Bruke Kifle:

This is ACM Bytecast, a podcast series from the Association for Computing Machinery, the world's largest education and scientific computing society. We talk to researchers, practitioners, and innovators who are at the intersection of computing research and practice. They share their experiences, the lessons they've learned, and their own visions for the future of computing. I am your host, Bruke Kifle.

Machine learning is undoubtedly transforming the world we live in. Advancements in modern computing technologies paired with the generation and availability of massive quantities of data have been key to enabling the adoption of machine learning across a wide range of industries and domains. However, with massive quantities of diverse data, there is a clear need for a highly performant general distributed processing system for big data workloads that allows users to process, transform and explore big datasets. Our next guest, Dr. Matei Zaharia, has worked to achieve that and much more in the field of data management and machine learning. Dr. Matei Zaharia is the Chief Technologist and Co-Founder of Databricks as well as an Assistant Professor of Computer Science at Stanford. He started the Apache Spark project during his PhD at UC Berkeley in 2009, and has worked broadly on other widely used data and machine learning software, including MLflow, Delta Lake, and Apache Mesos. Matei's research was recognized through the 2014 ACM Doctoral Dissertation Award, an NSF Career Award, and the US Presidential Early Career Award for Scientists and Engineers. Dr. Matei Zaharia, welcome to Bytecast.

Dr. Matei Zaharia:

Thanks so much. Thanks so much for having me here, Bruke.

Bruke Kifle:

I would love to start with a question I often like to lead with, Matei. Can you tell us more about your background and some key inflection points throughout your personal, academic, and professional career that ultimately led you to the field of computing and what you do today?

Dr. Matei Zaharia:

Sure. Yes, let's see. I was born in Romania in Europe, and I grew up mostly in Canada. I went into computer science in university mostly because I liked programming and I also liked how quickly you can just try the latest techniques for everything, because you could just start everything on your computer, you don't need special equipment or anything like that. I went to the University of Waterloo, I was fortunate to work with this networking professor, Kshirasagar, who got me interested in research. I was doing research, networking, and peer-to-peer systems part-time alongside doing my undergrad. After that, I applied to PhD programs and I ended up at UC Berkeley working with Scott Shenker and [inaudible 00:02:51], again mostly on networking things initially, but I became pretty interested in large scale data center computing and frameworks like MapReduce and just all the distributed computing frameworks that were coming out as well as Cloud computing. That's what put me on the path towards Apache Spark and towards understanding these workloads, looking more at machine learning as well.

It was definitely the right time to start exploring that in the research world because these technologies went from being used at a few large web companies to pretty much every other organization in the world. It's been a great, fun kind of field to be in.

Bruke Kifle:

Oh, certainly. You mentioned, of course, the important role of professors, faculty mentors who ultimately guided your interest in research as well as this new field that you're in. You highlighted the

development of Spark at Berkeley, of course, as being a key inflection point in your journey, but can you help us understand what is Apache Spark and what are some of the motivations for its development, considering some of the existing solutions at the time with MapReduce, for instance?

Dr. Matei Zaharia:

Apache Spark is basically a framework for writing programs that are going to run on a large cluster of nodes and process data and parallel mostly. There are a bunch of different components to it, but the core of it is just this API where you can write basically single machine code in Python or in Java or in other languages just on a single machine, and you can use these functional operations like Map and Reduce and other data processing operations like joins and group PIs and so on. You can write a program using these, and then Spark will take that program and automatically parallelize it across a cluster including shipping functions that you wrote and having them run in parallel on lots of items and then giving you back the result. What it means is that anyone who's learned how to do some data processing on a single machine, say with libraries like Pandas in Python, gets a similar library for working with a big distributed collection of data across a cluster.

You can also use your favorite single node libraries as part of your program and just call them in parallel on lots of data. The goal is to make it very accessible for many different types of developers and data scientists, just like people who write programs to run something potentially at large scale. On top of this basic sort of function that the engine provides, there's also a really rich ecosystem of libraries. There are libraries on top that run a full-fledged SQL engine that can do standard kind of analytical database workloads. There's a machine learning library that gives you lots of built-in algorithms. There's an incremental stream processing system so you can write a computation and then Spark will automatically update the result as new data arrives. There are many more libraries in the community that are just built on it. It's also a nice framework for just combining these high level libraries into a bigger application.

I think compared to the tools that existed beforehand, I would say Spark was one of the first to really focus on opening up these kind of large systems beyond software engineers, so figuring out APIs that just a data scientist or someone who doesn't just write programs as their main job but has maybe a math background or domain expertise can still be successful with them. That was one difference, the focus on Python and R for example helped with that. Then, the other difference is it focused a lot on composability, both from the programming perspective, you should be able to just call things that other people hold as libraries, say a machine learning algorithm. From the efficiency perspective, Spark can do very efficient in-memory data sharing between different steps of the computation, and that's what enables things like iterative algorithms for machine learning or stream processing or interactive SQL grades. These were the differences from the previous engines that existed. A lot of the research was figuring out how to make these things work and still be fault tolerant and efficient and so on.

But once we did that, you could get this really great ecosystem of libraries on top that now users can just combine to do things.

Bruke Kifle:

I see. Certainly abstraction, composability, and of course cost efficiency. Are there other salient features that you believe has made Apache Spark dominate and become the framework of choice for big data distributed processing?

Dr. Matei Zaharia:

I think a lot of the other ones kind of stem from this and also from the great community that's contributed to it and that's formed around it. We went from basically an academic research project mostly developed by grad students to something that a lot of the major tech companies actually started using and contributing to. Over time also, other non-tech enterprises also started building on it. These have helped the system become a lot better and have helped test it at a huge scale and just make sure it works on the widest possible range of workloads and data types. I think for a lot of people today, it's just a nice ecosystem to build on if you want something that's going to be reliable, that's well supported across the industry, that connects to pretty much every data source there is out there.

But, I think a lot of the reason why people built on it were kind of these design decisions. For example, to make it cheap and efficient and also just easy from a programming standpoint to compose things, and also to make it possible for the engine to distribute and optimize the combination underneath these APIs so we can keep improving the performance of your existing job, whenever you upgrade Spark without you having to rewrite your job. A whole bunch of work has gone into that to make it declarative and make it possible to optimize things.

Bruke Kifle:

You said something earlier which was quite interesting, which is the ability to scale something that was once simply a research project into a solution that's widely adopted across the industry, by Fortune 500 and large enterprises. As you look back with origins in academia and the open source community, how has Databricks for what it is now spun out? How did the early days of Apache Spark ultimately lead to you creating and co-founding Databricks as it is today?

Dr. Matei Zaharia:

I think being able to start a company that's just working to improve this technology and then to provide commercial services around it was very important to help us all really build out the project and get it to the next level. I started working on Spark I guess in 2009. I think we released the very first alpha open source version in 2010, and that's when I was just a grad student. Over time, more students at Berkeley started collaborating on it and building things on top of it. At the time, we saw there was quite a bit of interest, mostly from more tech centered companies but also from users, like I mentioned the data scientists, the non-software engineers who still want to run large scale computation in other organizations. We saw that there is demand for something like this, and it's quite interesting to figure out how to build that well and what to do in it.

We encourage the open source community, we encourage contributions from outside, we reviewed patches, we moved the project into Apache Software Foundation as kind of a long term home that's independent of the university, so that helped. But we also realized that just by doing research, we can never... It would be hard to invest a huge amount of effort into it and just have people working full time to make the project better. We were also excited, we saw that there is enough interest to justify creating a company in this space. We didn't want a company that just does, say, support and services around Spark, so we actually started a company that just tries to provide a complete modern data platform based on what we saw working at the most tech forward companies with it, and that does it in the cloud because that was another major kind of shift happening in the industry.

But, launching the company also helped us invest more into Spark and also get it to the point where it was good enough that other companies started contributing heavily and using it in production, and it kind of grew from there. We were excited to have or to try to launch a company in this space even regardless of Spark just because we thought it's an interesting problem and everyone is going to shift the way they do data management as they move to the Cloud. But yes, it also I think helped really

cement our ability to contribute to the open source project and we could hire engineers who just work on it and help build it out.

Bruke Kifle:

As you think about some of the solutions or offerings, you mentioned some different consumers or users that could benefit, whether it be data scientists, data analysts, business analysts, those who maybe aren't software developers or don't have software background. Who are the key stakeholders that you think about when you design solutions at Databricks? When we talk about this unified analytics platform, what do we mean by unified analytics and who's the beneficiary of this?

Dr. Matei Zaharia:

Yes, great question. The way many organizations work today, both companies and things like research labs and so on, basically just organizations that work with data, they always have many different types of users who all want to do stuff with the data that they've collected. Whether it's a research lab and you've got all this information collected from experiments or telescopes or whatever it is, or it's a company and does [inaudible 00:13:14] this information about how it's operating, everyone wants to understand what's happening or to build applications that user... Say a predictive modeling application or something like that. But, these people come in with very, very different backgrounds. It's very valuable to them to have a common sort of representation of all the data, like everyone agrees on the data types you have, the tables, the schemas, all that stuff, and also a common query language or a common semantics of different operations.

For example, maybe an engineer can write a function for computing a particular thing, say about your customer, and then anyone in the organization can call that function and get that metric in an accurate way as opposed to everyone trying to implement it in a different tool. With the Spark engine, we try to have one engine that can offer these interfaces that work for different people. For example, there's the SQL interface which works for the widest range of users, including users who just connect a visual analytics tool like Tableau or Power BI. This is a tool where you drag and drop, to say to create a chart, and it actually runs SQL on the backend, or something like Excel, it can also connect that way. That's one extreme. Then, there's the sort of users who will write in a scripting type language like Python. This would be someone like a data scientist where they do programming, but they're not just doing software engineering all the time, they're trying to answer questions or prototype things. That's where the Spark Python APIs help.

Then, there's the 100% dedicated data engineer. This person is usually responsible for the most important data pipelines being reliable, things being computed correctly. If anything's broken, they'll get paged in the middle of the night to fix it and they want to use the best software engineering tools out there to be able to test things, have static types to understand what's flowing through the system, do different... Clone the job and try out different versions of it and compare the results and so on. We tried to design the data model in Spark and in the Databricks platform overall, including the other pieces like the storage component, is all the same for everyone. Everyone can agree on, "Hey, here are the tables, here are the data types, here's what they all mean." The query model is also very similar. That means someone can write a function that other people can use, which is great for handing off knowledge between teams. But then the actual interfaces, you can use these different ones that can all call into the same functions and into the same data model, but are tailored to different sort of user personas.

We work a lot with all of these to make sure they can understand these things and that they map onto this common model that actually lets the organization work in a unified way. It might seem a little bit obvious, but historically at least this hasn't been the way that most companies work with data because...

Especially before the Cloud and the on-premise world, you would buy these different systems and provision them on different servers and they would each have their own way of storing data, their own query language and so on. Then, you'd have to do a lot of work just to connect them together and to make things consistent. We do think there's an opportunity to simplify things with this one engine that can do the different types of workloads and then one data store that is in the Cloud that everything can connect to you. You don't need to deploy different data stores for different workloads.

Bruke Kifle:

I see. What are some actual compelling scenarios or use cases of this one engine, one data load that you're seeing across industries from those who are actually adopting this solution?

Dr. Matei Zaharia:

There are quite a few of them, and it's really cool just to see the range of them. At one end, there's what I call kind of classic data warehousing workloads. This is what you would do with, say a system like DARA Data for example, which is hey, you load data into some tables, you can transform it, work on it usually with SQL, and then you can query it with SQL and maybe serve it to these interactive visualization tools or compute some reports and send those to people. This is kind of simple analytics workloads where you load it and then you ask some queries about it. For that stuff, using Spark for that means you'll be able to run it at very large scale in the Cloud and you'll be able to have separated compute and storage, so while you're running one query, you don't slow down the whole system, other things can keep working and access the data. But, it also means you could potentially use the other functions like machine learning or streaming on top of the same data model.

Some people are just saying, "Hey, I want this classic stuff, but I want it in the Cloud and in an elastic way where I don't have to worry about how many CPUs I provision and how much storage." At the other extreme, a lot of organizations, virtually every Fortune 1000 and probably even beyond that company has a machine learning team now and has a data science team. They're all trying to figure out how to do predictive analytics, how to do features in the products that actually use machine learning in some way, say recommendation engines or churn prediction or predicting failures. There are some really cool use cases that we've seen there. For example, we saw a lot of the biotech companies are now developing new drugs for diseases based on analyzing large datasets and understanding what's happening in them. A lot of industrial companies have instrumented everything they put out.

For example, every tractor you purchase from John Deere now has lots of sensors on it that evaluate how it's working and can recommend when to fix pieces or can tune it for optimal performance. The same thing with every jet engine that's used in airplanes and stuff like that, they're building all these interesting applications based on it. I think the really exciting thing for me is allowing people with minimal effort to be able to do these more cutting edge applications, machine learning, streaming and so on, on top of data they're collecting in addition to just the classic applications they can do.

Bruke Kifle:

You mentioned one term, data warehouses, and obviously there's been evolutions. We've had data warehouses, data lakes, and now there's the rise of the data lakehouse paradigm for modern data management. Why the need for this new paradigm? What challenges does it address? Why aren't warehouses and lakes sufficient?

Dr. Matei Zaharia:

Yes, I'm happy to talk about it. I think there are a couple of different things that sometimes get conflated here with these. There's data warehouse systems, that's the actual software that's managing data. Then, there's also... There are these architectural terms, there is an idea of data warehouse as an architecture for managing data in an organization which says, "Hey, before you open data to lots of users, have a formal way of organizing it and defining different tables, defining relationships between them so that it's not a mess, so that you can keep it accurate over time, extend it, and make sure everyone sees correct results." Both of these things are actually being rethought in various ways. The one I talk about the most is the technology piece. Historically, if you wanted a system that can store lots of historical data and then can do fast queries on it using SQL, you build these data warehouse systems, and they were designed to be deployed on their own servers.

When you are an on-premise kind of company, you have to buy new servers to deploy a new piece of software. They all assumed that they have full control of the data and they're the only interface to that data, so they were all using basically proprietary custom storage formats, then the only way applications talk to them is through SQL, and then within that, they get really great performance. Now, when you have everything in the Cloud and when you start having applications that don't speak SQL such as machine learning or maybe streaming applications, it becomes a bit of a problem that you've got lots of data locked into something that basically only one system can query through only one interface. That's where lakehouse comes in. The other kind of model that goes under it is what's called a data lake. A data lake is basically just low cost storage where you can just put files in any format.

This is what a lot of the Hadoop and Apache Hive kind of open-source ecosystem built up. They just said, "Look, I want to manage large amounts of data without loading that into these kind of limited proprietary systems. I just want to very cheaply store it, and then I'll load subsets of it later and do more sophisticated analysis." Data lakes are just based on low-cost storage and this kind of file interface where you usually use open formats that many apps can read for your files. Then, lakehouse is this emerging trend to combine the two and to get the data warehousing-like performance and management features, things like asset transactions, on top of low-cost storage and open format, so you don't have to convert the data to a different format and move it to a different system just to get fast queries on it. That's the model that we think is going to be the future.

We saw a lot of the digital native new tech companies who had to build their stack from scratch, just build something like this from day one, they never had the separate systems. At Databricks, we decided to focus our platform around this model and to figure out how to do that well. We think there's no technical reason why data in open storage formats can't be used to provide really great performance or to provide transactions or management features or all the things people expect, so we're just trying to give people that and to just simplify their data management overall by having one system that every app can connect to.

Bruke Kifle:

I see. Combining the best of the low-cost storage of data lakes and the open formats that it allows.

Dr. Matei Zaharia:

Performance and manageability features.

Bruke Kifle:

Exactly, ease of management for those.

Dr. Matei Zaharia:

It is basically a technical problem of how to do that well, but it seems that if you can do it well, it's very useful, it just simplifies things.

Bruke Kifle:

I want to turn my attention to another project that you created and have been actively contributing to, which is MLflow. One of the most challenging aspects of productionizing machine learning is not actually training the models, but as you might predict, it's the deployment and the monitoring to actually ensure your production grade applications are as you'd expect them. As you work on MLflow, how does MLflow actually help address some of the important pain points and challenges in the ML development and deployment life cycle?

Dr. Matei Zaharia:

Basically what we've seen is at almost every company that productionizes machine learning that actually tries to use it in a product that has to run well, they end up building what's called an ML platform or sometimes ML Ops platform, basically a whole bunch of infrastructure to support the machine learning application. This does a bunch of things. First of all, it usually trains a model... It trains a model periodically, kind of automatically because data is changing. It monitors that, it gives you metrics about what's happening, and it maybe will alert if things are way off and a new model isn't doing well, or the data looks different. It versions all the artifacts, so you can hold back as you do development and see what happened in the past. It also handles deploying the model and then actually serving it. For example, a lot of the large tech companies built infrastructure like this. For example, FB Learner at Facebook and Michelangelo at Uber and many other systems, but even the non-tech enterprise companies we talked to all had something like this.

With Mlflow, we basically created an open-source project that handles this problem. It's an open-source kind of extensible ML platform project. What it does is it gives you some built-in functions for common things people need to do with machine learning, for example, packaging a model and deploying it in different serving systems, or tracking metrics about it, or sharing experiments with a team and collaborating on those. It gives you these built-in things and it also gives you this extensible framework where you can plug in new pieces. For example, if you say, "When I build a new version of my model, I wanted to pass through some custom reviewing steps, maybe an automated test and then a human that approves it that says this is actually a good model or whatever," you can plug that into MLflow in various ways to the APIs and you can build this custom workflow on top of it. That's what it does.

We see, again, a lot of teams use it as they move from just doing some experiments and creating a cool model to creating an application that's supposed to run all the time, maybe retrain periodically, and be very easy to monitor. We've tried to do something that people can use even during the experimentation phase, it saves you a little bit of time, experiment management, and collaboration, but then it puts you in a spot where you can quickly shift the model to production.

Bruke Kifle:

Can such a model or a platform be useful for detecting and managing things like concept or data shifts in your upstream data and how that might be impacting production?

Dr. Matei Zaharia:

Yes, definitely. There are some built-in features at MLflow. There's integration with Shap for explainability, but it also allows you to put in some custom processing steps for your data, for testing your model and validating it, also for any data doing models serving an inference for the results that are coming back from that. You can use it to systematically plug in things into the pipeline. I should say MLflow doesn't provide its own algorithms. We're not trying to create a better explainability algorithm or something like that, we're just giving you the programming model or programming framework where you can write your application to have these pieces and it's very easy to instrument parts of it and to observe what's happening, to automatically collect information, to show it to people, and let them plug in things that listen to that information.

It's a lot like when you run, say a web application, there are these frameworks for how to build it that will handle certain things and make it easy to hold forward and hold back, things like Ruby on Rails for example, it's a lot like that. We're not trying to provide new algorithms or anything.

Bruke Kifle:

I see. I think, as someone working in the space of deep learning and production of applications of deep learning, this is certainly an area that interests me personally. As it relates to productionizing machine learning being one of the biggest challenges, I think most practitioners can certainly attest to that.

ACM Bytecast is available on Apple Podcasts, Google Podcasts, Podbean, Spotify, Stitcher, and TuneIn. If you're enjoying this episode, please subscribe and leave us a review on your favorite platform.

Looking forward, I know there was the recent data and AI sort of Flagship Summit, Databricks announced some of the recent announcements and some of the upcoming features or capabilities that they hope to make available. What are some recent announcements that are quite exciting for you? Looking forward, what are some future directions that you look forward to?

Dr. Matei Zaharia:

Yes, definitely. There's a lot of cool stuff happening in this space. I'll just mention a couple of them that I'm personally excited about. One is on the data system side. This summer at the Data and AI Summit, as you mentioned, Databricks actually open-sourced all of this storage management layer we have developed called Delta Lake. This is what enables that kind of lakehouse pattern where you have an open format for your data and you can have which data management features on top like asset transactions, versioning, time travel, holdbacks, things like that, and also improve performance. Delta Lake's been open-source for a while, but Databricks always had some proprietary kind of enhancements for performance on top of it for some connectors to certain systems. We just saw that this went from a brand new product in 2018 with no users on it to something where I think more than 70% of data that our customers put in the platform is in Delta Lake today, so it became this essential building block.

We wanted to open-source even the advanced performance features so that more companies and more products can easily build on this, integrate it, and people can manage the data in it and not worry about, "Hey, is this only usable from Databricks or somewhere else?" I'm really excited about it because I think it's one of the best and most novel pieces of technology we've built. There's already a bunch of interesting research on these kind of systems and I think open-sourcing it is going to enable a lot more. Also, I've talked to a whole bunch of researchers who are trying to do new things to make these kind of lakehouse systems more powerful and more efficient. That's one thing. If you're someone working in the system space, I definitely encourage you to check it out.

There aren't many things that go from zero to 70% of how someone stores stuff, which is one of the more critical things you have to do with data is store it reliably, in such a short time period. The other

one I'll mention on the machine learning side, it's in MLflow. We're starting this new component to help simplify the handoff of machine learning applications between the ML experts and then the engineering teams that operate the applications. I think this is one of the toughest things with production ML, because you've got often very different user backgrounds and user types who do these different things. You've got someone that's more like a data scientist or a ML researcher who develops models, knows how to evaluate them, and maybe how to tweak them to make them better. Then, you've got a production engineer that knows about how do I monitor a thing, make sure it's working, how do I improve the performance, how do I set it up so I can operate, hold back, and deal with outages and stuff like that?

We found it's really hard to have people that do both well, and it shouldn't really have to be that way. It should be possible to have different users who focus on these aspects. We have this new component that's called MLflow Pipelines that let's basically the engineering team create a pipeline where the ML researcher fills in specific steps of it, but the whole pipeline is operable, instrumentable, and controllable by the engineering team. Basically, it's a way to modularize the code everyone is writing. As a researcher, you get the contract that... Hey, if you work within this API, your thing will immediately be productionizable, you don't have to change your code and risk any problems with that. As an engineer, you get a lot of control about how stuff is passed around and how things are tested.

Bruke Kifle:

Very interesting. Now on the announcement to open-source all of data lake, what are some of the motivations that go into open-sourcing software? More generally, what are your thoughts on the future of open-source innovation?

Dr. Matei Zaharia:

I mean, I think in general, open-source is a very powerful force in the software industry and it's something that every software development company has to keep in mind. Certainly, enterprises who buy software are very aware of it and they're very aware... Everyone wants to design an architecture in their company that's future proof. Nobody wants to... If they can avoid it, they don't want to pick something that they'll have to revamp in five years because that vendor stopped doing the things that they need or whatever, the vendor's now charging a lot of money or whatever it is, or it's just locked into one. I think everyone has to consider how to do it. We just start with Delta Lake, at first we thought, "Oh, maybe this is for a few advanced users or something like that."

But actually we realized it improves everyone's quality of life, working with these large lakehouse datasets, so much that we actually want everyone to use it. We want it to be a no-brainer decision, in terms of risk of will you use this versus a more classic data format for your data, which doesn't have the nice features, like transactions and so on. That's why we wanted to make an open-source, so that people can feel like, "Yes, this is something I can keep using decades into the future, there'll be many vendors who support it, and I don't have to worry about petabytes of data locked into one vendor." Already, there are a ton of products that connect to Delta Lake, including all the major Cloud data warehouses and all kinds of open-source engines and stuff. We're hoping to see more of that.

That's one of the things we saw as a company, building things around Spark, is we don't have to go and bug lots of companies to integrate with us and make their product work with us, there are so many products that work with open-source Spark, so many libraries, whether free or commercial products, that automatically work well on our platform thanks to the open interface. We want the same thing for the data. If you put your data in Delta Lake, you can use all the tools in the industry to collaborate on it and you don't have to worry about that architectural choice.

Bruke Kifle:

Improving access, improving adoption, and ease of extension with other tools of your liking and choice.

Dr. Matei Zaharia:

Exactly, yes.

Bruke Kifle:

Great. I'd love to turn our attention to another hat that you wear. While you aren't shaping the future of Databricks as Chief Technologist, you're actively involved in the future of computer science as an Assistant Professor at Stanford University. I'd love to learn about some of the exciting research work that you're doing. We were chatting briefly and you mentioned the DAWN project, which recently culminated. I'd love to learn more about some of the contributions of this work. I was very much moved by the mission to democratize AI by really making it dramatically easier for those to build AI powered applications. What are some of the exciting contributions that you've observed with this project over the past number of years?

Dr. Matei Zaharia:

This is a project we started five years ago, a group of faculty at Stanford. We were really interested in this problem of how to let more people, more organizations successfully use AI. We looked at it from a whole bunch of angles. For example, we had Kunle Olukotun was one of the faculty members. He works on hardware and programming interfaces and compilers among other things, so he looked at that aspect of how can we make less expensive, more efficient hardware for AI, which is a super interesting area. I looked more from the system side. Peter Bailis was another professor who looked from the database side. There were other folks as well. A bunch of interesting findings came out of it. One finding was, as you mentioned, that productionizing ML is quite difficult. For many groups, this was the bottleneck of going from a prototype to an actual application that really works and has impact.

One of the projects I worked on with Peter, for example, was a new way to debug, monitor, and improve the quality of ML models. That's called model assertions, which is a little bit like assert statements in software where you have things you expect to be true about the application, you can apply the same things to the behavior of models, and then you can actually automatically detect when they're doing things wrong and also use it to supervise the models, to train them, to make them avoid that kind of behavior. We showed some examples of that, basically working with data for autonomous vehicles and for video analytics. That's one takeaway. Another interesting takeaway was that in some areas of AI, getting labeled data is actually the bottleneck. It doesn't matter how expensive it is to design your model, if at all, or to train it, actually getting labeled data is hard.

Professor Chris Re, another one of the PIs, had a whole line of work on minimizing the amount of human labeling you need and using weak supervision, which is basically using automated rules that guess at the label but may not be fully accurate, and learning from those. He's had a lot of success with that in quite a few domains where you can write these generic rules and hand them over a collection of, say legal documents or medical papers or stuff like that, and actually get a pretty good model. The challenge is how do you... In fact, you can do better than people who use just a label dataset with less effort without having to have people label millions of documents or images. These were a couple of the interesting themes, but for me the best part about it was just seeing people thinking about this problem from all angles, getting them all in a group together to talk about it, and learning across these. A lot of our work

kind of ended up mixing insights from the different areas that I think wouldn't have happened as easily without this group.

Bruke Kifle:

Certainly. That's very interesting. Another line of research that actually excites me is some of the work on retrieval based NLP. Undoubtedly, we've seen a lot of the great promise of large scale language models, but there are also very clear limitations around high training costs, high inferencing costs, the true feasibility of actually productionizing these large models, there's explainability issues, models are static, so there's freshness issues. How does this sort of approach of retrieval based NLP work? How does it address some of these fundamental issues that we observe with trying to make value out of these large scale language models that exist today?

Dr. Matei Zaharia:

Yes, this is the research area I'm probably most excited about in my group right now. Basically, so far we've gotten some really great results in NLP with these giant models that have lots of weights, something like GPD3. The idea with these models is you have a collection of knowledge, you have a bunch of documents from the web, and you train the model over it and it incorporates that knowledge into the weights. Then, when you do predictions, it can do stuff. It knows that the capital of France is Paris, it knows that the president of the US is Joe Biden, whatever. It has all this knowledge that appeared in those documents and it can use that in various tasks. But if you think about it, these are very expensive to train, they're also very expensive to do inference with, and they're very hard to update because if something changes, like after the next election, the president of the US changes, you got to retrain this whole model from scratch to give that.

You actually see this if you use GPD3 today. Well, I don't know about today specifically, but definitely when we tried it a while back, it was returning the previous president of the US, it didn't know that this changed, so it's a problem. The retrieval oriented approach is that instead separate the knowledge, you have the documents and you have some neural networks, but then when you're given a task, say you're being asked to answer a question, you search over the collection of knowledge somehow and then you read those documents, you pass the documents that you retrieved along with some context like the question and other information about your task into a smaller neural network and you produce the answer. The nice thing about that is you can always update the knowledge because you can just change these documents. You also get quite a bit more interpretability, you get a sense of, "Oh, why is it giving this answer? It's because it's right in this document and maybe that was confusing." It turns out to be a lot cheaper.

It depends on the task. These models can't do everything right now, but for some tasks such as question answering, these models that just... They're orders of magnitude cheaper than the large language models and they're much higher quality in terms of answers. There's a lot of work in this space. There's now, for example, there are people using retrieval for language modeling, which is a very general use case, the retro model does that. There are people using them for images as well as texts, so retrieving images and texts together and having interpretable results there. There are people using them for more sophisticated applications. One of the ones we built can answer questions that require looking up many documents all at once, not just one, and it seeks out new knowledge until it has enough to answer the question, but basically looking at concepts that came up in these.

Bruke Kifle:

This still requires some index to actually do the retrieval from, right?

Dr. Matei Zaharia:

It does, and you have to...

Bruke Kifle:

I saw an interesting analogy of this retrieval based NLP as being an open book exam, so there's still the need for...

Dr. Matei Zaharia:

Yes, it's an open book. Actually, a lot of the work is on... That indexing itself and that search is done using a neural network also by maybe embedding the documents into some kind of vector space and then searching for nearby vectors.

Bruke Kifle:

I see.

Dr. Matei Zaharia:

A lot of the work is on how to do that better also, which then immediately improves these. Or, how to co-train the indexing and look up together with, say the question answering, so that they're tailored for each other.

Bruke Kifle:

Very interesting. You said there's already some line of work thinking about image search. Are there areas of investigation around multimodal information retrieval, image to text, text to image?

Dr. Matei Zaharia:

There's a little bit.

Bruke Kifle:

Are there other areas where you see this potentially being applied?

Dr. Matei Zaharia:

There's a little bit. There's not a ton yet, but there is some work with images. I think it could be useful in other areas too. One example actually that I'm curious about would be reinforcement learning, because if you think about it, you have this history of training exercises you did for your model. Again, even though you ran all those, you then just condensed everything into a bunch of weights. But what if you could look up what are past training situations that look similar to this, what did I do, and what was the outcome? Maybe you'd be able to improve performance there. My group's focused most on the NLP use cases because there are so many of those and they're very easy to interact with, but I think it could be useful in other places too.

If you think about it, again, from the production ML perspective, if you're going to productionize one of these models, you want to be able to interpret what it's doing and also to fix it if something is wrong, and you want it to be fast also, to be fast enough to actually run. This helps with that, but it also gives you these nice ways to see, "Wait, why is it making that prediction?" If I want to stop it from doing this, what do I change? Here, you can just change the documents it's pulling out that are misleading it.

Bruke Kifle:

Right. To turn onto another responsibility that you have in academia as a professor and as an instructor, what are some of the biggest gaps or opportunities that you see in computing education? I know certainly, going through school myself, I know there was a big interest in recent years in the field of machine learning. But ultimately to develop and deploy capabilities, it goes beyond model training. What other areas do you feel that folks could emphasize in computing education to really ensure that computer science graduates are well equipped with all the capabilities to develop and deploy end-to-end systems?

Dr. Matei Zaharia:

Great question. I actually think one of the biggest shifts that most universities are missing is the shift to software as a service or basically software being delivered through the Cloud somehow. This is happening everywhere, whether it's just a user-facing kind of productivity app like Google Drive or Salesforce, or it's actually the platform. You can get a database as a service from Google or from Microsoft or you can get machine learning, whatever, predictive... [inaudible 00:47:32] predictions as a service from Amazon and so on. The thing is, all of these are now being delivered continuously. My experience and the same with I think everyone who does it is that building a production Cloud service and maintaining it is really hard. That's what we see at Databricks, and that's what many of the companies we work with see. There isn't really any education on it.

I think it's more than engineering. Also, I think it could require new programming models that are going to work well. It could require new designs of systems, for example, how do I make my systems so I can easily roll out a new version of my app and then roll back to an old one? Or, how do I make it so it can isolate requests from different tenants and just guarantee that no tenant can interfere with the performance of another tenant too much or stuff like that? I think this is a super interesting area. I would love to have a class where you teach about these things, but also a class where instead of students turning in an assignment every couple of weeks with a little programming thing that we run some tests on, they actually deploy a service on day one and then it has to keep operating and serving requests throughout the semester. Then, they have to put in a new feature, implement, I don't know, pagination, implement whatever GDPR compliance or whatever without corrupting the data or otherwise breaking it, because that's what they'll have to do in a real job.

Bruke Kifle:

Well, it seems like you have the syllabus for your next course all prepared.

Dr. Matei Zaharia:

Yes, potentially.

Bruke Kifle:

Yes, but I would say there's certainly a huge divide between how folks are trained in university and the reality of how things operate in industry and in production. I think oftentimes internships or hands-on experiential project based learning becomes the best avenue to do that. I think this is certainly a very noble model of learning that could certainly benefit many folks as they make the transition and look to actually develop end-to-end systems. To wrap it up, I'd love to touch on two things. One is we talked about a lot of your work with Apache Spark and Databricks, but also your responsibilities as a researcher, as an advisor, as a professor. These are two very difficult jobs to manage. How do you juggle

your work in industry as a practitioner, as a Chief Technologist at Databricks with your role in academia as a researcher, as a professor, as an advisor? Do you find these two worlds colliding? Do you find them very different? In one way, does one role influence the other? Is it your industry experiences are really influencing your role in research, or is it your research experiences that influence your work at Databricks?

Dr. Matei Zaharia:

Yes, I think the roles are definitely different, there are different concerns in each one. I do think a nice thing about a faculty job is it does give you flexibility to work with companies in various ways and that is kind of part of the job, that's just the way it works. But, they are super different. I do learn a lot of stuff in both that influences what I do. Mostly, I think it's been seeing stuff in industry that I think is a big problem that isn't really studied and researched, and then thinking about it from a research perspective. I've tried to keep them fairly separate in most cases because I don't want to have some kind of conflict of interest or students feel they're working on something that benefits Databricks or whatever. But there are often just long term things like this thing with everyone in industry is writing services, but there are hundreds of papers each year on debugging and stuff like that, I don't consider that's an insight that can lead to research projects that kind of benefit the whole industry.

But things that are very specific, usually I would do that work only in the company.

Bruke Kifle:

I see.

Dr. Matei Zaharia:

Certainly within the company it helps just to see all the perspectives you see at the university and say the future of hardware or ML models or stuff like that. But, they're both interesting. I think the big difference is in industry, you often get a lot more resources behind a particular thing, and it's very hard to match that in academia. You can't hire tens or hundreds of engineers to build a thing and then to maintain it. Of course in academia, you get to explore a lot of stuff and if something doesn't work, you can just switch to something else and so on. It's very flexible that way, and you get to teach people in a different way as well, to mentor people.

Bruke Kifle:

Very interesting. To wrap it up, I would love to provide you the opportunity to share any parting remarks. But to provide some structure, what are some exciting future directions that you see for the field of data management, machine learning or computing at large? What are some of the exciting areas that you hope to see some of the greatest promise in?

Dr. Matei Zaharia:

I mean, I think this is a great time to look at both machine learning and computer systems. I would say, I mean I think everyone is very excited about the potential of these large models and deep learning workloads and so on. I think it's super interesting, but I would also say if I were to give advice to students or people getting into the field, I would also say to take a look at the computer system stuff because there is this huge change to Cloud computing and these systems ultimately underlie a lot of what's happening in other places. There'll be a lot of demand for engineers, researchers, and so on that know about this stuff. Even the large language model stuff, I think in many places it's become basically a

systems problem of how do we scale out things more, how do we do it more cheaply and so on to train more models?

It's become less of a modeling problem. Of course, there are a lot of improvements you can do there too. But, the point is that if you can do that stuff well, it'll have a lot of impact on real applications that are out there.

Bruke Kifle:

Awesome. Well, it's been a pleasure speaking with you, Dr. Matei Zaharia. Thanks so much for joining us at ACM Bytecast.

Dr. Matei Zaharia:

Thanks so much for having me.

Bruke Kifle:

ACM Bytecast is a production of the Association for Computing Machinery's Practitioner Board. To learn more about ACM and its activities, visit acm.org. For more information about this and other episodes, please visit our website at learning.acm.org/bytecast. That's learning.acm.org/bytecast.