EPISODE 1161

[INTRODUCTION]

[00:00:00] JM: Data labeling and model training require tools to enable humans to work in the loop more effectively. The human in the loop is necessary to train models via human-labeled data. Humanloop is a platform for streamlining the tasks of the human in the loop. Raza Habib is a founder of Humanloop and joins the show to talk about NLP workflows and his work on Humanloop.

[INTERVIEW]

[00:00:30] JM: Raza, welcome to the show.

[00:00:31] RH: Thanks, Jeff. It's a pleasure to be here.

[00:00:32] JM: We're talking today about training and deploying machine learning models, specifically NLP models, natural language processing. What is difficult about training and deploying natural language processing?

[00:00:43] RH: That's a great question and gets really to the heart of what we're trying to do at Humanloop. So what I would say is that the process for deploying machine learning models today is both extraordinarily data-hungry and also extremely waterfall. And what I mean by that is that to train these modern neural network models, deep learning models and especially in the NLP space transformer based models that have really revolutionized things in recent years, often requires a very large labeled training set. So this is a dataset where you have inputs, which are often text and outputs which are labels that have been human curated. And when I say that it's extremely waterfall, what I mean is that the process of building a machine learning model today is still quite disjointed across multiple stages and often multiple teams. So typically you find the dataset that you want to train a machine learning model on and then there's a team of annotators that has to annotate it. That gets handed from them often to a data science team who maybe are not software engineers by trade but they're people who are machine learning engineers or machine learning experts. And once they've trained models, it's often a different set of people who are then responsible for deploying and handling ML ops and monitoring.

And as a result of this sort of multiple stages of handoff or multiple steps, it's difficult to update models very quickly, but also there's a huge opportunity missed, which is that actually because the data gets labeled at random or the model isn't incorporated in the annotation process, there's a huge amount of redundancy baked into the data labeling. And so what we're trying to do at Humanloop is actually make this entire process a closed iterative process instead of something that's very waterfall. And the benefits of that are many fold, but one of the big ones is that you can use the model to find the most valuable data as your annotate.

[00:02:30] JM: And tell me a little bit more about what specifically you are building at Humanloop.

[00:02:35] RH: Sure. So in the first instance, it's a platform to annotate data, train and deploy these natural language processing models. So if you were to use the Humanloop platform, you would upload your data either by the API or CSV or you can drag and drop a file. And you basically get taken to an interface where you can start annotating that data or a team of annotators can annotate that data. And whilst they are labeling it or annotating it, we train a machine learning model in parallel.

So for example, you could imagine that you have a team of people who want to classify their support tickets. So they want to automate support ticket routing. So they would load up the existing support tickets into the platform. They would have their customer support people label those tickets. And as they're labeling, the machine learning model learns in parallel. Selects the most valuable data to annotate next so that instead of labeling entirely at random, you're always labeling the things that will help the model the most. And it also gives you real-time feedback into the performance, which is something that is traditionally missing. So in the normal way to deploy machine learning models, you only really find out about the model performance after you've labeled all the data. But we actually give you some kind of metric of performance in parallel. And then once you've finished labeling or the performance is reached a standard that you're happy with, then there's a hosted API with the deployed model that's hosted on GPU that's ready to go. So you can simply take that API endpoint and integrate it into whatever process you're hoping to use the model for.

[00:04:02] JM: How does Humanloop change the way that machine learning teams act with each other?

[00:04:08] RH: This is a really good question. And first and foremost I think the goal here is to try and sort of have fewer separations of handoffs and allow a more data-centric approach to machine learning training. So instead of spending lots of time going back and forth with annotation guidelines to an annotation team potentially outsourced or if they're in-house, often expensive resource, and pinging back and forth, the hope is that actually this is a continuous process with all of these different teams working together to the annotators and the data scientists continuously rather than in a multi-stage process.

[00:04:48] JM: So you mentioned that the machine learning model training and the text annotation is parallelized. Can you talk a little bit more about that?

[00:04:57] RH: Yeah, definitely. This is really at the core of what we do. So we use a machine learning technique that's been around for a long time called active learning, but we actually use some research from myself and one of the other co-founders PhDs to try and make this scale really well to modern sized deep learning models. And the idea behind active learning is that at any point in time a model will have uncertainty about the world or uncertainty about the thing that it's being trained to do. And so you can use that uncertainty to guide the annotation process. And so the way this works in practice is you start off with a small number of annotated documents and you train a model on that partially labeled dataset. And then the model can tell you which of the unlabeled data points for which it doesn't yet have a label is going to be most valuable in terms of improving its performance. And so you prioritize the annotation from there. So the way it works in the tool is you would annotate a small number of labels, or the team would. The model would rapidly update. And then based off that update, it then prioritizes what data points should be labeled next. And what we do a little bit differently from traditional forms of active learning is that we take into account two different types of uncertainty. So when selecting data to label, you're always trying to choose the data points about which the model is unsure or from which it can learn the most, but there's really two reasons that a model can be unsure about the correct label for a given data point. So imagine I have a sentence and I want to put it into one of two categories. It could be a tweet, and I want to say what the sentiment is. The model could be uncertain, because it's never seen a sentence like that before, in which case it would be a really valuable sentence to label for the model. But the model could also be uncertain because the sentence is just inherently ambiguous, and even humans would find it hard to label that. And so what we do is we take into account model uncertainty using a technique called Bayesian Deep Learning, which allows us to find the data points which are uncertain because the model lacks knowledge, rather than uncertain because the data point is ambiguous. And doing this can give a very large reduction in the number of data points you need to label to get to a fully trained model.

[00:07:08] JM: So how does Humanloop fit in with the other tools that are available in the NLP developers workflow?

[00:07:16] RH: I think the heart of it is that we think that a lot of the models are getting gradually commoditized over time. So people are increasingly using very similar architectures. So whether it's modern transform based models or even before that, it was still quite similar in terms of word embeddings followed by some form of RNN often. And so we're trying as far as possible to make those choices automatic and abstract them away and actually change the attention of the data scientists less from selecting model architectures and tuning hyper parameters, which can largely be automated and more towards curating datasets. And in this regard, we've been really inspired by things like [inaudible 00:07:55] view of software 2.0. The idea that actually what fundamentally affects the quality of a machine learning model isn't the code that it's been built with, but most often than not is the quality of the dataset. How well labeled is it? Do I have good coverage? And the best way to improve a model once you've run it and trained it is to actually investigate where do I need more data? Or do I have data examples that were mislabeled and how do I find those? And so we are trying to make the whole process primarily data first. And in some sense it's a different way of thinking about building NLP models.

So we do interact with a lot of the traditional tools in the sense that a lot of the libraries that people are used to using spaCy, Hugging Face, Flair, we've encapsulated their models into our backend, but we actually try and choose the most appropriate model automatically and have the way that people develop these models be much more focused on the data.

[00:08:51] JM: So in the ideal world, you're taking advantage of these different NLP systems like Hugging Face or SpaCy, but you're abstracting it away so that the annotator, the human in the loop does not have to deal with that kind of development.

[00:09:06] RH: Yeah, that's right it's not purely the models from Hugging Face and spaCy, because we build on top of them to make the active learning work well. But the fundamental architectures will be very familiar to people who use those libraries. And in addition, we have a few of our own as well.

[00:09:22] JM: The name of the company is Humanloop. So what is your vision for who the human in the loop is? Like what kinds of qualifications does that person have?

[00:09:32] RH: Yeah. So in the first instance, the people using this today are typically data science teams, because those are the ones who feel the pain of data annotation and need to train models. But the longer term vision for the company is really to make training these systems so intuitive and so similar to teaching a person that actually the end user can become the domain expert themselves. So we've worked with lawyers and we've worked with teams of tax accountants currently still working with data scientists. But in the long term vision of the company, we want to make it possible for anyone to be able to automate these tasks and achieve their own goals by programming computers in this novel paradigm where instead of programming by explicitly writing down instructions one after another, you actually program by specifying a goal and providing examples of what it is that you want. In the current instance, those examples are in the form of annotations. But you can imagine that that definition could be broadened in the future. And so the mission is really to think about a new paradigm of programming and democratize access to that to many more people.

[00:10:35] JM: And what kind of applications become easier to build as the Humanloop becomes more powerful?

[00:10:42] RH: So firstly, the first big impact is we dramatically reduced the number of labels required. And so in places where the human annotator or the human in the loop would have otherwise been very expensive, this improves the ROI from those applications and suddenly makes them much more appealing. So places that we've sort of seen a lot of interest are legal firms where actually document understanding and contract understanding, building these systems and building them in a bespoke way has traditionally been very expensive, because you need the lawyers to do the annotations themselves. But if you can make it sufficiently efficient that a lawyer can give this system a small number of examples and have it learned from them, then applications where you have a lawyer or a doctor, or we've even worked with teams of tax accountants. These are applications where previously the cost of annotation itself would have been prohibitive.

The other thing that sort of we think is really important about human in the loop deployment is dealing with situations where you need a system that works 100% of the time. So machine learning as a software paradigm, one of the ways it's different, there are many ways it's different from traditional software, but one is that it never works 100% of the time. A machine learning model will not generalize to unseen data with 100% accuracy. You're very lucky to get high 90% accuracy. And so you need some way of dealing with those edge cases, the cases that the model can't handle. And this is also somewhere where actually having well-calibrated uncertainty estimates is important.

And so another form of application or the utility of having a human in the loop deployment is that in the instances where the model is uncertain, you can naturally fall back to that human who's in the loop. So you could imagine that you have some human executing labeling data or teaching the system and it automates the tasks that are easy. So if it was, for example, a tax accountant categorizing expenses, this is a task that actually is being done at some of the large accountancy firms around the world that a human expert will look at expenses and figure out their tax treatment and put them into various categories. And they can do this, teach the system. When the system is confident, it can start automating their work. But on the examples that are new or it hasn't seen before, it can fall back to the human. So deploying these systems that require some human handoff or fallback or are safety critical is much easier with the human in the loop deployment.

[00:13:02] JM: Tell me a little bit more about the engineering behind Humanloop. So it's an application that the user just loads onto the desktop and you wire in something like Zendesk tickets. I mean, you could use Zendesk as an example, like if you're doing ticketing NLP over the tickets that are coming in for a software company, how would Humanloop work for that application?

[00:13:26] RH: Sure. So you can connect the data source into Humanloop via API. And we are in fact working on things like Zapier and Zendesk integrations to make that particularly easy. And it's a web-hosted application. So you would load it up in the browser and each ticket would appear from the end user. Or if it was integrated, it could come directly from Zendesk. So the ticket would come. The customer service rep would read that and recognize the category that it needed to belong to. So for example, maybe this was a query about returns and then there's a template that they want to automatically use. So they would tag that as a returns related query. And the model would automatically learn from that and it would look at the unlabeled support tickets that sit in the queue and work out which of those are best to label next to maximize its learning.

So maybe it's learned really well about the concept of returns, but it doesn't know about late delivery, for example. So it'll go and ask you to give it more examples of those data points until it's confident. And then as that's being trained, there's an API endpoint which you can send queries to for prediction. So these could be coming for when new support tickets come in once the system is trained. You would send them to this API endpoint, and we send back the predictions for each class or the most likely class alongside their confidence. So we think that this support ticket is related to returns with 75% confidence, or 90% confidence.

Now if the confidence is very high then maybe that just triggers some kind of automated process. So perhaps you have a template that you always send, an email template, and that just gets sent automatically. But if the confidence isn't high, then the ticket gets rooted back to the annotation interface where the human can then re-label and the model will update from that.

[00:15:11] JM: And can we talk more about the engineering behind Humanloop itself? So now that we've talked through an example, what exactly is going on under the hood in the Humanloop application?

[00:15:21] RH: Yeah. So the back end is built in Python, and what happens is when you load up the data, we do a bunch of things immediately. So firstly, we just extract like are there any labeled data points? If so, we immediately train a small model on that. If not, we take you to the annotation interface. So we get you to label – So you have to label a small number of examples. So until you've labeled a seed batch, it's purely an annotation interface. But the moment you get to that first seed batch of labeled examples, then we have hosted on AWS a range of NLP models and we immediately select one of those, start training it. Train that model. The train model then sends back accuracy statistics to the frontend for the viewer to see, and it also then basically maps over all of the unlabeled data points and scores each of those unlabeled data points. Finds the top fraction based off the active learning scores that we have and then queues them up for the human to annotate next.

And the models that we have running on the backend, as I said, range from transformer-based models with sort of fairly state-of-the-art architectures down to more sort of simple deep learning models, which just have traditional, say, what are called glove vector embeddings or other traditional word embeddings and recurrent neural networks on top.

[00:16:41] JM: And is the front end of the Humanloop application, is that an electron app?

[00:16:46] RH: No. So that's been built – it's a React frontend. It’s a JavaScript with React.

[00:16:50] JM: Okay. So it's completely in the browser.

[00:16:52] RH: Yeah, completely in the browser.

[00:16:53] JM: Tell me about some of the difficult engineering problems in building Humanloop.

[00:16:57] RH: There's a lot of different challenges. I think the biggest challenge is actually on the machine learning side, is figuring out how to scale some of these active learning and Bayesian uncertainty techniques to models of the type of scale that we have today. So typical deep learning model for NLP might have millions or even billions of parameters. And if you're going to model uncertainty about them, then the way that typically works in a Bayesian paradigm is that instead of finding the single best parameter estimate for neural networks – So the normal way to train a neural network is you would optimize it with, say, a gradient descent optimize or some form, maybe Atom that kind of comes out of the box in TensorFlow or something like that. That returns you the single best estimate for the parameters and then you use those parameters for your application going forwards.

But in reality, there's actually uncertainty about what the best parameters are. Having seen a small number of examples, you can never be completely sure what the best parameters are. And so what we do is we try to model a distribution, a probability distribution over what the parameters are. So you could imagine this could be a Gaussian distribution, say, although there are many options. And if you want to do this, then actually that means you have to store immediately a huge number of more parameters. So even just storing the parameters of the probability distribution can itself be prohibitive.

And then oftentimes to do something like find the uncertainty estimates requires inverting a very, very large matrix, and all of these things become analytically and computationally intractable very quickly. And so the hardest things we've had to do are actually to come up with approximations to these techniques. So it's really infeasible to invert, say, the hessian of a neural network in any reasonable amount of time for these modern scale applications. And so we have to do numerical approximations to get that fast.

So there's been a lot of research that has gone into how can we do these Bayesian updates of these bayesian approximations in a reasonable amount of time at this enormous scale? And then the other challenge is that I mentioned that we retrain the models in between these batches of annotation. So a human or a team of human annotators annotate a few examples, the model retrains, and then we rescore the data. Now to retrain an entire neural network of this scale in its entirety could take many hours or days, and it's not really practical to have the team of human annotators sitting around waiting for the models to retrain in full.

And so what we do is we actually do a very rapid partial Bayesian update to a subset of the parameters and use that to give fast feedback to the annotators and then we run the slower retraining sort of in parallel so that there's immediate feedback coming back to the interface. And then when the model has retrained maybe after a few hours or potentially overnight, we then do the full update. And so from the perspective of the user, it always feels responsive and reactive and you're getting feedback every time you label a small number of data points. And then we do a longer retrain that might be quite time consuming less frequently, but we hide that from the end user.

[00:20:06] JM: How do you test the application?

[00:20:08] RH: Yeah. So I guess there're a few different things to test. But one of the big ones is that you're getting a reduction in data labeling. So one of the things that we do is we run tests where we compare labeling using our tool versus labeling at random and look at how the accuracy or the performance metrics of a model function as a number of labeled data points. So what we're looking to see is that we're always getting better performance. So these are offline integration tests on a range of datasets to see that we always get better performance at the same number of data points than if you were labeling at random.

But actually testing machine learning software is fundamentally different as well to testing traditional software because it is stochastic. So we have regression tests as well for an entire neural network training model. So we have a set of datasets that we train on. And when we make changes to the architecture, we make sure that we don't see performance get worse on those. And then you also have all the standard testing that you expect; unit tests and integration tests, for each of the individual modules. But what is different about it is making sure you also have those regression tests, that when you're changing architectures you're changing models. You're not making the performance of the system worse.

[00:21:19] JM: Zooming out a bit, can you tell me what was the motivation for building Humanloop in the first place?

[00:21:26] RH: Yeah. So it really came about through conversations between myself and the other co-founders all of whom have been at the intersection of academia and industry for many years. So one of my co-founders, Jordan, was previously working at Amazon Alexa, as was Emine. And we're also joined by the Professor David Barber, who's the director of the UCL AI Center in London. And we were all seeing that when we were doing projects in industry, they were often bottlenecked firstly by this need for data annotation so that they would – Projects would take a very long time because of how much annotation was required. And also they were delayed by this handoff between the teams. So we saw, there was like one project that I saw in particular which was a fairly simple NLP application for text categorization that from when the team had the idea to do it, to when they finally got the model deployed, took them almost two years. And a big part of the delays that were involved in that process were because of the disjoint communication between the various teams involved. There was a team of subject matter experts who had to be annotating the data. And they weren't themselves technical. And there was an outsourced data science team who were training the models. And they had to explain sort of how to annotate clearly to the subject matter experts. And there's a lot of nuance in these data annotations that is difficult to communicate. So they would label a dataset and iteratively go back and forth refining the instructions to the annotators. They would repeatedly train models. And it just took them an extremely long period of time to do something that to us as researchers felt like it should be extremely simple. And it felt like the tooling was making this process much harder than it needed to be. And that there was this huge opportunity to make the whole process much more intuitive and much faster, much more data-efficient. And that if you were able to do that, you would also increase the access to how many people could use these technologies, because so much of it can be done purely through annotation.

[00:23:25] JM: What's the division of labor across the team in terms of working on different areas of the application?

[00:23:30] RH: Sure. So on the engineering side, we split between – So I've been mostly working on the machine learning itself and I'm making the active learning work well. And we're getting advice and input from David and Emine who are both excellent academics with amazing research track records there. And then Jordan has a lot of experience on frontend design and UX design. So he's been really owning the development of the user experience and the React front end. And Peter has been making the whole thing stand up and handling the DevOps challenges that come with deploying these very large models and handling the load balancing and the horizontal scaling that's required as more and more models get run on the system. And then all of us have been trying to spend a lot of time speaking to customers, because we think that, really, to make this work well all of us need to understand deeply what it is that the customers care about. And so sales is still something that we all as founders want to make sure we continue doing until we really can't anymore.

[00:24:28] JM: You've talked a little bit about uncertainty. Can you talk more about the importance of uncertainty in NLP models?

[00:24:36] RH: Yeah. So the uncertainty is important for two reasons. One is post-deployment. When you have high uncertainty, it's an indication that you probably don't want to let the machine learning model handle it entirely on its own and you might want to have a human intervention or back-off to some other kind of system. And the other reason why the uncertainty is important to be aware of is because it allows you, as I said, to find the best data points to label. And that's the core of it.

I think I spoke a little bit about the two different types of uncertainty that are important here, right? And I mentioned before there's uncertainty that comes inherently from ambiguity or what we would call noise, and these are examples of data points or sentences that genuinely are difficult to categorize, like humans would actually disagree on this. And this is one of the things that makes annotating data for NLP so challenging is that actually you'd be surprised by how much different people might disagree about the annotation for a sentence. So especially for things like sentiment, say, if you ask four different people for a response, you often get four different answers. And so figuring out how to handle quality and what examples might need multiple labels and where you need to do quality assurance is made a lot easier if you have good uncertainty estimates. It's also extremely important to make the active learning work well. And then finally, it's needed post-deployment. So when you integrate with a human in the loop, you can find the places that you need to back off to the human.

[00:26:06] JM: And you also mentioned earlier that under the surface, there's all the different AI systems. So you've got Hugging Face and spaCy, and Flair, and these have their own NLP models. Tell me a little bit more about the integration between those NLP model systems and Humanloop.

[00:26:27] RH: Sure. So we essentially have a unified interface for any model. So it's a fixed interface. That means we can try and chop and change between different libraries. And so integrating those was just a question of taking the existing API interfaces that they have and matching them to ours. And then once we've done that, we then are building on top of each of these models to make the active learning work well. So it's essentially – They all end up feeding into the same model on top of them for the final few layers from our perspective. So I don't want to go into too much detail about how we make that magic work, but the core of it is essentially just making sure that from our perspective there's a uniform interface irrespective of what the model library is. So whether it's one of our own models or whether it's one that comes from these libraries, we've just unified that interface.

[00:27:17] JM: Can you tell me a little bit more about the state-of-the-art of the NLP tooling? So just how these different frameworks interact and what their different pros and cons are?

[00:27:27] RH: Sure. So if you're looking for transformer based models, these sort of highly parallel models that have taken the NLP world by storm in recent years that are almost entirely attention-based, then really the most popular library by far is Hugging Face, and it's a really well-maintained library, a huge community and has been growing in popularity enormously recently. They've got a large library of pre-trained models that users are uploading and a very responsive team.

The only real disadvantage is that it is all transformer-based models. So if you want something that's different to that environment, it's not necessarily supported there at the moment. But within the world of transformers, I think they're definitely the leading player and everything has been designed to run. It's enterprise level software. It's very performant. And they are also really fast at getting ideas from research into production. So that's been very impressive.

And then Flair, I think, is less of a production level library, but has some advantages, and that it probably still has the state-of-the-art models for certain natural language tests related to sequence labeling and named entity recognition. The maintainers or the original authors of Flair were also excellent NLP researchers who made some breakthroughs in word embeddings that take into account neighboring contexts that have proved really useful for named entity recognition. And so in those types of models, they often have advantages. And spaCy is another extremely well-maintained, extremely popular NLP library that really covers this whole spectrum. I think they also now have their own wrappers to Hugging Face transformers as well, and they have a very intuitive API. They have a lot of pre-trained models. So if you want to be rolling your own solutions, then those are all excellent options.

[00:29:21] JM: What about Snorkel? What's the role of Snorkel in NLP development?

[00:29:25] RH: Yeah. So Snorkel I think is a really interesting project, and there're two arms to Snorkel, right? There's the company and then there's the original open source project. And Snorkel is a complementary technique to active learning to what we've been talking about so far at trying to solve this problem that we discussed that training these modern machine learning models and specifically NLP models is very data-hungry. And the way Snorkel tries to approach this is they say, "Okay, oftentimes, instead of having a label, we might be able to specify a heuristic that would allow us to classify some data, but in a very inaccurate way."

So for example, we've mentioned the example of sentiment analysis a couple of times. So it's maybe one that's worth sticking to. So if I wanted to label the sentiment of a sentence, I might have a heuristic that if it contains the word bad or the word shit, that it's probably a negative sentiment sentence. Now that's not always going to be true, there are counter examples, but it might be true, say, 70% of the time. And the idea of Snorkel is that if you can – So they call these heuristics labeling functions. And the idea is that you can specify a labeling function as a rule that takes in a sentence of text or a document and then makes a best guess at the label, but isn't always going to be accurate. It's not as good as a human.

Given a collection of these rules that are not correlated with each other. So they will sometimes disagree. You can use them together to make a best guess for the label for each of the instances in your dataset. So some of the labels could come from humans. And then the rest would come from these automatically generated label rules. And given those heuristic rules, you then make a best guess for each label. And once you've got a best guess for each label and the confidence, they then train a model against that label set. So they essentially automate the labeling approximately and then train a model against those automated labels.

And I think this is a – It's a great technique especially when you have no label data at all. So it helps you overcome what's called the cold start problem where you have no labels to start with. And even active learning is not possible until you have some labels. It does have some added complexity and that now you have to manage this collection of rules, and it can grow – You can get a very large number of rules required, and it can sometimes also be difficult for the domain experts to write the rules in the domain specific language that is required by Snorkel. So those are some of the downsides. But I think it's a really good technique to overcome cold starts and is complementary to active learning. It's actually something that we're likely to incorporate into the Humanloop platform in the future, because the way we view it is that weak learning is a great way to get started, and active learning is a great way to get to a very performant model. So often in machine learning, getting to 80% accuracy or getting to some sort of decent level of accuracy is not that hard. But then the final few percentage points of performance, going from a model that is okay to a model that is very good and good enough to be deployed in production is where most of the effort is spent, and that's where active learning becomes particularly useful in helping you curate these datasets. And so the two techniques are very complementary, but useful at different stages of the pipeline.

[00:32:52] JM: Taking a bird's-eye view of the NLP space more broadly, what are the outstanding problems in NLP that you see as unsolved.

[00:33:00] RH: Oh gosh! Wow! That's a big question. And common sense reasoning is still very much unsolved. I mean, ultimately, we've made huge strides in recent years, right? Ever since the advent of deep learning and NLP more or less switched wholesale over to deep learning-based methods. There's been enormous progress in machine translation, in speech recognition in speech synthesis, in question answering, in text generation. All of these have seen big performance gains, but we're still a very long way away from a system that has any semblance of understanding in the way that a lay person would think about it. Common sense reasoning, grounding language in the real-world remains open, and there's a really interesting paper ACL this year that was debating whether or not language models are actually ever going to be able to, on their own, come to develop this kind of knowledge of the world or whether you need other contextual information.

There was a lot of excitement a month or two back around the announcement by open AI of GPT3, which is the latest language model they train. So a language model is a large NLP model that's trained in an unsupervised fashion on a huge corpus of data often just to predict either the next word or to fill in gaps in a sentence. The idea is you're trying to learn a probability distribution over language itself, and people are really excited about GPT3, because this was a huge model trained on essentially a huge subset of the Internet. And it had very interesting capabilities in terms of generalization. So it could do things like generalize four-digit addition. So having seen addition in the dataset, it was able to pick up on some idea of this concept. And when prompted a string that looks like an addition, it could complete that string with the correct answer a significant fraction of the time despite the fact that it definitely wouldn't have seen that example in the dataset. So it was actually generalizing a concept rather than just memorizing it. And it's also able to generate very long passages of coherent text that can sometimes seem even creative. So some people have shown that you can write comedy dialogues using GPT3, and that's very exciting.

But I think most NLP researchers, most AI researchers recognize that it's still only a facsimile of understanding. It doesn't as yet feel like it's able to have an understanding in the world as a lay person would understand it. Even very simple common sense concepts can make it fall over, and it takes a lot of effort to get good results out of it. So there's a huge way to go from where we are now to NLP systems that are able to read text reason and deal with them the way that a human would. But the rate of progress has been absolutely remarkable and things that felt just impossible even a few years ago at least to me now feel like they're happening all the time.

[00:35:55] JM: Are there applications that people try to apply Humanloop to where the NLP is not powerful enough yet?

[00:36:03] RH: There definitely will be. So far most of our customers have had a pretty good understanding of what the limitations of NLP are and what it's good for. So we haven't yet come up against a use case, but it's not that difficult to break NLP today. It's still early days. I think the technology is improving incredibly quickly, but it doesn't have the level of understanding that a human has. And that becomes apparent if you have very difficult question-answering tasks. You have very difficult summarization tasks, it becomes apparent pretty quickly.

And then the other place, which is maybe more mundane that NLP still has challenges today is in lower resource languages. So most of the progress has happened in English, and there's been progress as well in other European languages and languages that share a lot in common with English and also in Mandarin, but in languages that are less well-resourced in terms of how much data is available, the performance gains haven't been as good yet. So one of the more mundane challenges is just translating some of the success that has happened in the languages that the researchers typically speak, right? Because a large number of the researchers are sitting in Western Europe and the United States and making the performance equally good in other languages.

[00:37:23] JM: Are there any other areas within the domain of Humanloop that you'd like to discuss?

[00:37:29] RH: I think that the final thing to say is that we really view what we've built so far as the first step on a really long journey towards something very new and quite exciting. And we've kind of touched on this before, when you ask sort of who do we see the end users as being, right? And today the end users are almost exclusively data scientists, people who know quite a lot about machine learning and natural language processing. But I think that the way we think about deep learning is it's really a new software paradigm. It's something that's fundamentally different to how software was trained before, right? The data becomes much more central than the code. You have constant memory footprints and compute runtimes. It's not 100 accurate. It really requires a rethink in how we program computers, a whole new suite of tools to do this. And what we're most excited about is because training machine learning systems is much more about teaching by example and providing data and curating datasets, the range of people for whom this should become accessible in the long run should be much, much larger. And in much the same way that every time we've abstracted computer programming, whether it was the move from kind of machine code to assembly, or from low-level languages to high-level interpreted languages like Python, there's always been an expansion and a democratization of the number of people who are able to use these tools.

What really we want to do in a long-term vision is use this new style of programming to make what is today only accessible to software engineers and data scientists accessible to many, many more people and make it possible for domain experts to achieve their goals by working collaboration with computers themselves and by programming them through example. And that's the thing that we're really excited about. And so for us, the vision will be complete when a non-technical domain expert is able to teach a computer to achieve their goals. And this is just the first step.

[00:39:23] JM: Okay. Well, Raza, thanks for coming on the show. It's been a real pleasure talking to you.

[00:39:26] RH: Thanks, Jeff. It's been a pleasure talking to you as well.

[END]

