, they do make life and death decisions, but generally that software has been extremely carefully tested, very carefully written by hand and tested. It has not been created by something like machine learning, which is only probabilistically correct. So, one of the, I think the temptation we see as we see with big data, that we can build computer vision systems now that probably can find the road and find all the obstacles and recognize when the dog runs in front of the car and all these things, it's tantalizingly close to being able to have a computer that could drive a car automatically, even in urban environments. I mean, it's one thing to do it on the freeway but another to do it around Corvallis. But we have to realize that software is being built using

machine learning, and we can't provide the same kinds of guarantees that we can provide with carefully hand-coded and tested software.

[1:50:05]

So, unlike, say, the standards we have for avionics software, I think we need to development techniques to provide similar levels of quality assurance and control before we put software in charge of these high-risk settings. And I am worried that in sort of the race to market between the different auto companies and the non-auto companies who are moving into it, will there be a tendency to put something out before it's ready? I mean, when our app crashes on our iPhone, it's not life-threatening. And so, we tolerate just horrible software quality in most webpages and so on, and there's this sort of agile development which is very rapidly getting software out the door and sort of get it out and then fix the bugs later. That's not going to work for these high-risk things.

So, of course there are software engineering teams, say at Boeing or here in Salem, I'm forgetting, they sell GPS devices, but avionics too. It's an Oregon company that does this, that knows how to build high-assurance software. But those people don't seem to be working for Google or Tesla. So, I think that we need to—the industry needs to think very carefully about this. So, one solution, perhaps, is to have very self—a monitoring on these systems, so there's a part of the system that's watching the other system and checking to see if anything unusual is happening, and if something looks suspiciously wrong, then falling back on some safety policy. So, if it was like driving a car, well now we're going to slow the car down and pull off the side of the road kind of thing. So, we have to decide, in all these dangerous situations, is there some safe way of getting out of them? Because if there isn't, then there's no safe way to hit Control-C or reboot, and we better think carefully about that.

So yes, I think we—I mean, anyone who's written a computer program has lost control of it and had to kill it, and so there is certainly the possibility that we can lose control of this technology, and we need to make sure that it will be safe, that we will have a way to stop it and that it will be safe to do so.

CP: Well Tom, I want to thank you very much for this, this has been very fun and interesting for me, and you've obviously got a lot going on, you're a busy guy and I appreciate you being so generous with your time and your perspective today. Thanks very much.

TD: Well, you're welcome. It's been a great time here, so thanks a lot.

[1:52:51]