STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Creating Software with Machine Learning: Challenges and Promise

DR. PETER NORVIG

Director of Research, Google Inc.

The President's Distinguished Lecture Series • Stevens Institute of Technology

April 24, 2017

Thank you for the gracious welcome. Hello, everybody. Up in the cheap seats there, "Hi!" We're going to talk about creating software, making stuff. And here are some typical makers of the earliest 20th century. They made stuff like this, but then in the late 20th century, we started to get a new class of workers and they made stuff like this that looks strange. That's what it looked like back then. I actually was alive in those days. Now it looks more like this, but it hasn't changed all that much. It's still hard work; it's just mental work rather than physical work. So what is the process?

Well, we start off with a programmer. They get an idea, and then they start to work through that idea and very carefully write down all the steps of how to do it. "Under this condition, do this," and so on, and then you have a program that can do something useful. And that's traditional software development. But in recent years — and this goes back to the beginning

A video of Dr. Peter Norvig's lecture, which includes his slide presentation, is available at **stevens.edu/lecture.**

of the computer industry but has really taken off in recent years — we get people like this, who've proposed a new way of making software. And this new way starts with a computer rather than with a human programmer, and it's the computer that's going to do most of the work. The computer is going to write the programs for us.

How does it do that? Well, we still have a person with an idea on the side, but their idea is, "Here is a problem I want to solve, and the way I'm going to solve it is I'm going to feed the computer a bunch of examples, a bunch of data, and I'm going to tell the computer a little bit about how to process these examples, and then the computer is going to write the program for me from looking at those examples." But normally, the computer doesn't write the program in the same form that a human did. Instead we have different forms, and they look more like this: Sort of a black box that you can't really understand very well, but at least I've shown the lid open a little bit, so that we can peek inside.

This is a program. So the computer is writing a program for another computer to run. We also call it a model. It's a model of the domain, a model of the examples that we see. Many of you are engineering students, so you are all familiar with this and some of you, way back in high school, you did this, you built these models. So you took a science class, you went out and made some measurements, and then, either using a ruler or using a calculator, you calculated the best fit to those data points — and that's a model.

"So that's what the shift is. Rather than having the programmer say how to do it step by step, we're now telling the machine, 'I can't tell you how to do it; I may not even know how to do it.'"

Once we have the model, we can formulate it mathematically, and then we can throw away the data points because the model is a complete summary of everything we knew from the data. And what can we do with that? Well, if we have a question, "What would the value be for this point? I've never seen this point before," then we can use a line or use a mathematical formula and we can get the answer. So, from some examples, we generalize to find the answer to an example we've never seen before. And that's really all there is to machine learning, is coming up with a model; and the general model we had in mind was it's going to be a line of some kind, gathering the data, fitting the data to say, "Here is the exact line that is the best fit for this data," and then now we've got a model and now we can apply it to new cases.

So all of machine learning is like that, and the complications are that the models are a lot more complicated than individual lines, the data is a lot more complicated than a couple of points on a graph, and the algorithms for how you compute what's the best model, given the data, are also more complicated.

So, this is what the field is. What's the promise for it? Well, there's two things. First, it allows us to go fast. Instead of having the programmer sit down, and days and weeks and months go by developing the program, you throw some data at the machine and you can instantly get a response. So you can do things much faster than you could ever do before. And, secondly, as General Patton knew, we can now change how we approach problems. Patton said, "Don't tell people how to do things. Tell them what to do and let them surprise you with the results."

So that's what the shift is. Rather than having the programmer say how to do it step by step, we're now telling the machine, "I can't tell you how to do it; I may not even know how to do it." So, say something like recognizing faces. I can recognize the faces of my friends and family, but I have no idea how I do it. It's in my subconscious. I couldn't, for the life of me, write down how to do it, but I do know how to train a computer to do that task by showing them examples. So we can do things that surprise us, things that we would never be able to do as a programmer, as well as just being able to move faster. So that's the promise.

Now I want to show you some examples of how this works. We showed you the graphing of the straight line, but let's do something more complex. So I'm going to look at the task that we call word-sense disambiguation. And here's a task: I give you a sentence like this and then I say, "What sense of [the word] bank is that? Is that bank one, the riverbank, or bank two, the money bank?" As a human, you can answer that pretty well. I want to teach a computer to do that, and the inputs that are going to be available are these dictionary definitions, exactly as they are, and a bunch of sample sentences that happen to contain the word. And from that, I want to build a program that I can now give a sentence it's never seen before, and it will say "bank one" or "bank two."

Now, I think about how can I solve that problem. I say, "This seems pretty complicated. When I solve that problem, I've gotta know the dictionary definition of all these words. I've gotta know how the syntax of the English language works, with noun phrases and verb phrases and all that stuff, and I have to know some facts about the world. I have to know what kinds of things are money banks, and what kinds of things are riverbanks." It seems like it would take me forever to do this, and, if I did, now I've only done a word-sense disambiguator for this word "bank." And now if I want to do all the other ambiguous words in English, that looks like a lot of work; and in every other language as well. I'm giving up. I can't do all that. I can't write all that stuff down — *but could I possibly train a computer to do all that?*

Let's think about how we could do that. I can't train the computer to know all these things about all the facts about what a bank is and how the English language works, so I'm going to come up with the simplest possible model I can think of, and see if that model happens to be good enough. So here's the model: I take these dictionary definitions, I get out my scissors, and I cut them up into individual words and put each of those words into a bag. I do the same thing with the other definition. So now I have two bags of words, and here are the words that they contain. And, by the way, there is a technical name for this model. It's technically called the "bag of words" model. And what this model says is, "In bag one contains all the stuff you need to know about sentences you could say with a riverbank, and bag two contains all the stuff you need to know about sentences you could say with a money bank." And this model claims that the way an English sentence is constructed is not by somebody thinking in their head about all the meanings, and dictionary definitions, and the noun phrases and the verb phrases. The way we construct a sentence is we reach into the bag and we pull out words, and we put them next to each other until we're tired of pulling out words.

Now, that sounds like a terrible model of how English works, right? It doesn't work like that. It's much more complicated. A statistician, George Box, said, "Well, it's not so bad. All models are wrong anyway, and some of them are useful, so don't worry so much that that model is a terrible model. Ask, is it useful?" So is it? Let's try. I've got these two models. These two bags are seeded only from the words I got from the dictionary definition. Now I'm going to take all those sentences. The sentences didn't say which was bank one or bank two, but they're still going to be useful to me. So even though the sentences didn't have the right answer, I can still use them to improve my model. How do I do that?

I take the first sentence and I look at each of the words, and some of the words don't appear in either bag, and some of them appear in one bag. So the one in blue, the loans, occurs in bag number two, and some of them occur in both bags. The word "the" occurs pretty evenly across both bags. So I do that, and then I do a little bit of counting up and say, "Which bag is that sentence most likely to have come from? What's the probability that if I pick randomly from bag one, I would create that sentence, versus if I picked randomly from bag two, and I would create that sentence?" And the answer is it's more likely to have come from bag two, because that's the bag that contains the word "loans."

So now, I've made a prediction. In this case, the prediction happened to be right, but more importantly, I'm now ready to upgrade my model. I take all those words and I throw them into bag two. Now, bag two is a better model of what it means to talk about a money bank. I keep on doing that for thousands or millions of words, and pretty soon I've got big bags — but they're more effective at this task of disambiguating words.

So you start getting bags with counts like this of lots and lots of words, and you realize that everything I need to know about machine learning, I got on *Sesame Street*. All it is, is counting. We're building this model by counting up how many times this word was relevant to sense one or sense two, and now I can apply it, and now I can take sentences I've never seen before and score them. Some of the times I'll get the right answer.

Occasionally, I'll get the wrong answer — that's okay. Let's see how well I do. So here's a graph. I've trained it on one million words of the sentences that have "bank" in it.

There are four bars here because there are four details of slightly different algorithms for how you count and how many points you get for having the word or not, and so on. We get 80 to 85 percent correct.

Now, in the old days, like ten years ago, if you were an academic, what you would do is, you would say, "Well, the best one of those got 85 percent, but I'm going to be very clever and I'm going to invent a new algorithm, a new way of scoring, and I'm going to try to get to 86 percent. If I do, then I can publish a paper about it. And people play that game, and that's why there's four of them there. Somebody did one, and three other people came along and said, "No, I can do better." But what we soon realized is arguing over that and trying to come up with a better way is less effective than just saying, "Well, that was one million words of data. I can go back out on the web, and I can easily get up to ten million words, and look — boy, that's a big difference. And, "Ten million words, that's not too much. I can get 100 million words. Or I can go up to a billion words."

"A statistician, George Box, said, 'Well, it's not so bad. All models are wrong anyway, and some of them are useful, so don't worry so much that that model is a terrible model. Ask, is it useful?'"

I'm going up and up and the differences between these algorithms, we used to think it was so significant whether you used algorithm one or algorithm two. Those differences are small, but the difference from having more data tends to be big. Now it looks like we're pretty much asymptoting out, so if I kept going, I'd probably get diminishing returns, and so the best I can do is probably somewhere around 95 or 97 percent correct — but that's pretty good, considering the only work I did was to go out and gather up words and throw them into bags.

And the phenomenon here of saying, "Here's an application where if you have a small amount of data, tens or thousands of words, you don't do very well, but if you have a lot, millions or billions, you do really well" — that's this notion of this data threshold where you pass over from doing poorly to doing well. And you could say that companies like Google are built on looking for curves that look like this and saying, "Can I find problems for which I can do much better if I have a lot of data?"

So that was one task. Now I want to move on to another task, or actually a whole field, with a collection of tasks, the field of computer vision. And why is that useful? Well, you can use it as one of the components of a self-driving car to figure out where the pedestrians and other cars are, so you can drive safely. You can do things like this: This is an application of Google Translate where you point your phone at a sign in another language, and it translates the sign and puts the text in the right font right into the image on your phone.

And, for me, as an amateur photographer, I spent a lot of time in the past organizing my photos, attaching keywords to them, and putting them into folders and reorganizing them when I felt I made a mistake, and then Google Photos came out with this ability to do that automatically. So I said, "Let's try it. I'll strip all the keywords off all my photos, I'll throw them all in and see what happens." This is where I get an advantage. You guys don't get to show me any of your baby pictures or travel pictures, but I get to show them to you because it's part of the presentation, right? And so you come out like this. It's got categories, and the categories are pretty good. If you click into the categories, it's done the right job, it's put everything in the right category. And beyond the categories, I can search for things.

"...this was a true result, but not a useful one. But we didn't give up there. And the field as a whole said, 'Well, maybe the problem is we're only asking for components at the lowest level.'"

So "birds" was a category. I can search for "pelicans," and I get them. "Flowers" was a category. I can search for "clematis" and — oops — I only got four out of five. So the top right one, that's not actually clematis. So it's not perfect, but it's doing a really good job. And beyond just doing binary classification of "Is this in the category, or is it not?" it's also doing graded classification. So if you ask for a keyword, it will show you the best examples. So I ask for "nose," and I get a bunch of really prominent noses, even though I have lots and lots of other pictures that happen to have noses in them.

Okay, so that's pretty transformative, that all this can be done, these millions of categories, all done automatically just by looking at examples and having some clever work of dealing with them.

Now, I want to give you some idea of how this type of stuff is done, but there is a cliché that every time you show a math formula, you lose half the audience — and I'd have to show so many formulas that we'd be out of audience members. So I won't do that, and instead I'll use an analogy. Here's the analogy: Suppose you run a company that sells these kind of tile displays, and you have a catalog where people can order — this looks like a nice scene of Tuscany, or something like that. You've got a catalog with a bunch of different tiles and people can go for them, but then you say, “You know what? This is the internet economy, so we should have something where people can upload their own picture and then we’ll make the tiles and ship them out to them.”

It turns out that is really popular, but there is a problem, that making custom tiles one at a time is really expensive, so you’re not making a lot of profit on this. So you get the idea that what we should do is, we should have inventory of tiles, and when people upload their picture, we should go into that inventory, get out the tiles that could be pieced together to make their picture, and ship that out to them. Now the question is, what tiles, what little pieces should you put into inventory? That’s the question.

How do you answer that? Well, one way is to say, “What pictures am I going to get in the future?” and that’s probably representative of pictures I’ve seen in the past. Take pictures that people have already had, or wherever you can find them, get a bunch of pictures, and then there’s some fancy math that I’m going to omit here, which says, “Out of all those pictures, what are the most important pieces?”

You probably want to have a piece that is all blue because there’s a lot of sky and lot of ocean and you need some blue for that. It doesn’t matter if it’s exactly the right shade, because we’re allowed to be a little bit off. You probably want some noses and eyes and faces because lots of pictures have people in them, and so on. I’m just thinking off the top of my head, but there is a mathematically precise answer to “What are the component pieces that best make up all the pictures in this collection?”

So that was tried. Bruno Olshausen at Berkeley was one of the first to try it. He did it, and then he looked at the results, and it was very exciting to see. What do you see? This is like the primitive set of all things that can make up all pictures. And what he discovered is, pictures are made out of lines. So that’s kind of disappointing. It’s true, but we already knew that in kindergarten when they first handed us a pencil or a crayon. We knew you can make anything out of straight lines. And here there are some that are horizontal, some that are diagonal, some that are diagonal the other way. So this was a true result, but not a useful one. But we didn’t give up there. And the field as a whole said, “Well, maybe the problem is we’re only asking for components at the lowest level. What if we had three levels of inventory, and so the lines make up small pieces, and then those can make up bigger pieces, and those make up bigger pieces still?” And that was the transformative step.

That happened around 2012 — there were some precursors before that — and it actually worked! So when you do that, at the first level, you still get the lines, but at the next level, you get things like eyes and noses, and at the next level, you get things like faces. So this system automatically said, “A face is an important thing, and I’m going to allocate some of my inventory to that.” We’ve gone on from there. We build systems now, not with just three levels, but like with 20 or more levels, and there are lots of complications to it, but it all comes down to saying, “Find the most important pieces, combine them together, and now when you get something new, you can choose from that inventory to create a new one.”

And what can you do with that? I want to show another example. We’ve shown a little bit of language stuff, and we’ve shown a little bit of computer vision stuff. One of the tasks is to put those two together. So here’s the task of caption creation. So given a picture, can you make a caption for it? So here’s a picture, and we ask a human to come up with a caption for it. They said, “Three different pizzas on top of a stove.” Then we trained the system by showing it lots and lots of pictures and corresponding captions and it learned a model of how to go from picture to caption. And then we showed it this picture, and it came up with two pizzas sitting on top of a stove-top oven. That’s pretty good, because you have to look really carefully to see that on the left there is a tomato, and it looks like a spinach or a pesto or something, so it’s doing a good job.

I can go on like that, but it’s no fun to show the good jobs. So let’s show some of the mistakes, both because they’re fun and because it gives you some idea of where this type of system breaks down. So here’s a rather unusual picture, and the caption it came up with is, “A couple of giraffes standing next to each other.” So the system had no experience with horses in purple pajamas, but if you look at the pattern on the pajamas, it looks a little bit like the reticulated giraffe pattern. I don’t know why it came up with a couple rather than one, and that’s part of the problem with these systems — they make mistakes, and it’s hard to understand what the mistakes are from.

Here’s another one, “A reflection of a dog in a side-view mirror.” Objects in mirror may be larger than they appear, and here dogs are just a lot more common than elephants. It didn’t have any experience with elephants in mirrors. Here’s one, “A man riding a skateboard.” I don’t know if you can see it from the back, but there are horizontal marks in the floor boards there that look maybe a little bit like the deck of a skateboard. There aren’t any wheels on the skateboard, but the system just somehow imagined that nobody, not even the King, could be in that position unless they were balancing on a skateboard.

Okay, so this is the promise: We can do amazing things. It’s not quite perfect, but what are some of the challenges? We said we could go really fast, and that’s great, but sometimes when you go fast, things happen. What do we have to watch out for? Now, one of the fascinating things that’s come up in the last few years is this notion of adversaries. I can do pretty well on most queries. When I did the disambiguation, I got up to 97 percent, but what if there is somebody out there who is specifically giving me an example that's especially hard for my program? It understands what my program is doing, and it's trying to actively defeat it, rather than just observing nature.

Here what I have is, schematically, the decision boundary of pictures, and all of the pictures on one side are labeled as pandas and all the pictures on the other side are labeled as gibbons. So it's done a pretty good job of that. You can see there are a couple of red and green that are across the wrong side, so it means those are ones that it got wrong, but mostly it gets them all right. But now, I can ask, "What if I took this picture of a panda and that dot represents this place in space where that picture occurs, and what if I just moved it, ever so slightly, until it went over the boundary and into gibbon territory?" I can do that because I have an understanding of my model, and I can do some math, and I can ask, "What's the minimum number of pixels that I have to just tweak a little bit in order to shift it over and make the computer give the wrong answer?" Or is it even the wrong answer — right?

"with a machine learning system...There is no one place where you can say, 'Everything was good flowing in, and it was bad flowing out.' Rather, everything affects everything now. So we can't debug it, we can't change it, and we can't really ask it to explain itself. There's a lot of work going on to make that better, but we haven't got there yet."

So here's the picture of the panda. Here's the computation of how much I have to move each of the pixels in each direction so some of the pixels get a little bit more red, some get a little bit more blue, and this .007 here means I'm adding mostly pandas and only 7.10 of a percent of the little bit of shift. It's a very small shift. And what do I get? Is the picture going to look like a half-panda, half-gibbon that is slightly more gibbon? What's it going to look like? What do you think? The answer is, it not only looks like a panda, it looks like the exact same picture — right? So I've made this change, which totally fooled my program but has absolutely no effect on fooling your visual system.

Clearly, the program is doing something very different than you are, and by looking at the fuzziness of the noise we added in the middle, it seems that it has something to do with very high-frequency changes that your visual system just doesn't detect. So the aggregate accuracy, overall, might be very good, but it's making some really bizarre mistakes, and we want to understand that. And this really changed my way of thinking about this. Where I used to think that we had divided up the world pretty well, and maybe around the boundaries between two different types of images, we would make some small mistakes — but now I think it's not like that at all. Now I think there are these very narrow corridors that we get right, and in between them, we have no idea whatsoever. That there's just zillions and zillions of possible pictures, and most of them, we have no idea. And the only reason we get a lot right statistically is because everybody takes almost exactly the same picture.

So there's a problem in debuggability and explainability. If I take a traditional program and it has a bug in it, there are techniques I can use and I can say, "Aha, I know that the bug must be in this module right here, because coming into it everything was good, and coming out of it, everything was bad, so, therefore, I have isolated the bug. It has to be here." But with a machine learning system, it's just not like that. There is no one place where you can say, "Everything was good flowing in, and it was bad flowing out." Rather, everything affects everything now. So we can't debug it, we can't change it, and we can't really ask it to explain itself. There's a lot of work going on to make that better, but we haven't got there yet.

There are also a lot of issues around privacy, security and fairness. Privacy — if we're going to base systems on data and data is valuable, then it would be nice if we could collect data from lots of people, but we don't want that data to leak out. We want your data to be secure, and there are some nice systems for being able to train multiple systems, aggregate them together in a way that you can never get out individuals' data, but that is a continuing research field.

Fairness, as well, is a big issue, where we have to say, "What is it that we really want, and are we achieving that properly?" So I've said we've changed the way we programmed from saying, "This is how we're going to do something step by step," to just saying, "Hey, here are some examples, and here's what I'm telling you you're trying to achieve." But that puts a lot more emphasis on this decision for what is it that we're trying to achieve.

"It used to be software companies would release only once a year, you got a new update. Now it's much more frequent, because we can distribute over the web rather than by putting things in boxes and shipping them out, but we're not yet to the point where we can reliably say, 'We're going to have a new update every second as new data comes in.' We don't have the systems in place to do that kind of reliability. So these are the types of issues that we're dealing with. I see a lot of promise. I also see a lot of challenges."

Let's give you an example. Let's say I'm doing a speech recognition system, and I achieve a certain level of accuracy, and I want to increase that. So I'm asking, "What's my measure? Where am I at now?" and I say, "My measure is the total number of correct recognitions of words over all my users." I'd say, "I'm at 95 percent, and I want to get to 96 percent."

Now, say I come up with an idea that says, "Hey, we've been trying to optimize across all of our users, but if we split the users up into groups, depending on the way they speak, then maybe I could optimize one of those groups at a time and I could increase those." You'd say, "Great! That sounds like it'll work." Now, if I'm going to do that and I have a limited amount of engineering resources and there are five or ten different groups of speakers, who am I going to work on? Will I work on the large group with millions of people, or will I work on the small group with only thousands of people?

Well, if my goal is to increase the overall score, then I've got to work on the big group, because then I'll get a big return for a small amount of work. So that means, without me having any ill intentions, I've said my system is encouraging me to help the majority group get richer and the minority group get poorer. For speech recognition, this group could have to do with the shape of their larynx, or something like that, and in other applications, it could have to do with other demographic-type factors.

If we're okay with that, if we're okay with saying, "The people who speak in this kind of voice will do better and we don't have time to improve it for these small minority group speakers," then that's what we'll get. If we think that's not fair, then we have to change our objective. Then we have to say, "I thought I was trying to optimize over everybody, over the average, over all people, but instead, maybe I want to optimize over each group individually, maybe optimize over the score per group rather than the score per person."

And then I have to decide which groups are worthy of that kind of protection and which groups aren't. So right-handers versus left-handers — does that count? People who are wearing blue shirts versus purple shirts — does that count? Or which groups are deserving of fair treatment, and which groups are just sort of contingent and we don't have to worry about? So that's a big question to deal with.

And, finally, there is this problem about the sands of time, or change. We call it nonstationarity, which is just a fancy word, and it means that we train these systems with data that we have. But the world doesn't stop — the world goes on, and new data is out there, and the world changes. And so a system that was trained to do one thing can degrade very easily over time as circumstances change, and now we have a whole new problem in saying, "How are we going to deal with that?" Are we going to update the system continuously? If we do that, are we going to lose old wisdom? Do we keep stuff from exactly one year ago because everything is different during the Christmas holiday and we don't want to forget what we did last Christmas? In the winter, can we forget what we did in the summer? How do we deal with that, and how do we make systems that are secure, safe and verifiable, because we built the whole software engineering system on the idea of when do we release new systems?

It used to be software companies would release only once a year, you got a new update. Now it's much more frequent, because we can distribute over the web rather than by putting things in boxes and shipping them out, but we're not yet to the point where we can reliably say, "We're going to have a new update every second as new data comes in." We don't have the systems in place to do that kind of reliability. So these are the types of issues that we're dealing with. I see a lot of promise. I also see a lot of challenges. It's exciting to work on these challenges, and there are a lot of opportunities.

A video of Dr. Peter Norvig's lecture, which includes his slide presentation, is available at **stevens.edu/lecture.**