EPISODE 1044

[INTRODUCTION]

[00:00:00] JM: Machine learning models require the use of training data, and that data needs to be labeled. Today we have high-quality data infrastructure tools such as TensorFlow, but we don't have large, high-quality datasets. For many applications, the state-of-the-art is to manually label training examples and feed them into the training process.

Snorkel is a system for scaling the creation of labeled training data. In Snorkel, human subject matter experts create labeling functions, and these functions are applied to large quantities of data in order to label it. For example, if I want to generate training data about spam emails, I don't have to hire 1,000 email experts to look at emails and determine if they're spam or not. I can hire just a few email experts and have them define labeling functions that can indicate whether an email is spam. If that doesn't make sense, don't worry. We discuss it in more detail in this episode. Snorkel is a pretty cool project.

Braden Hancock works on Snorkel and he joins the show to talk about the labeling problems in machine learning and how Snorkel helps alleviate those problems. We've done many shows on machine learning in the past and you can find those shows on softwaredaily.com. If you're interested in writing about machine learning, we have a new writing feature that you can check out by going to softwaredaily.com/write. You can write about what you're learning about through the podcast and that might be useful if you are an active learning type of person.

[SPONSOR MESSAGE]

[00:01:47] JM: This episode of Software Engineering Daily is brought to you by Datadog, a full stack monitoring platform that integrates with over 350 technologies like Gremlin, PagerDuty, AWS Lambda, Spinnaker and more. With the rich visualizations and algorithmic alerts, Datadog can help you monitor the effects of chaos experiments. It can also identify weaknesses and improve the reliability of your systems. Visit softwareengineeringdaily.com/datadog to start a free 14-day trial and receive one of Datadog's famously cozy T-shirts. That softwareengineeringdaily.com/datadog.

Thank you to Datadog for being a long-running sponsor of Software Engineering Daily.

[INTERVIEW]

[00:02:42] JM: Braden Hancock, welcome to the show.

[00:02:44] BH: Thanks for having me.

[00:02:46] JM: Developing machine learning models requires the use of training data, and that data needs to be labeled. Explain what training data is.

[00:02:56] BH: Yeah. With the most profitable by far branch of machine learning right now, supervised learning, you train a model by showing it lots and lots of examples of basically the decisions that you wanted to make, the classifications that you'd prefer it. It'd come to learn, so that when it sees a given input, it's able to come out with an output of it. Right now, it's basically a lot of show and tell with models, and if you want it to be able to work well in the future, you need to show it lots and lots of examples where you've already told it what the answer is, and that's what it goes and picks up on.

[00:03:26] JM: How does data typically get labeled?

[00:03:30] BH: Yeah. The typical approach right now is almost comically manual in a way. It's surprising at least to me personally that we still generally work at such a low-level, but right now it's generally that each individual example you look at, you or someone being paid by you I guess, crowd working is often a way people will approach this. But you look at individual examples and decide what label you would give it as a human and you work that down and you move on to the next one, and so it scales very much linearly in terms of time and cost of labeling as human eyes essentially need to see each and every example that you're going to show to your model.

[00:04:04] JM: Tell me more about how that results in problems, that manual labeling process.

[00:04:09] BH: Yeah. I mean, for one, it gets expensive real fast especially when the person who has enough subject matter expertise to do a labeling is someone whose time is expensive. When you've got a very narrow domain where there aren't a lot of experts and all the time labeling needs to be done by you, then that can add up, because to get very best performance at a lot of these models, you need thousands, tens of thousands, sometimes hundreds of thousands or millions of labeled examples, and that adds up real quickly. Sometimes there are limitations in terms of who can do the labeling.

There are also often issues around how static this is. I think anyone who's worked with machine learning and especially labeling, training data in production, has run into this problem where you specify the specs by which data should be labeled. Then after crowd workers or others you're working with start labeling, you realize, "Oh! That's not quite precise enough. I actually meant that we should handle these edge cases in this way or that way," or you realize that maybe your schema is not quite right. Maybe we need actually three classes here; positive, negative or neutral, and not just positive or negative.

There's the element of like learning what exactly you intend for your model to do that can change overtime, but also just the fact that products that are out in the world or live machine learning applications naturally experience, changing product requirements overtime or just the data distribution shifts. There's this effect where your training data essentially becomes either ill-fit for your problem because it's now of the wrong schema, or there's a distribution shift.

The examples that you're using to teach your model what to do no longer reflect the reality of the examples that it's going to see at test time. So it ends up being subpar in those areas. That's why this manual labeling process ends up being not just a one-time cost, but an ongoing, recurring wheel of burning time and money really to get optimal performance. You need to be constantly labeling fresh examples.

[00:06:04] JM: I know this is a very subjective and project-dependent question, but how much data do I need for machine learning? Do I need as much as I can get? Is there diminishing returns to training data? Is there some cap on how useful an amount of data I need can be?

[00:06:27] BH: Yeah, that’s a great question, and the short answer is of course it depends, which is very helpful, I know. But no, it depends on a number of things. There are some rules of thumb. I mean, in general when you have a relatively small set of features that you’re learning to learn weights for, you can get away with having a smaller dataset. Examples of these are I’ve got a spreadsheet, I’ve got structured data fields, maybe 10 attributes for each thing that I’m classifying, and I just need to learn basically how to combine these 10 attributes. That’s pretty doable with a small dataset. You might be fine with a few hundred examples depending on the complexity of how those different features interact. But a lot of the places where we’ve seen the most exciting progress in machine learning in recent years is over what are traditionally very hard domains to work in, ones that are very high-dimensionality. That’s things like images or text.

In these domains, we’ve been seeing huge lifts by taking advantage of deep learning models which are able to essentially learn relevant representations, learn relevant features to use for a given problem. They’re able to do that because they’re very deep networks. They’re not just learning weights for combining features that are in the lower levels essentially learning relevant characteristics of the data to pay attention to, and in the higher levels they’re learning how to combine them. This is very powerful, but to train that many more weights to – Now these models end up having not tens or hundreds of parameters, but truly millions or hundreds and millions parameters in some of the most recent state-of-the-art models. The tradeoff is we can get awesome performance, but we really do need massive amounts of labeled training examples typically to accomplish that.

[00:08:05] JM: The subject matter experts that might be manually labeling data, if we’re talking about a manual data labeling process, these subject matter experts could vary in their levels of expertise. So maybe if you had doctors labeling pictures of tumors, for example, and you wanted to manually label these. Maybe you have some doctors that label them. Maybe you have some medical students that label them. Medical students have less experience. So you end up with less accurate labeled that, but I can get more medical students. If I had more medical students labeling the data and they were able to label a higher quantity of the data versus the doctors who might be labeling a lower quantity of data, but at a higher quality, how does that tradeoff spectrum gets explored? What’s the impact to the end model when we talk about a high quantity of lower quality data versus a low quantity of high quality data?

[00:09:14] BH: It's something I think that we've observed in recent years with this deep learning evolution as it were as people have found ways to successfully train these very large models that often it's saddening in a way. There's actually a great post about it, how we keep on thinking that making these models more human-like will make them do better, but in the end, just throwing more data at it often seems to bring the largest results, the largest boosts.

Personally, I spent a lot of time in the natural language processing field, NLP, and some people refer to 2019 as the year of the transformers, which is a particular model architecture that's been used to great effect. Essentially, organizations, individuals and researchers keep on training on larger and larger amounts of data. Oftentimes, automatically generated data that were able to scrape from the Internet and do some clever sort of techniques for labeling it, but the result continues to be this ongoing upward trend that rather than painstakingly picking the perfect hundred examples to show a model, if I can get 100,000 good enough extremely cheap to acquire data examples, that ends up being the winning factor.

Obviously, this doesn't work in all cases. There is potential if you've got these med students versus doctors situation. There may be something that only the doctor knows. Ideally, the ideal solution is we have a way of combining some votes from doctors and some votes from the medical students and we get extra quantity from the med students and we learn to trust a little bit more the inputs from the doctor, and there should be ways to combine these, and manually combining them becomes a bit of a whack-a-mole problem when you have lots of different sources of supervision signal that you know are of varying quality and accuracy in correlation with each other. I guess maybe I'm still in my punch line here, but that's one of the things that Snorkel's been able to do well, that we've spent a lot of time on the theory and academic side proving that there are ways without additional human input to automatically learn what sources are worth trusting more or less so that you can take advantage of things that are easier to get in high quantity even if you expect the quality to be a bit lower than if your most expert-expert was seeing each individual example.

[00:11:27] JM: If I was feeding data into a model, let's say I had some data that was higher quality and some data that was lower quality. If I wanted to make my model aware of how confident I was in the labeling quality of different datasets, how can I do that? Could potentially just weight the higher quality datasets more? How does it work in implementation?

[00:11:55] BH: Yeah. There's a variety of ways that people do this in practice. One option, one thing that we've done with Snorkel is as we combine different sources of labels, and they may disagree, three of my labelers say that this is true and one of my labeler says this is false. One thing you can do is just say the majority has it. If there're more votes positive, we'll say your positive, and then you just sort of round everything.

But another option is to actually express in the labels themselves what your confidence in them is. If I'm comparing one expert vote versus two non-expert votes, maybe I really trust my experts a lot more. I'll say, "Despite two-thirds of the votes being negative, because my experts, the one saying positive, the final label for this is going to be 60% confident positive or something." It will be different for every situation depending on how much your sources vary in accuracy or how many contributing labelers you have, say, for an individual point.

But as you combine labels, one way is to make these labels not hard integers. Do you belong to class 0 or class I? Are you true or false? But rather to say, "Express your confidence. Make it a distribution." Each data point gets basically an assigned distribution of how sure we are that it has each particular label. Then from there, it's typically a pretty simple modification to standard machine learning models to have them base the loss on that sort of weighted value rather than just a single hard truth number.

There's one other option as well, and this is something we've dabbled with with Snorkel in some fun and public ways, which is multitask learning. That's maybe a separate conversation from how you supervise in general, but there are sometimes ways of changing the order in which your model sees different data. You could pre-train essentially on the lower confidence data to just learn the most relevant representations that you want your model to have access to, and then you fine tune, as they call, it on your very most accurate data, and that's where your model essentially will just learn the best way of combining these relevant features in a way that's most specific to your example.

Those are a couple of techniques, and there are other ways as well. I'd say, in general, there is a lot of additional exciting research to be done in this area in terms of how to combine data and supervision sources of varying signals, and that I feel we've been happy to contribute to in a number of ways. But by no means, is it saturated or solved.

[00:14:19] JM: All right. Well, I wanted to give us a preface for the domain that we're talking about before we get into Snorkel. Explain what Snorkel is.

[00:14:29] BH: Snorkel, it began as an open source project at Stanford a number of years ago, four or five years ago, and it is – Yeah, you could say a framework for automatically or programmatically generating training labels to build training sets. The origins of the project where we were looking – And this is I guess a formula that I would recommend in general for researchers, is take a look at what people are doing in practice to get good results and then see if you can formalize it. Look for the hacks. Then if people are doing it, presumably it's survived the Darwinian sort of setup of research and there must be some value in it somewhere.

We looked at what are things that people are already doing to satisfy this big need that they have for getting enough labeled training data for models, these increasingly data-hungry models. We looked for a way to formalize that to theoretically ground it and make it into a repeatable understandable process. In Snorkel, we recommend an approach where instead of labeling each individual example in your training set, you have a large collection of unlabeled data and your inputs are no longer individual labels on those examples, but rather labeling functions.

These can be a number of different things. As far as the framework is concerned, it's just a black box function that when it sees an example, it will assign it some label, its vote, for what the label should be on that example or it can abstain if it thinks it doesn't have expertise enough to label this example. I'm anthropomorphizing here, but really, this can be as something as simple as a heuristic that says, "If I see this keyword in a sentence, label it true. Otherwise, abstain."

Basically, these labeling functions the user write become now the proxy labelers that will take your intuitions that you would have used to label examples, but now they're encoded in something that can be applied at massive scale. The user writes these labeling functions and these are executed on your large amount of unlabeled training data. Snorkel includes an algorithmic component in addition to that abstraction of a labeling function that then will automatically learn how much to trust these individual sources of labels and which ones are correlated and how to most appropriately combine them into those confidence weighted probabilistic labels. That then, as a result, gives you a large automatically generated training set that you can now train a model on to accomplish the task that you care about.

[SPONSOR MESSAGE]

[00:17:07] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:18:56] JM: To refine this, there’s a term in the Snorkel paper – There is a paper about Snorkel, by the way, if listeners want to check that out. But you mentioned the term data programming in that paper. Can you define the term data programming and explain what that means?

[00:19:13] BH: Sure. In the beginning, we first said there's a thing here. There's a thing people are doing and we want to formalize it, and we came up with theory, and we came up with methodology around it, and then we needed a name for it. So we thought, “What is it that we’re actually doing here? How is it that we are actually approaching machine learning here? How can we describe that?”

Then the realization was that we are – Train a machine learning model by focusing not so much on minor tweaks to the model architecture, but really by shaping, by sculpting a dataset, and that's what is going to be utilized by the model to solve our problem.

Data programming I'd say based on the way we've used in the papers in the past describes this general approach of taking programmatic labeling sources and combining their outputs in an unsupervised way to create a training set for yourself. But from an intuitive standpoint, I think of it just as the focus on if you want to increase performance of the machine learning model, almost always, in my experience, your time is best spent by going back to the data and seeing if the labels that are there are clean enough. If you don't have labels, do you need more, or are there ways that you can sort of, in bulk, update the labels? Maybe even heuristically with things like labeling functions to shape it to be the right – Conveying the right types of trends that your model then is going to pick up when it's trained to basically mimic what's in your training set.

[00:20:41] JM: Getting into how Snorkel works in more detail, you've alluded to this term, but I'd like to go in more detail. Tell me what a labeling function is.

[00:20:52] BH: The labeling function, you could think of as like the core primitive under the hood that Snorkel works with. Basically, where in traditional approach to machine learning, the input that a system gets from humans is individual labels. So that's individual examples get an output from you that says what you think the class is. Labeling functions just kind of go one level higher, and instead of you saying this example is true. You say, “When I see this property in an example, label it true, or when an example exists in this other database that I’ve already collected, label it true, or when a crowd worker who maybe doesn't have as much expertise as me but likely it better than random, when a crowd worker votes a certain way or says true, label it true.”

A labeling function can really be a lot of things. It can be a wrapper around existing models that you have or, as I’ve mentioned, a pattern matching sort of rule, or another heuristic that you have. But in terms of like functionally, all that matters is the labeling function is a function of some sort that when a data point comes in, it will either output a label for it with some accuracy, which may be unknown, and we’re going to loom that automatically, or it abstains and says, “This isn't in my wheelhouse. So I won't vote here.”

Then the key is that by the time we've written a number of these labeling functions, a dozen or 20 or so and combine their outputs, even if none of them individually is extremely high accuracy, you do find that by combining all these, you’re able to actually generate fairly high quality labels, but most importantly you can create them at orders of magnitude larger scale presuming that you have enough unlabeled data to apply them to.

[00:22:34] JM: Right. In each of these labeling functions, they’re defined by a subject matter expert. I mean, well, depending maybe you could have multiple labeling functions defined by a single subject matter expert. Can you give me an example of the relationship between a subject matter expert and a labeling function and how does a subject matter expert create a labeling function?

[00:23:01] BH: Yeah. Having been the effective subject matter expert a number of times back when I was doing my PhD. We had different products. We worked on different domains. We worked on where we needed to achieve high-quality. I’ve done this process quite a few times myself. Typically, what I would do in the new domain is flip through the examples and look for – Just think to myself essentially, when I go to label an example as true or false or whatever the class may be, what is it about this example that's given the evidence to me that that's how I should label this?

Maybe just to make this concrete, let's say there's a dataset of emails that I want to classify and some of them are spam. So I want to basically train a classifier to filter out these to automatically recognize spam before it hits the inbox and remove it. Some of them are not spam, or to use a colloquialism from the field, we'll call them ham.

So if I'm looking through emails and I see an email that says, "Dear Sir or Madame, I have a great opportunity for you. I would like to sell you some Viagra. It's way cheaper here than anywhere else. Act now." I think to all of this, that's pretty obvious that that's spam. But why is it? Well, there could be a number of things that tipped us off. It could be that there are lots of misspellings or grammar mistakes in it or that I see a particularly formal Dear Sir or Madame at the beginning, and that's maybe something I see more often in spam emails, or perhaps most glaringly, the fact that they're trying to sell me a Viagra, which is something that doesn't happen very often in legitimate business email.

I could just say, for this example, that spam, and move on to the next example. But I may end up seeing tens or hundreds or thousands of emails as I label that have these same properties, that Dear Sir or Madame, or act now, buy, buy, buy, or mentions of prescription drugs. Those same dozen or so clues to me that this is a spam email.

One way to teach a model that is to label enough examples over and over and over again that have that property that I hope that now my model will pick up on that, or alternatively, I could write a labeling function and say, "When you see the word Viagra or give it maybe a list of prescription drugs, when one of these is present in the email, then label it as spam." With that one input that I've given it, that one small heuristic, I may be able to automatically label thousands or tens of thousands depending on how large my dataset is automatically. Then we're able to take a lot of these. Even though there may actually be a small number of emails that were valid that mentioned a prescription drug or that had legitimate typos that were just on accident and not maybe of a more suspicious nature. By the time we've combined a number of these different signals, we're able to generate still a fairly high quality dataset for a classifier that can end up being – I mean, in the high 90s in terms of accuracy at the end of the day.

[00:25:54] JM: In practice, would a single subject matter expert define multiple labeling functions to be run over the same dataset or would you do something like you get a panel of subject matter experts and each of them define a single labeling function. How does the generation of labeling functions by subject matter experts work in practice?

[00:26:23] BH: Yeah. It’s a good question. Historically, you’d most often see – This is done like a crowd worker setting where you've got people who are not special experts on your topic, but who you think will be able to produce usable enough labels especially if you get three or five of them to input their votes and then you'll take the sort of most common class. So that's your typical crowd worker setting.

What I found though particularly powerful about Snorkel is it really does enable now very small teams, sometimes even a single person, to on their own now label an entire dataset of 100,000 documents without having to form or to get a warehouse full of people somewhere labeling to help them. I'd say the most typical setup that I find is there is one or two domain experts on a project, maybe a small team. Maybe you've got a data scientist to subject matter expert and a machine learning engineer working together or something. It really depends on the setup.

In general, I'd say all it takes is one person who knows the problem well to be able to produce these labeling functions. Because each function is relying on a different signal, you're able to get some of that independence that you'd ideally like to have when it comes to combining lots of different sources. You used to be able to get that when you're just getting labels. You get labels from different people such that if one person is relatively low quality, then they'll be sort of overruled by the others.

But now you can have a very small number, one or two experts, writing a bunch these different labeling functions, and if of those are of lower quality than the others, than the other in a labelers, which in this case is functions, no longer humans. That’s something we’ll be able to automatically pick up in the system and then down weight or remove automatically.

[00:28:08] JM: You gave the example of spam classification, and if I'm classifying spam, I can imagine a variety of text-based labeling functions. As you said, like if this email contains the word Viagra, label it as spam. If this email comes from some set of domains in Eastern Bloc countries perhaps, I want to label it as spam. Both of those labeling functions could be applied to the same sets of emails and give you a signal in both of those cases. Just take give people more examples of how these labeling functions work, could you describe how these labeling functions might work in an image classification problem, for example?

[00:29:01] BH: Yeah. That's been a really interesting area of work because it's true. When you've got a text, there is a natural sort of unit to refer to, and that's words or sometimes metadata, like you mentioned, if I've got the originating IP address of these emails. With images, there are number of different approaches we've taken in the past. One of them is actually still taking advantage of metadata. We actually had a very interesting project with some researchers at Stanford Medicine where they had X-rays, and these X-rays did not have labels for whether or not they were indicating a problem. Whether they were acute or emergent, indicative of an issue that needs additional attention or they were benign. But they did have accompanying doctors reports, which actually was a text, again.

Interestingly, they were actually able to write labeling functions over the text to apply labels to these X-rays. Transfer those labels on to the images from their corresponding text reports and then train an image classifier that only looked at now the image. It was interesting. They were actually able to train a very highly effective image classifier that took only images as input. Didn't require a doctor to take a look at it first and write notes, of course. We were able to match about eight-person months of labeling of those images with Snorkel taking only about a day of them writing labeling functions over these associated reports. That's one setting, which we sometimes call cross-modal, where you take advantage of other modalities of the data and transfer them to images. But in other cases, in other situations, that's not an option. So we do actually want ways to refer to properties of the images, and that's a bit trickier, because whereas text has this natural unit of a word, the natural unit for an image would be a pixel or something, and it's tricky to write a labeling function based on a value of individual pixels.

In practice what we do instead is we preprocess the images to add what I think of as additional hooks, additional things that we can refer to as we read our labeling functions. There are lots of great off-the-shelf open source pre-trained models for doing different things like identifying common objects or identifying blobs, essentially, segmenting pictures.

In the medical case, in a different project, we're able to make some of these primitives where we were able to automatically identify basically portions of the image that had differing properties that essentially were blob-like for lack of a better word, and they were able to comment on things like if the blob has a certain perimeter to area ratio or if it's in a particular location in the image or different things like this, using their domain expertise, they could still talk about what would be fishy or not, but they were referencing these sort of extra layers above the image. Not the pixels, but the things that we are able to detect in it and sort of make available to the annotators.

Another example for the nonmedical folks here might be there's another project where were making a visual scene graphs, basically trying to identify types of scenes or activities in images. One example would be we're looking for instances of a person riding a bike. Maybe this is for a self-driving application. We want to be very aware of where the bike riders are. A model that's been trained on image net, this is just an image classification model, can, probably with high accuracy, identify instances of people and instances of bikes. But no one's trained a model as far as I know to explicitly just recognize people riding bikes.

One thing we could do is preprocess these images with this tagger that marks with bounding boxes where all the people in bikes are and then write our labeling functions over those primitives. We could say if there is a person box and a bike box and the person box is lower than the bike box in the picture, then the person's probably not riding the bike, or else they're upside down and they're in trouble, or if the person box is twice as large as the bike box, then probably the person is in the foreground. So they're probably not riding the bike.

These types of additional layers give you things to talk about, and you can take that same approach with time series data or video data or other types of data that don't have the equivalent of keywords to refer to essentially.

[00:33:30] JM: It's powerful example. These labeling functions, the generation of data through labeling functions, it's referred to as week supervision. Can you generalize this term? What is week supervision?

[00:33:46] BH: Yeah. If you go actually on, for anyone listening, on snorkel.org, we've put up a bunch of blog posts that we've a created over the years of work on this project. Some related to particular papers, other just answering frequently asked questions. But one of those is a great summary that we wrote up about week supervision. What is it and how it relates to other topics within data labeling and supervision in general?

But to summarize that here, I'd say week supervision has to do with supervising or basically providing labels for data at a higher level. It reflects this idea that the labels you generate via a week supervision approach will likely be of slightly lower quality, say, our strength than if an expert were to look at each individual one and say, "With 100% confidence this is true and 100% confident this is false."

Instead, we're able to generate much larger amounts of labels that are strongly correlated with the true examples, but perhaps less confident. Perhaps a little bit noisier, but that's often a tradeoff we're willing to make because of the benefits we now get in terms of being able to label 10 or 100 times or a thousand times more data or being able to rapidly adapt so that when there is a change in our product needs or in our data distribution, the labeling process is now one of updating a couple of labeling functions and pushing the re-execute button rather than whipping up an updated set of guidelines for annotating and going back and touching each individual example again.

[00:35:18] JM: If I'm using a labeling function to label my dataset, have you done any benchmarks for how the accuracy compares to a human expert individually labeling each piece of data?

[00:35:36] BH: Yeah. It's a very frequent comparison that we've been interested to make, because at the end of the day, we don't want this to just be a science project. If it doesn't solve real problems, if it's not actually usable in the wild, then what's the point of doing it? That's always a focus at least for me personally of my research. I want to make sure that it's solving real problems and not just interesting. There is obviously a place for purely theoretical work as well.

Basically, in most of our papers related to Snorkel, which again you can find at snorkel.org, we've done that kind of comparison. Looking at how hard is this problem. If I had ground truth labels, we'll say, manually annotated labels. How do we compare? It obviously varies what that sort of ratio is by domain, but we found in a number of domains that we're able to actually get even higher accuracy, higher quality classifiers or models trained using the week supervision approach, and that's because each individual label may be slightly more inclined to be incorrect. But because we have so much more data for our model to learn from, it's able to learn more generally high quality boundaries for itself.

Sometimes we find ourselves getting very close to the baseline comparison of what if we had just put up the time and money to label all these ourselves one-by-one? We are happy to say we're within one point accuracy or one point F1 score for an extraction task. But other times if we have enough unlabeled data and we've actually been able to prove that we do expect the same sort of scaling that you'd get by having more labeled data if you have enough labeling functions. But if we do have enough unlabeled training needed to convert into a training set by just tossing in more raw documents or raw data points, you can often even exceed the score that you could get if you just had a set amount of labeled data.

[SPONSOR MESSAGE]

[00:37:36] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW CONTINUED]

[00:39:12] JM: In practice, the labeling functions and their results are used to build a generative model. You might have a situation, let's go back to the spam model. Let’s say we have 100,000 spam emails and I define 100 different labeling functions. Maybe I even have 100 different people who say maybe there are varying expertise in identifying spam, but I have each of these hundred people develop their labeling function. One says, “Okay, I want every email based on it where its IP address, where it originated from. I’m going to label it is spam or not. If it comes from this certain city that's well known for spamming, then I want to label this spam. Otherwise it's not spam. Maybe I have other people that are generating labeling functions based on the – Whether an email has Viagra as a string inside the email,” and you have these hundred different labeling functions. You apply it to all 10,000 emails in each case, and then you have a set of labels. You have labels that have been applied to each of those pieces of data, and then you can use all of that newly labeled, functionally-labeled data, to build a generative model. Can you explain this step? Explain that the step of taking all of this data and building a generative model.

[00:40:50] BH: Yeah. In our papers is we talk about the generative model. that's generally referring to the model that we use that learns how to combine the votes from these various noisy labeling functions and convert them into a single confidence-weighed label. In our very first iteration of data programming, we used a probabilistic graphical model here. We modeled this problem as follows. We said that each data point has some true label, which we don't know. That’s a hidden value for us. But there are things that we can observe, and that's the labels output by these labeling functions.

If we assume that each labeling function has a parameter that describes how likely it is to vote and a parameter that describes how likely it is to be correct when it votes, that these have some correlated effect with the true label, which is to say that if, in general, these aren’t just random labeling functions, but they’re on average slightly better than random, at least, then we can recover some of these unknown values by just looking at the maximum likelihood situation that would've created the labeling matrix that we’re now observing. Originally, this probabilistic graphical model was what we called our generative model that would, at the end of the day, produce a probabilistic-weighted label for each example that had one or more labeling functions voting on it.

Since that first paper, we've been able to do a number of pretty awesome additional improvements and refinements and I've actually switched to a different technique where it's more of a matrix completion- based approach. Our papers walk through that more. It's an even more kind of scalable and stable process, but the idea is the same. Ultimately, there's an algorithm here in the middle that we'll use based on what we can observe, which is how these labeling functions are voting, infer the most likely values for the things that we can't see, which is the actual "true accuracy" of each these labeling functions and the most likely confidence-weighted label for each example.

[00:42:55] JM: Once the generative model is created from these labels, what do you do with it? What can you do with that generative model?

[00:43:05] BH: Yeah. I'd say that's perhaps the most frequent question we've had over the years about Snorkel, is if I've got all these rules and if I have a way of combining them to output one label per example, then why do I need to go train another model? What do I gain from now training a discriminative model, like a logistic regression classifier, or a deep learning model?

The key I'd say, there's a number of cool benefits here in things you can do by taking this generative model and using its outputs to train another model, which we sometimes call the end model or the discriminative model. But perhaps the biggest one is generalization. These labeling functions, we allow them to feel be experts in small domains, which is to say that they don't have to each be a pseudo-complete classifier that can make a reasonable prediction on all examples. Each individual labeling function may be 80% accurate on the 5% of the dataset that it labels.

When you combine all these labeling functions, we'd obviously like to have high coverage of our training set, but in almost every case, we find that there is usually a portion of our dataset that has no labels. None of our labeling functions actually apply to that specific example. we may find that by the time we've combined our labeling functions, we only have votes for approximately 60% of our training set, but with like a model that now can make a prediction on more than just 60% of incoming emails.

This generative model was trained using the outputs of these labeling functions and the 60% of the dataset that we did have some signal on, but then we take this and train an additional model, the end model or discriminative classifier, on these examples, we can use a feature set that's more generally applicable. On the very simplest case for this spam classification example, that could just be – The features could be a bag of words, basically word counts for words in each email. But the nice thing about this now is we can generate that set of features for every email that comes in rather than for just the 60% that our labeling functions cover.

To maybe make this more in concrete or intuitive, we can even look at the simplest case. Say I just have one labeling function. My one labeling function says, "If you see the word Viagra, label this email spam." That labeling function may only apply to, let's say, 5% of my dataset. So that's very low coverage. If I tried to use that classifier alone, I would miss out on all sorts of spam emails that would make their way through, because they didn't match that very specific feature that I was able to mention in my labeling function. Maybe that's because there are other types of spam. Maybe because they're selling other types of prescription drugs or maybe because they misspelled Viagra or had a little asterisk between each letter to make it stand out or something.

Our label model, in this case, basically the combination of our labeling functions that combines their outputs, would be a pretty lousy classifier. But if I take the dataset labeled by that labeling function, that t 5% of 100,000 emails gives me a data set now of 5,000 examples that I can say are spam and I train another classifier on that, then you can imagine how this classifier, when it sees 5,000 negative examples, will observe all sorts of other correlated features. In those 5,000 emails that mention Viagra, I will almost certainly see additional instances of shifty IP addresses being used. Maybe, like you said, from Eastern Bloc countries of cities where there are known to be lots of spam attacks, or I will see other prescription drugs being mentioned, or I will see other high-frequencies of words about buying and click on this link and send money via wire transfer and things like that.

Even though I only have one input signal there that covers a very small portion of my dataset, when I transfer this knowledge to a new model via a labeled training set, it's able to pick up on all sorts of additional features including ones that I never specified in one of my labeling functions, but which do co-occur with the training set that I've now formed. That's the power that we get here, is you can have a very direct input, very interpretable inputs, these rules that I write that are very natural for me as a subject matter expert to talk about. But then the final output of the Snorkel system is not just a rule-based system with all of its susceptibleness to being brittle, because it's rule-based, but I actually get out a trained machine learning model that can handle now the long tail of examples that look similar to, but not exactly like those examples that I rule labeled.

[00:47:36] JM: Are there certain domains where the snorkel way of doing things works better than in other domains, in other certain domains, where it's just for some reason or another not workable to use Snorkel in this way of generating weekly supervised data? Are there domains where you can only have strongly supervised data and the models trained from that strongly supervised data?

[00:48:09] BH: Yeah, that's a great question. An important one is, because I think very few techniques work well everywhere. There are a number of things that we sort of require or presuppose when applying Snorkel. One of those is that you're going to have lots of unlabeled data available. If for some reason the type of data that you're collecting is extremely hard to get and you only have 500 examples, it's really not worth it for you to try and create this system for automatically labeling those 500 examples when you may be able to just as quickly label them yourself and then you've maxed out and there's nothing else to apply these labeling functions to. You might as well have as high quality labels as you can get given that you had a small dataset size.

One of them is if you don't have unlabeled data to apply these to to create an automatically generated large training set, it may be worth just manually labeling. Another thing you need to consider is do you have sources of signal to refer to? With text, often that's a no-brainer. Yes, you got access to the words themselves and there’s a lot you can do with that. We mentioned how there are workarounds for things like images and videos and things by using primitives, but there are lots of machine learning applications out there and domains where it's applied. Something you have to look for is as humans label this data, is it possible for them to ever give a reason for why they’re labeling something in a certain way? Because if so, then that can become labeling functions for them. If not or if it's very hard for them to do, then you may have a harder time expressing rules that can be applied sort of in absentia of a human looking at each individual example.

A third thing you'd want to consider is do you have models in your domain that can take advantage of large datasets? With text and image, I think the proof is in the pudding. Absolutely, the very best models today are ones that are able to learn their own features. Not for every problem, of course, but there are models that are prepared to not saturate after the first thousand examples, that if you give them 10,000 examples, they'll do even better. If you give them 100,000 examples, they'll do even better. Those are areas where it's particularly advantageous to have massive training sets, and therefore there is a bigger advantage to using a type of framework like Snorkel that allows you to magnify your ability to label training data quickly and efficiently.

I guess the final thing to consider, and this is less where Snorkel is feasible and more just where it's most advantageous. when we go looking for relevant areas to apply Snorkel to, we often look for places where it's either hard to acquire labeled training data, because subject matter experts are very rare or very expensive, or we look at places where labels have, let's say, a short shelf life.

The data distribution is ever shifting or you're in an adversarial setting where whatever your model is classified as spam, the spammers are going to come to recognize that and start trying new techniques, start trying to mask their IP addresses or start trying to use other words or start trying to misspell the words that your keyword matcher was looking for. These dynamic settings where you think you're going to need to frequently update your labels. That's a great candidate for where you would want to have a more dynamic approach rather than sort of a never-ending wheel of labeling.

[00:51:20] JM: Do you have any good examples of case studies for how Snorkel has been applied and has solved a significant real-world problem?

[00:51:31] BH: Yes, and we're really proud of all the places that Snorkel has been able to be used. Again, on the snorkel.org website, we've got a collection of all of the public-facing artifacts we've been able to produce. There have been a lot of fantastic collaborations over the years in the lab. Of course, I'm most free to talk about the ones where we've been able to put out something that's public-facing, but some of the ones that stand out for me, I was able to be a part of a project with Google. There is actually a port of the open source that was moved into Google's infrastructure called Snorkel DryBell, and with them, we were able to apply Snorkel to some very economically valuable pipelines, we'll say, and see some pretty massive wins. They found themselves in a situation where obviously Google has immense labeling budget, but there was still definite advantage to being able to automatically generate labels both for fast turnaround as well as for just the ability of combining lots of different types of signals across an organization.

In that paper, if you just search for Snorkel DryBell, you'll find it, but we're able to talk about how Snorkel was used as sort of a middleware for dumping in lots of different rich signals from across an organization. Resources that they already had, but Snorkel allowed them now to combine them all into one classifier that was supervised with the combination of these different sources and techniques, and that was able to provide, in some cases, double digit improvements over baseline systems that we compare to there. That was a very memorable one for me.

We also include on the on the website links to papers we've written with Intel, Chegg, IBM, Apple has a system as well that was inspired by Snorkel. Those are some of the companies. There're also been a number of science projects. Just to say, projects that are focused more on, you could say, scientific goods. I mentioned that Stanford Medical example.

Another product I was able to be personally involved with was Gene-Wide Association Studies. There are periodically papers that are published that say we found a significant relationship between this gene and this disease or phenotype, and there's definitely a desirable goal of having all these findings in one place and a one searchable database for everyone to go to so that same work doesn't get lost or duplicated.

We were able to spin up an automatic extraction system for being applied to hundreds of thousands of these documents and automatically classify and identify these relevant things being found. We're able to take this automated system and automatically extract a large database of these relationships, and that's now open source and publicly available as well for research or other uses.

Those are a handful of examples, but really, it's been I think over 40 or 50 organizations we've been able to work with and the applications have really covered the spread in terms of domains and types of problems and even data modalities.

[00:54:24] JM: How do you anticipate the project evolving in the near future?

[00:54:28] BH: The project began as a number of us began our PhD's at Stanford, and now most of us have graduated. So we are in the process of transitioning it out of Stanford and into a project that will be supported in different ways. It's still open source and we'll continue to always be the current library that's out there. But there's also I'd say – Aside from just the individual current project, I think there's a lot of interesting things to happen on the research side around this whole sort of paradigm of this new way of approaching machine learning where we want more signal collected in easier and higher-level ways rather than sort of that manual approach.

The lab, Chris Ray, is professor who led a lot of this research at Stanford, and a number of his students have continued on in some very interesting areas related to this. Some of those include moving what I would say higher up the stack. So rather than writing labeling functions and code, maybe we can infer what the functions should be from natural language. That was actually a project that I started that some others have continued to work on where users would just explain why they were labeling something a certain way in natural language, and we had a parser that was trained to automatically convert explanations into functions that could then be applied, which was pretty cool.

Even further up the stack, there's been some pretty neat work about observational or passive forms of supervision, where by just observing the way that a person uses a particular search interface or application, we're able to come up with relevant signals that could be used for additional supervision.

There's also some interesting work in the direction of – I sort of hinted at this before, but other types of modalities. When it comes to doing machine learning over video, for example, that's just massive amounts of data. There's no way you could have humans feasibly labeling every frame of video in a large corpus of videos. That's an area where there is potential for big win of these – Or by these somewhat automated approaches and some of the work recently published out of that group has been pretty neat in this direction.

[00:56:34] JM: Okay. Well, Braden, it's been a pleasure having you on the show, and I'm really impressed with the Snorkel project. So, best of luck.

[00:56:41] BH: Thanks a bunch. It was a pleasure to talk to you today.

[END OF INTERVIEW]

[00:56:52] JM: If you can avoid it, you don't want to manage a database, and that's why MongoDB made MongoDB Atlas, a global cloud database service that runs on AWS, GCP and Azure. You can deploy a fully-managed MongoDB database in minutes with just a few clicks or API calls and MongoDB Atlas automates deployment and updates and scaling and more so that you can focus on your application instead of taking care of your database. You can get started for free at mongodb.com/atlas, and if you're already managing a MongoDB deployment, Atlas has a live migration service so that you can migrate your database easily and with minimal downtime, then get back to what matters. Stop managing your database and start using MongoDB Atlas. Go to mongodb.com/atlas.

[END]