Building AI

Alex Ratner on Programmatic Data Labeling For Machine Learning

Podcast Transcript

Craig: Hi, I'm Craig Smith. And this is Eye on AI.

MUSIC

Last year I had Evan Sparks and Ameet Talwalkar on the podcast to talk about Determined AI, their startup, which aims to provide cutting edge deep learning software infrastructure for everyone. Since then, in line with their mission to democratize ML infrastructure, they decided to open source their deep learning training platform in hopes that it will help machine learning engineers more easily build better deep learning models. Ameet, a professor at Carnegie Mellon University and a leader in the emerging machine learning systems research community, is joining me in a series of five episodes to talk to some of his friends and colleagues about various aspects of the machine learning pipeline, from data preparation, model development, and training to hardware management, and deployment.

So far, we've spoken to Turing award winner David Patterson about the explosion of new computer chips for machine learning and to UC Berkeley's Joe Hellerstein about data wrangling, the first step in the ML pipeline. Today, we talk to Alex Ratner, an assistant professor at the University of Washington and a co-founder and CEO of Snorkel AI, about programmatically labeling training data for supervised learning models.

I hope you find the conversation as interesting as I did.

MUSIC

Craig: Alex, can we start by having you introduce yourself? A little bit about your education, how you got into where you are today at the University of Washington, and a little bit about the genesis of Snorkel.

Alex: I'm Alex Ratner. I'm an assistant professor at University of Washington in the computer science department. I'm one of the co-founders and CEO of SnorkelAI, which is a company that's commercializing an end to end platform for building machine learning applications based on a new approach to programmatically labeling and managing training data.

This is an offshoot from an academic and open source effort, also appropriately called Snorkel, that started back when I was doing my PhD along with my

co-founders at Stanford, including Chris Ré who's a professor there. And the key idea and kind of key motivation behind Snorkel, both the company and then, all the way back to 2015, the broader academic project, was just looking from the perspective of a machine learning systems researcher at where the pain points and bottlenecks were in the evolving machine learning stack. And zeroing in on this one around training data.

If you want to teach a machine learning model to route customer support tickets or classified documents or help identify anomalies in medical images, you have to start with a bunch of data that's been labeled according to this particular task. And sometimes this labeling can be done cheaply if you're trying to train a machine learning model to detect cat vs dog vs hot dog, or a pedestrian vs stop sign. But we were especially interested in these settings where you have data that's often private where you need experts to label it, where it changes often.

So maybe you have chest X-rays that need to have an expert radiologist read them, or you have a financial document that needs a lawyer to look at it. How do you get this type of data labeled at scale sufficient to train and these modern, very data hungry machine running models? And the key idea behind Snorkel is just to, first of all, make this data labeling and creation process the first-class citizen of the system and of the ML development process. And second, to let these users, especially subject matter expert users, programmatically label the data using what they know. Using things like key words or heuristics or patterns or other sources of signal, rather than just through labeling data points by hand. So that's the high level in a nutshell.

Craig: And we're talking in this series about the machine learning pipeline. In this case, we're really talking about supervised learning, which is one stream of machine learning.

Alex: Definitely supervised learning is the focus. So again, label some data, say we want to classify legal documents. We take a bunch of legal documents that are labeled with the right tag or the right classification. And then the model is supervised by that training data and then now can do it on new data. So supervised learning, absolutely.

Snorkel is motivated by modern machine learning models that have a lot more power and push button capacity, but trade that off in terms of a lot more complexity, many more parameters and therefore they're commensurately more data-hungry.

And deep neural networks are a great example of this. They automate, or obviate really, a lot of the tasks that used to be so tricky, like picking out the right features of a text document or an image that the model needs to look at. Much more powerful, even in many settings, but they have hundreds of millions of parameters, hundreds of millions of knobs to tune, to learn the right configuration for. And they need commensurately more labeled training data to power this.

So, Snorkel is motivated by these data hungry approaches like deep neural networks. But it's certainly not restricted to it. And for example, in the commercial platform, Snorkel Flow, often we encourage users, we bake it into the UI, to use a very simple model like a logistic regressor or something like that, to start iteration cycles before going for one of these heavyweight deep neural networks.

One distinction that's been crucial here from a machine learning systems perspective, from the perspective of "What do people spend their time doing when they're building machine learning for real world use cases?" is this distinction of deep learning models and other, what are often called, representation learning models, where they not only learn how to draw a line in the sand of how to classify data, but they also learn how to represent data; what features to look at, versus other approaches where you'd rely on a human engineer to pick out the features to look at.

So, for example, in an image, before, you'd spend all your time as a machine learning engineer doing what's often called feature engineering. And this is still a big task, to be clear, but it used to be a huge, huge one, especially for data that's unstructured, like text or image. And you'd spend all your time saying, "Okay, well, we should really look for this task. For chest X-Ray classification, we should look for these types of jagged edges and this type of signal and these types of bright blobs. And let's feed those particular features into the model and let it make its decision over them." Whereas, with a deep learning or representation learning method, you just feed in the raw pixels and the model learns from data, not just how to make the decision, but what features, what aspects of the data to look at. So, it does more for you. But it requires much more data to learn that is the high-level trade off.

Ameet: That's absolutely right. And it requires more data. It requires a lot more compute. The models themselves are more complex, less understood. So, there's less intuition in terms of necessarily why they're working. But...

Alex: Less interpretability.

Ameet: Yeah. But it's certainly the case that, for instance, a computer vision class today is going to be very different than it was even 10 years ago. 10 years ago, you'd spend most of the class talking about the sorts of things that Alex just mentioned. What is a sift feature? Different types of hard features, sift features.

Alex: Histogram of gradients or this and that.

Ameet: And it was just all about, "How do I transform my raw pixel representation into a new representation that experts have devised in such a way that the information is expressed in a more predictive fashion?" Now you don't do that.

Craig: And consequently, the volume of labeled data is much higher because the deep learning network has to discover the features on its own.

Ameet: Exactly. They're higher capacity. They can learn more. They can, in some sense, learn anything. They're very, very expressive. But an older model might've had a hundred or a thousand or 10,000 parameters to learn. Now, you have 10 million parameters. A lot of classical statistics and machine learning theory just literally go out the window. We really don't understand how these things are able to generalize and learn given how high dimensional they are. And that's an open, general machine learning theory question. But you can't argue with many, many, many of the empirical results.

Alex: And another way of thinking about it, going back to the knobs example: imagine you have a logistic regression model that's trained over 100 hand-picked features. You basically have 100 weights to learn, 100 knobs to learn the best configuration from some data.

And you could imagine even doing that as a human, you know, you just look at some data that's labeled, tune and tweak the knobs. Even with a hundred, it would be very nontrivial, but you can kind of imagine. Now imagine you have 100,000,000 knobs to tune. You'd need to look at a lot more data to find anything near the right configuration.

And what does that map to in the real-world experience, from the machine that he systems pragmatic perspective, let's say? Well, if you go to a random organization and that has a data science team or ML engineering team and say, "What's blocking you today?" Like “What are your people spending time on? Where are you stuck?"

To Ameet's point: Five, ten years ago your most common answer would be: "We're trying to pick the right features. We're iterating on trying to pick the right features and we're building the feature extractors and et cetera."

And in some types of data, that's still the key problem today, to be very clear, machine learning and where various solutions, academic or commercial, are most appropriate, very segmented by use case by data type, et cetera. But for certain data types like text, image, all these really tricky ones, that used to be the answer. We're doing feature engineering. That's what we're stuck on. Today, it's much more like, we're waiting for our sister team to label a bunch more trained data, or to fix a bunch of training data.

MUSIC

Craig: Can you give us a sense of scale between the size of a labeled data set you would have needed for a typical computer vision model ten years ago, or before the advent of deep learning, or the validation of deep learning and what you need now?

Alex: I'd say ballpark, thousands of data points and you'd see saturation with older models and now for complex data types, you could get hundreds of thousands, even millions of data points and still be seeing improvement.

The more data you add in, with a simpler model, you just get saturated where you're not improving the model by adding more data. With the deep learning model, you'll often see up to hundreds of thousands or millions of data points for a text or image problem that's difficult, continued improvement with more data.

Ameet: That's absolutely right. And there's been a lot of really recent advances in NLP. With these transformer-based models, like GPT-3 just came out and people have been talking about it a lot and kind of over the last few years, the main difference between these different models is that they're bigger models trained with more data. Conceptually the same thing, it's just more and more data seems to really help. And the amount of data we're talking about is absolutely massive.

Craig: And then today, people are putting their data sets out to business processing organizations, many of them in India, they're using Mechanical Turk as a crowdsourcing solution to label. There's a growing use of synthetic data that creates labeled data, virtually or out of the computer. And your approach, Alex, is very different because particularly the hand labeled efforts or solutions that are being offered require hundreds if not thousands of people who are largely unskilled, following a template to label data. There are pre-labeling systems, whereas data is labeled, a neural net learns what is being labeled and pre-labels, and then the human labeler is just correcting or adjusting. But can you describe how Snorkel works and some of the algorithms behind it?

Alex: Yeah, yeah. So, I love the list you gave. And so, I could start with my attempt at a high-level overview, which is, if you imagine like a decision tree, start at the top: I don't have enough labeled data. How do I get more? And then, high level branches. Okay. You can get experts, by experts I just mean someone who knows enough to label to high quality. So, get experts to label more data, you label data points by hand. And there are various ways of trying to speed this up, of trying to pre-label, especially for tasks where, you're not just saying, okay, is this class A or class B, but you're trying to say, put a bounding box around something in an image. Then you can have efficiency gains by presetting the bounding box.

Like some approaches do, or pre-filtering the data points a little to pick the ones that might be more informative to reduce the total number you need to label, which is traditionally called active learning. That's one bucket of approaches when you don't have enough labeled data.

There's another set of approaches where I'll broadly just say transfer learning, or also you could include multitask learning under this, where basically the strategy is to reuse or pool data across tasks. So, the classic transfer learning thing is I'm going to

train a model or pre-train on dataset A and then try to tweak it to use on dataset B. But, it gets something out of the training on A, where I did have a lot of appropriately labeled data. Or jointly train a bunch of tasks together in a multitask learning approach, so you kind of share information between the labeled data sets.

And then the third category approach, which we did not invent. We just spent a bunch of time building the first system we're aware of back at Stanford that really made this the front and center and first-class citizen of the process, which is, we'd call programmatic or weak supervision. Those aren't entirely synonymous, but the basic idea that, rather than labeling data by hand, we can ask experts to say what they know in the form of something like a function. Like look for this keyword, or look for bright blobs in the image. Or frankly even, apply this other model that I trained before, apply this legacy rules-based system that I had from before.

Anything, whether it's an expert's head or an external knowledge resource, like a dictionary ontology, a model, a legacy rule-based system that an organization has, use that to label data. And the benefit of this, of essentially labeling data with programs rather than by hand, is all the benefits you could imagine from this shift from hand labeled data to programmatically labeled: you can do this a lot faster. You can write a couple dozen, we call them in Snorkel, labeling functions, rather than labeling hundreds of thousands of data points by hand. You can do this in a couple of hours, rather than a couple person-months. You can very easily tweak or modify the way you're labeling the data, which you just can't do with hand labeled data.

You can audit it, again in a way you can't do with hand labeled data, to say, "Hey, what is going into my model? Are there any biases or any mistakes that are informing or supervising the model?" You can have modularity, you can actually reuse bits of these programmatic labelers across tasks. And also, it's more privacy compliant, because you can programmatically label your data rather than having to ship it to humans somewhere often out of your org.

Ameet: It's easier to audit a set of programmatic rules?

Alex: Exactly. And I can get back to this, but I have a whole other rant I could go on about academic definitions of interpretability, which I think is a fascinating line of work, but also pragmatic notions of interpretability. And we found, especially on the company side, that there are basic things like being able to audit what is training your model by looking at these labeling functions that satisfy a lot of the goals of interpretability, but are much more pragmatically achievable compared to, say, the goal of interpreting exactly what a deep neural network is doing and thinking with a hundred million parameters, which is outside of the realm of what we can actually do today.

Ameet: Interpretability is an area I work on actively in my research group. I tell people that the reason I'm working in that area for research is that it's brand new.

We don't have almost any of the answers.

Alex: When a field is exploding on the academic side, that's where you should have at least a little bit of skepticism. On the commercial side, there's still a lot of research around the cutting-edge parts of Snorkel, but the basic stuff is very pragmatic. Whereas, I'd say interpretability is just a very wide-open field all the way down to the theory.

Ameet: Absolutely. I agree.

Alex: So I think in practice, what people care about is, and what we try to be very precise about saying is, we say, "Look, especially since in the commercial platform you can use any model you want, we can't guarantee that we're going to be able to help you interpret every decision that model made, because this is a cutting edge, active area of research and still very much the wild west. But what we can do is, first, we can let you audit what is going in." And then, the second piece I will say, but this is orthogonal to Snorkel, we do this in Snorkel Flow the platform, but it's not specific to our core idea. A lot of the academic community is enamored with the idea - and I think it's exciting - that these big black box models that go from input to the very last output you want to achieve and do it all. But in the real world, I find that most processes are better to break up into sub component steps, rather than detecting activity detection right from a camera feed. First, break it up into frames and then detect people and objects and then make a decision about activity detection. And this ability to inspect intermediate points in a pipeline of modeling steps is another thing that people practically use when they have issues of interpretability.

The core idea that started the whole Snorkel project back at Stanford in 2015, we were looking at the community, we were saying “Okay, well, people are getting stuck on labeling data by hand. It's becoming a bigger and bigger problem as these bigger and bigger deep neural networks come into play and really seem advantageous empirically.”

And we're also seeing, both historically but especially now in practice, all these kinds of tricks and hacks people are taking to label the data, often called weak supervision. And we took all that and said, okay, putting it together, it looks like there's a shift that could be accomplished towards this idea of programmatic labeling.

So then going back to it, there are all these advantages, but of course, no free lunch. The disadvantages of programmatic labeling is that if you ask a subject matter expert to write down a heuristic or a rule, it's very hard to make it high accuracy. Very hard to make it high coverage also. This is why rules-based systems are very difficult to get to work and why machine learning, or at least in high dimensional data like text or image, it's why machine learning has kind of taken over in the last few decades in those areas. But the idea in Snorkel is that if you can have an expert write a couple of these programs to label some of the data, you can clean that up using theoretically

consistent techniques and then feed it to a machine learning model that can generalize beyond it. So, to make that extra concrete, take as an example one of the users of Snorkel Flow, the commercial platform. But I'm just going to speak fluidly between that and the core open source project and the idea is, since we're talking at that level: A big bank is trying to classify some complex legal documents in nuanced and actually ever shifting ways. And again, this is difficult because those documents cannot leave the bank. Even if you had people who knew how to label them in some outsourced configuration, it can't leave the bank.

Within the bank, they needed a special team of legal analysts to actually label them appropriately. And how you want to label them as always shifting first there's 10-way classification. Then it needs to be a 12-way classification the next week. And the type of template that's coming in changes.

And so, this shifts all the time and it's not that you're looking for a cheaper way to get this labeling done. It's just that you just stop trying, because it's just not practical. So, with Snorkel the idea is, okay, let's sit those legal analysts down for an hour and say, "What keywords are you looking for? What patterns are you looking for? Did you write some rules or train some models beforehand that weren't good enough, but have some signal? Let's dump all of that into Snorkel." And the net of that, I can state some results just for color. Say you label somewhere between 70 to 80% of the data with those labeling functions and Snorkel cleans them up. Well, now Snorkel can train a model that, in this case for them, got 97% accuracy. So, it's kind of like bridging this rules-based approach as input with machine learning models, modern ones that can generalize, to capture this complex data as the output. That's kind of a high-level sketch.

The thing that has been really productive has been this decoupling of subject matter expertise of what the labeler knows from the model by this very, very simple, narrow API of, "Just feed me data points with labels. This training data." And this is great because it allows CS nerds like us to sit in a very decoupled way and just crank out new and better models and machine learning techniques. But if you go to the real world, especially in these expertise-heavy settings, where you don't have a lot of training data and you do have a lot of people who know rich things about the domain and the problem, it's almost like asking those people to play 20 questions, or rather 20,000 questions with you or with the model, rather than just telling it what they know.

The one toy example, imagine a legal analyst wants to tell a modern, deep learning model, "Look, if you see the word 'employment' in the title, label it and it should probably be an employment contract." Not a hundred percent correct guarantee. But a pretty strong signal. Right? There's no easy way to inject that directly into a modern model, to just tell it. The only way to communicate it in the de facto approach these

days is just to label data points one by one until the model can guess that that is one of the features you're looking at.

Imagine even as a human, if I wanted you to guess what word I keyed in on some 300-page document, and I can only communicate to you by answering yes/no questions. It would take you many, many data points, even as a human, to understand what I was trying to tell you. So, the high-level idea in Snorkel is to let the expert just directly tell the model what they're looking at and let that bootstrap the model to getting trained.

Craig: The descriptions of the data points, or the key words, for example, you convert that into code. Is that right?

Alex: Yeah. We call them labeling functions. So, it's just a function that labels data and can abstain. And then we model this in a certain way. So, a lot of the work that we did on the algorithmic and theoretical side at Stanford was: Okay. If you have a set of these labeling functions that are kind of just black box functions that either label data or abstain, and you expect that they're going to have very different accuracies and different, even, correlations with each other, you want to up and down weight them appropriately. You don't want to double count two labeling functions that are nearly identical. How do you do this? And since the whole problem in the first place is you don't have enough labeled data, how do you do it without labeled data? And our work builds on lots of very old theoretical work in the space, such as from the crowdsourcing literature and even far back, and then extends it for this programmatic labeling setting.

The core intuition there, and again, this is an old intuition, I think a very, a very cool one, is that, imagine you have a room full of people. In this case, it's labeling functions for us. But say a room full of people. And some of them are mostly always telling the truth. Some of them are most often lying. If I ask enough questions and just observe patterns of where those people tend to agree with each other and tend to disagree with each other, I can probably recover, under very loose conditions, the accuracies of each person.

Ameet: That's even without any labeled information? Or do you need to have some labels?

Alex: No labels needed, and now I'm getting deep into the weeds, but there's a symmetry breaking issue. In other words, one classical problem is that intuitively you could estimate it, but at the end of the day you don't know whether, say, 60% of the people are honest and 40% are liars, or 40% are honest and 60% are liars. You get this fundamental kind of symmetry that you can't break.

We break it in Snorkel just by assuming the 60/40 case. We assume that the user could write labeling functions that are so bad that they're actually almost adversarial,

but that those are going to be in the minority. So, you have to make that assumption. And you have to assume that the labeling functions or the people in the room are not all totally correlated with each other. If you were asking this room full of people these questions, and they were all, like, colluding to give the same answer, then you just don't have enough independent sources of signal to do this. But if you make some light assumptions, and I'll just switch back to calling them labeling functions, if you have at least three labeling functions or three sets of labeling functions that are not totally correlated with each other, and you make this assumption that the person who's writing these labeling functions is in aggregate trying to make them work even if they make mistakes, then you don't need any label data to recover provably consistent estimates of how good they are.

What does this all net to the practice? Snorkel is not a "garbage in; gold out" system. It doesn't mean that you can just write total garbage labeling functions and expect something miraculous to happen. But you can write labeling functions that are kind of, what we would call, noisy or weak. Maybe they're 60-90% accurate on average. And maybe a couple of them are redundant copies of each other. And rather than having to manually go through and adjudicate their disagreements and figure out all the details of which ones you should trust more when they disagree, you can just dump them into Snorkel and Snorkel will take care of doing that part of it for you.

So, Snorkel is very much a human-in-the-loop paradigm. It still relies on a subject matter expert to drive the process iteratively. But it turns machine learning development, and we've seen this over the years and reported on it in the literature and now see it through the company with customers, it turns machine learning development from this kind of one and done process where 90% of the time is on labeling the data, and then, if something changes in the world, you kind of just have to start all over from scratch, which is quite painful in the real world, to much more of an iterative development process where a subject matter expert is trying to write and refine these labeling functions guided by Snorkel Flow.

MUSIC

Craig: I understand how you would do this in labeling text data with keywords, for example. But for something like visual or audio data, how do you write those functions? Can you give an example of how a radiologist would describe what he's looking for and then how you would convert that into a labeling function?

Alex: It's a great question. So, there's kind of two ways we've gone about this. The first way, and this was a big part of the thesis work done by my former lab mate, Paroma Varma, and now co-founder at the company, the basic idea is in the approaches that she and I and others in the lab pushed for was: let's try to make some basic building blocks, we called them primitives, and have people write labeling functions over those. So, I'll give an example. Imagine you're trying to, this is kind of

silly and toy, but to train an image detector to detect if someone is riding a bicycle. Or if someone is, a gun or engaging in violence or something like that. You could first, and very easily today, apply a bounding box object detection algorithm, there's many of them pre-trained off the shelf, and tag all the bicycles and the people. And then you could have the labeling function say things like, "Look for the bicycle and the person to be horizontally aligned. Look for the bicycle and the person to be of the same ratio and scale. Look for the person to be above the bicycle by a little bit."

And so, the labeling functions are now written over these kinds of building blocks that were pre-populated for you. Because to your point, it's very difficult to imagine how you'd read a labeling function directly over pixels the same way you could write over words. The second line of work that we did around dealing with image video and similar data is often you have settings where you have, at training time when you're trying to train the machine learning model, some image or video data and some other data like text or structured.

So, for example, we did a paper with a couple of teams, including some radiology teams at Stanford Hospital in the VA healthcare system, where at training time we had chest X-rays, hundreds of thousands of them, and none of them had the exact label we were trying to look for. In this case, it was for a triaging application. So, we were trying to just train a model to look at an incoming chest X-ray and decide whether it could sit in the queue or needed to be urgently read by a radiologist. So, in this case, that exact label wasn't there, but we had hundreds of thousands of images and we had all these unstructured notes that were text.

And so, what we did, we call this the cross-model paradigm in such a similar use case deployed in a case study we did with Google, you write the labeling functions over the text, but then train a model over the image. So, when a new image comes into the hospital system, the labeling functions you wrote are inapplicable because there's no text report for it yet. But the model you trained is over the images, and so it can function. These are two techniques that we've used for adopting this paradigm to images. And obviously, Snorkel is not the most appropriate technology for every single, image or video use case.

But we found that in many different types of use cases, using one of these two techniques was actually pretty advantageous.

Craig: And for the user, particularly, the user of Snorkel Flow the commercial platform, presumably you've built templates that people can fill in with different kinds of natural language...

Alex: Yeah. we wrote about this on the academic side as well, and this idea of, why do people like code and why would we care about this shift from hand labeling to programmatic labeling? And the answer would be, if it stopped there, we probably shouldn't care as much. But the exciting part is that once you've shifted the code, you

can build on all the layers of abstraction above code that are traditionally used to automate any kind of software development or functionality there. You don't need to write labeling functions in C or in Assembly. You know, you can write them in very high-level templated ways, or even make them in push button ways. So, this was part of a big set of reasons why we spun out the company from Stanford, was that we were serving our users. And at Stanford, this was mostly doctors and scientists and journalists. And the number one request we got was, "Please don't go try to prove a sharper bound on like structure learning for Snorkel. Please just help to make it faster, to write and iterate on labeling functions" and other things like that.

And so, what we've done in the commercial platform, Snorkel Flow, is make, let's call it, our target is aspirationally 80, 90% of the types of labeling functions you'd want to write, the types of signals you'd want to bring to bear for a given data type and machine learning problem, make them push button. So that a subject matter expert that doesn't know how to code or doesn't just want to spend the time coding, can rapidly create a set of labeling functions without having to actually write code. And then you can always go back and write codes with the edge cases. But this ability to have push button supervision, and then of course, there's also crazier stuff on the research side, like my former lab mate and another cofounder, Braden Hancock, did a line of work where you could actually describe your rationale for labeling things, which would then get compiled by semantic parsers into labeling functions. But the key idea is that once you make this crucial shift from, supervising models by hand to supervising models with code, you have so many layers of abstraction you can build on top of that.

Ameet: I have a few questions. So, this is super cool and inspiring, and I think it's such a great idea. And,I love that it's open source, and the idea of making it even further accessible and practical for users in commercial settings makes so much sense. Two technical questions I have. At a very high level, my understanding is that you can use something like Snorkel to essentially take unsupervised data along with expert knowledge to create labels in a programmatic, automated way, in some sense. Now you have an actual label data set. Is the idea that now you can treat it as a typical supervised label dataset, and then just train, say, your deep learning model? Or is there something special you have to do during training as well, once you get these reconciled labels?

Alex: There's always like the big picture idea, and then there's all the nuances that go into productionizing a platform around it. And so, big picture, the one kind of shift that's not overly complex to understand or to adapt to is that the output of the labeling functions stage is a set of probabilistic or confidence weighted training labels. Just intuitively expressing, if you have 10 labeling functions that are all high accuracy, according to Snorkel's estimates, that labeled this one data point all in the same way, you want to weigh that more in your model training than one that was under

contention and really confused. Or, the labeling functions were effectively confused on.

So that is one difference. I'd say the other big difference is that if I have a hundred unlabeled data points, let's say on average, maybe a user will practically write labeling functions that cover maybe 50 of them. And it may also be in a distribution that's very different from the natural distribution, which is actually one of those nuance points that ends up having a lot of impact. And so, the goal of the overall pipeline is to take those 50 maybe oddly skewed distribution-wise data points that were labeled by the labeling function part, and train a model that can now label all 100 data points effectively.

Ameet: So, the two things are: One, the final labels are probabilistic, not binary. And then they're incomplete. But that second point you said you have another model that goes from the incomplete set of labels to the complete set of labels?

Alex: That's just the model you're training. So, the first part of the Snorkel Flow pipeline is the labeling functions, and what we called in the literature, the label model that re-weights them and fuses them together. Then to your point, Ameet, you're just using that as a training dataset.

Ameet: Another way I think about it, then, is you could argue that with something like Snorkel, because you're doing it programmatically, you might end up with a much larger set of labeled data points than you would if you're doing it one by one. So maybe two differences can be, one, that the size of your labeled dataset is actually bigger in Snorkel, which is a benefit. The labels themselves are maybe noisier and probabilistic rather than binary. Not to say that a human labeler is perfect anyway. But once you're at that point, the idea is that you can train a standard deep learning model. If you had an ability to train a standard deep learning model and handle probabilistic labels, that next step is a black box.

Alex: Yeah. And again, there are finer grain pieces. We had a line of work we called "slicing" with this notion of a slicing function, where you can, rather than labeling parts of a dataset, you can specify certain kind of important parts of the dataset and that affects then how you actually construct your model to learn representations over those special slices of the data. So, there are nuances. There's a broader picture of stuff we've done here that really is end-to-end. But that core idea of, you can slot in any model here as a black box, that is absolutely correct and very central to the design. That modularity, even on the academic side, to us is very important. Because look at the pace of new models coming up. You want to be able to, whether it's us or a user, to slot in the new image model that came out that's better for their task, and have that be pretty decoupled.

Ameet: Part of the reason I'm asking this, too, is just thinking about this with my Determined AI hat on. Kind of the theme of this podcast, generally, right? We've been

talking to a bunch of amazing people over the last few weeks. You can very coarsely view the machine learning pipeline as having three stages: everything before training, preprocessing data, labeling data. Then actually model development and training. And then everything afterwards, actually deploying it and monitoring it and so on. And I would argue that those three steps themselves, while not completely separate, right? There're obviously ways that they need to integrate with each other. They are kind of somewhat distinct.

And Determined is largely focused on the model training and development part. Really focused on deep learning for a lot of the same reasons that we've been talking about on this podcast. That they're really complex and really hard computationally, they're data hungry, which means that from a systems point of view, they're really challenging to train and develop and just keep track of in production. But on the flip side, they're what you want to use in many cases, because they're so powerful. But, those sets of problems, in some sense, to me, seem complementary to this idea of, "How can I feed this very data hungry model development and training component," right?

The first bottleneck of that is, "How do I get the data in the first place?" There's this other bottleneck of, "How do I now train this?" I have a good fortune of having this very, very valuable resource of huge amounts of training data. "How do I actually develop these models and train them and tune them and so on?" That's kind of where we fit in, and it seems like these two things are happy to fit together.

It is certainly the case that for us, we are banking on the fact that the more data you have, the harder the training problem is. But as you said, it's exactly the problem you want to be having because without data, there is no machine learning.

Alex: Oh, exactly. It's very complementary, even in that kind of incentive structure, because the bet that we've shown like when we brand studies on the academic side with Snorkel, it's if you have 100,000 data points and 100,000 labeled data points, then our metrics would be, okay, how close can Snorkel labels get training-wise to the gold standard. But the places where we blow traditional supervision out of the water is where maybe you have 50,000 data points labeled by a radiologist, but you have 500,000 unlabeled ones sitting on the hospital server. You can now label them with Snorkel Flow for no extra effort other than some compute cycles. And then, to your point, that also then makes the training burden that much harder. We've learned it's very important to support the ability to train a model, see where it makes mistakes and then go back and iterate on your labeling functions as a complete loop. We also have Python STK and full connectivity, because we understand that it depends on the problem, but when you have a problem with a lot of scale that's like a serious production problem, after you've done some those initial iterations, we want to be able to couple them with the best in class approaches for those next stages of the pipeline, like Determined, to flow into.

And so, I think we learned you need both pieces for a Snorkel-like approach. You need to have some way to quickly iterate with just a simple model and then you need to be able to use best in class tools and platforms to get to those second two stages, like you said.

Ameet: That's super cool. I had another technical question, and I think you sort of answered it. But I'm going to ask it anyway, just maybe to have you rephrase it, which is: The reason we like deep learning models in practice is that they're higher capacity, they're richer, they, very bluntly, get better accuracy on some of the problems that we care about. But at the same time, the labels that you're generating, if I assume that human labelers are the ground truth, which again is a caveat in and of itself, but if I do make that assumption, then that's sort of the upper bound on your performance. Assuming the same amount of data. Right? And granted, you can have more data, which maybe allows you to overcome this barrier. But, I guess the question is how do you reconcile the fact that the deep learning models are richer, higher capacity, can learn more complex decision surfaces, with my assumption, which is maybe wrong, but my assumption that the labels that are learned per data point on Snorkel are slightly noisier? Or is there anything that needs to be reconciled there, or is it just with enough data things just magically work?

Alex: Yeah there are a couple answers to this. as you noted, in practice, ground-truth, or often called gold hand labeled data are not often so ground-truth or gold. one of the data points that always sticks in my head I heard at a bioinformatics conference. In this community, a lot of the annotation is sponsored by the National Library of Medicine. These are people who are specially trained to, annotate biomedical literature, for example. And there, the inter-annotator-disagreement rate, how often labelers disagreed again, amongst, specially trained experts for this task, was like 20 to 30%. So, even ground truth can be noisy.

And that is where we have seen Snorkel labels actually be better because you can keep refining them. That's kind of the second aspect I would say, is that, ultimately at the end of the day, especially when you actually have a platform version of this, so it's fast, it's an iterative process.

So, with hand labels, you have some quality bar that you get. But you're kind of fixed. With an approach like Snorkel, maybe there's some noisier data points to start, but you can iteratively go to the parts where the model is not performing well enough and refine or sharpen the labels.

Even add in hand labeled data points and tactical, as we had a paper with Apple with the basic Snorkel technology, it was a version of it called Overton where my colleague, Chris, talked about this hybrid approach where hand labeled data and programmatic data can work together. Regardless, it is definitely the case that some of these labels can be noisier to start, but this ability to iteratively refine them, which

you just can't do with hand labeled data in that way, often has a big practical impact, is what we found.

MUSIC

Craig: You have a dozen experts input things that they're looking for, and the platform converts that into functions. Let's say you have a collection of a million images, or a million x-rays. At that point, it's just a push button by the data scientist, is that right? Or even someone more junior to apply that to the data set. And on that first iteration, and I understand then that you can refine, but are there any metrics about how far that gets you in terms of accuracy, compared hand labeled data? I mean, you were talking about how hand labeled data isn't necessarily accurate, there's a lot of noise in that data too.

Alex: First, I would say just about the platform, Snorkel Flow, and I think this is just a philosophy of certain kinds of data science platform building. We try to make things as fast as possible at the push button level, but really take an onion approach. Right? Our core users, we expect them to be experts who often want to move very quickly through things, but want extra levels of depth that they can get to when they want to dig in and get more technical.

So that's the whole design philosophy of everything, is push button at the start, so that a non-data scientist subject matter expert, or a data scientist who just wants to crank through really quickly can really do that push-button experience, like you said. But when they want to go do it and they want to define stuff via code or via some SDK or API, or they want to connect to a more heavyweight solution, they can do that, too, just as easily.

That's kind of the design philosophy that we very much built the system around, the platform. And then to your point about quality comparison, it's most often what people care about is some kind of time-cost trade off. And so, as we were talking about with Ameet just before, if you have 50,000 labeled data points, and you get 50,000 unlabeled data points to Snorkel, we can match the quality of the labeled data points. You can iterate until you match the quality. You know, in many cases, you have much more unlabeled data than you could ever label by hand, in which case Snorkel Flow can do much better, right? It's sort of like a larger amount of slightly noisier data can outperform a smaller amount of very pristine data, is the way to think of it intuitively. And then it's really just the ability to move faster, both in getting your model up in the first place, as well as adapting it over time as input and output goals shift.

Craig: And you were saying that there are cases where the programmatic label can surpass the quality of hand labeled data.

Alex: Yes. especially when you have more unlabeled data. So again, a lot of the cases that we work on, you've got a bank with hundreds of thousands of complex legal documents internally, or a radiology center with hundreds of thousands of chest X-rays. And again, their bottleneck is they just don't have the ability to label them. And so, by letting Snorkel leverage all that unlabeled data and all the info stored in that unlabeled data, you can do much better than you would if you spent the same time or even 10 times the same amount of time labeling data by hand.

Craig: And do you have some comparisons on cost and time? Take a standard data set size and what it would cost in terms of time and manpower using programmatic versus hand labeled?

Alex: we have a bunch of comparisons in the literature. Like most things with machine learning, it's very dependent on the data type and the machine learning problem type. And for Snorkel, what types of labeling functions you're writing. There are some settings where you can write a couple labeling functions and you just get a lot of bang for the buck. Other ones where you don't have those kinds of knock out labeling functions. It's a little bit more involved. And so, it ranges from use case to use case.

We've seen settings, like the radiology example I've been referring to, where we replaced, I think it was 8 to 14 person-months across a couple of these different medical monitoring applications, 8 to 14 person-months of hand labeling with roughly 8 hours of writing labeling functions. And we got within one point. So, with that huge reduction in cost, we were within one point of ROC AUC, that was the metric we were optimizing, of the two approaches. Slightly lower with the Snorkel examples. But again, essentially the same for a drastic reduction in time.

That was kind of like that, that low hanging fruit scenario where you had all these labeling functions you could write, leveraging the metadata and things like that that were just noisy. But in Snorkel, [they become] really powerful. You know, we published a case study with Google, a couple applications there, where we showed we could replace tens to hundreds of thousands of hand labels.

And so, I don't have stats on what that means in terms of time or cost, but, you know, those are tens or hundreds of thousands of labels. And they were regularly labeling and relabeling, because it was a setting where there were shifting dynamics.

There are settings where there is a massive difference in time and cost. And then that most common setting we see is where you could imagine labeling the data by hand, and you could imagine doing it and maybe spending a month or two, and you could stomach that cost. But it's the fact that in a month, or a week, or even a couple of days, you're going to have to go back and relabel it again that is actually the most painful part. And the ability to, instead, go back and just change some code or tweak some templates with a GUI. And again, I think there's a thing that the field of ML

systems and just practical machine learning in general is realizing is that getting a model to a certain accuracy, like we've optimized for on the academic side, is one thing, but maintaining it over time and adjusting it, and serving it, and retraining it. That's the bigger cost over time.

Craig: Do you use different augmenting strategies, then? Once you've got a certain base of programmatic data? when you and I spoke before you talked about augmentation.

Alex: Yes. So, on both the research side, and we published about things like data augmentation, I mentioned slicing, and this is also in the commercial platform, we're very interested in a general way of "How do people essentially program modern machine learning models via the data?" And, you can often call this weak supervision, but let's just say, how do you program models via the data? Because it seems like what a lot of people are doing. And one of the things you do is you label the data. And that's kind of the initial point we started at. It's very clearly, for supervised learning, the most important one.

But there are also a whole bunch of other techniques that we saw over the last five years of this research project, and now at the company, that were having huge impacts on model performance and what people are focused on. Data augmentation is one of those techniques. Not something we invented, it's something that people had used to incredible effect. You look at any image state-of-the-art number of the last five to ten years and the gap between data augmentation, not-data augmentation -- in the case of images, the canonical example's like, you know, rotating or stretching or blurring an image to kind of create transformed copies. But essentially, you're teaching the model, "Hey, rotations shouldn't change how you view the world." So, that's in a nutshell the idea of data augmentation. The difference in performance is like ten times more than the difference in performance between like the first and second fanciest models.

So, it's a huge effect. And it's just done in this kind of ad hoc way. And so, in 2017, we did a paper on automating data augmentation. Based again around, now, instead of a labeling function, a transformation function. So, you'd ask an expert to provide some atomic operation, like a rotation or a shift or swapping a synonym, and then use that to augment the data.

MUSIC

Craig: That's it for this week's podcast. I want to thank Ameet and Alex for their time. If you want to learn more about what we talked about today, you can find a scrolling transcript of this episode with speed controls on our website, www.eye-on.ai.

We love to hear from listeners, so if you have comments or suggestions, feel free to reach out.

And remember the singularity may not be near, but AI is about to change your world. So, pay attention.
