EPISODE 810

[INTRODUCTION]

[00:00:00] JM: Datasets can be modeled in a rowwise relational format. When two datasets share a common field, those datasets can be combined in a procedure called a join. A join combines the data of two datasets into one dataset that is often bigger than the initial two datasets independently occupied.

In fact, this new dataset is often so much bigger that it creates problems for the machine learning engineers. Arun Kumar is an assistant professor at UC San Diego. He joins the show to discuss the modern lifecycle of machine learning models and the gaps in the tooling.

Arun's research into improving processing of joined datasets has been adapted by companies such as Google. Some of that research has been adapted into open source machine learning tools that improve the performance of machine learning jobs with minimal new code required. Arun is a true expert in the contemporary field of machine learning and it was a great pleasure to talk with him.

Some recent updates to Software Engineering Daily land, a new version of Software Daily has been deployed to softwaredaily.com. It's got a lot of new clean features and we'll be eventually rolling these out to the apps. I know many of you have been using the apps to get the ad-free episodes and you have been enduring the pains of our not so great mobile apps relative to your familiar podcast player. We know that there are some performance issues. We're going to get those ironed out.

In the meantime, if you can report any of those performance issues to our FindCollabs thread, which is also in the show notes for this episode, just report any problems. We're really trying to get the Android app and the iOS app cleaned up so that we have a place that's actually pleasurable to listen to Software Engineering Daily and have comments and have discussions and kind of a place for our community to collaborate with each other.

Speaking of collaboration, the FindCollabs $5,000 hackathon ends this Saturday, April 15th, 2019. So there's still time to post your projects and find collaborators, and it's far from a finished race. There's plenty of time to submit your project, and it can be an old project that you're just looking to get some mileage and find some new people to work on with. It can be a totally new project. It can just be a set of ideas, wireframe. It's pretty open-ended hackathon, and all those details are in the show notes for this episode.

With that, let's get on to today's show.

[SPONSOR MESSAGE]

[00:02:40] JM: As a software engineer, chances are you've crossed paths with MongoDB at some point, whether you're building an app for millions of users, or just figuring out a side business. As the most popular non-relational database, MongoDB is intuitive and incredibly easy for development teams to use. Now with MongoDB Atlas, you can take advantage of MongoDB's flexible document data model as a fully automated cloud service. MongoDB Atlas handles all of the costly database operations and administration tasks that you'd rather not spend time on, like security, and high availability, and data recovery, and monitoring, and elastic scaling.

Try MongoDB Atlas today for free by going to mongodb.com/se to learn more. Go to mongodb.com/se and you can learn more about MongoDB Atlas as well as support Software Engineering Daily by checking out by the new MongoDB Atlas serverless solution for MongoDB. That's mongodb.com/se.

Thank you to MongoDB for being a sponsor.

[INTERVIEW]

[00:04:02] JM: Arun Kumar, you are an assistant professor at UC San Diego. Welcome to Software Engineering Daily.

[00:04:07] AK: Thank you for having me, Jeff. Glad to be here.

[00:04:10] JM: Your research focuses on the machine learning lifecycle of a software product. Describe the machine learning lifecycle of software as you see it today.

[00:04:21] AK: So the lifecycle of machine learning powered software depends on the context in which the machine learning techniques are being used. People start using machine learning in all sorts of settings, like classical database resident data, data warehouse resident data, you want to do, say, churn predictions, sales forecasting and so on. In emerging web companies, you want to do kind of click through rate prediction, computational advertising. So the data environment there is different. In, say, computational journalism, you want to rank stories and you want to prioritize what the journalist want to access. Those sorts of settings are different.

But overall, across these very diverse set of environments, where machine learning techniques are powering software, you can abstract out kind of three main parts of the lifecycle of the machine learning powered software. The first – So I call these the source, build deploy, three stages of the lifecycle. The sourcing stage is where you go from the raw data in your data-driven application to the form of the data that you want for machine learning algorithms to start consuming, and this involves a lot of steps; data integration, cleaning, preparation, feature extraction and then schematization, identifying the features, all of that. That often ends up being a huge bottleneck for data practitioners. But once you have that, then comes the process of building the model.

Now, in the building stage, you have to apply a lot of ML techniques for things like feature selection, hyper-parameter tuning, algorithm selection. Often that happens in a human in the loop fashion. So that's the model building phase. You have to worry about concerns of accuracy, training efficiency, interpretability and all of that. Once you've done that, you obtain a prediction function that you can integrate with your applications. Things like, "Okay. On a future customer, I'm going to deploy this model. They are just likely to churn. On this particular user that is logged in to my website, I'm going to deploy my model. They're going to click this ad with this probability." Something like that.

So the third stage is just deployment. Once you have gotten your train prediction functions, integrating them into your application. Because your application space is so diverse, like I said, it could be data warehouse resident data, they could be web companies, they could be journalists and all of these. It entirely boils down to what is the application environment into which the machine learning model is being integrated into. But you'll see a lot of challenges there for kind of monitoring, making sure that the machine learning model is behaving as you'd expect. Tracking the predictions and feeding that back into the first part of the lifecycle, which is sourcing, because the data and the application are seldom static. They keep evolving overtime. So you cannot just use the same model forever. You have to be in the loop and make sure that is refreshed and it's kept up-to-date.

[00:06:57] JM: Source, build and deploy. So those are the three phases of the machine learning lifecycle as you see it. As you alluded to, the applications that we could potentially build within this lifecycle are quite diverse. There's also really diverse tooling that you alluded to. Not only do we have machine learning frameworks. We've got databases, data lakes, data warehouses, streaming frameworks. Describe the tooling landscape as you see it today.

[00:07:27] AK: So in terms of the tooling, because of the diversity of the kinds of applications and the diversity of the kinds of datasets people want to use, I would categorize the tools for machine learning systems as like maybe five groups of tools. The first and probably the most widely used are the in-memory frameworks. Things like Python, and R, and there's a huge number of libraries in both of these programming environments, scikit-learn, Matplotlib, seaborn, Pandas and what not in Python. Then R has this whole community called CRAN, hundreds of libraries contributed by various professionals.

The data sourcing part there is typically outsourced by people who use these sorts of tools to what they call data engineers, and these are people who use databases, data warehouses, traditional SQL and user-defined function style programing. But increasingly, these two roles, the data scientist role and the data engineer role, are being merged and they're being called ML engineers.

I was just amused yesterday when Ben mentioned that ML engineer has emerged as the second most preferred title for people in this space, which it didn't happen like three years ago or something like that. This is what Ben pointed out from the survey yesterday at the keynote.

But, yeah, I've known this for like three, four years now. ML engineer has been the title that's been around.

The first, like I said, in-memory tools. The second one is in certain applications, if your datasets are much larger than main memory. Like if you have a single node and you have, say, 32 gig RAM, but your dataset is a terabyte. Then in the past what people would do, people primarily in the enterprise settings who used tools like SaaS or SPSS would downsample the data and then load it into memory.

Downsampling is inherently reducing the value proposition of the data, because you spend all these money to collect so called big data and then you're throwing away most of that data. So what a lot of organizations has started doing is building new ML infrastructure that scales to the amount of data that they have. This started like a decade and a half ago, what is called in-database analytics. The focus was the database community realized that to do data analysis, it's not just SQL style OLAP or business intelligence queries, but rather statistical and machine learning computations need to be run on database resident data.

So Microsoft, Oracle, they all came out with these data mining toolkits that scale machine learning algorithms to data that is actually stored in the database. So now your machine learning algorithm is transformed into a sequence of queries over the database. How do you rewrite a machine learning algorithm's implementation, or this database resident data was a focus of research and there's a lot of tools that came out. I myself was involved in one popular open source library for this called Apache MADlib. It came out of EMC Pivotal. That was one aspect.

In the late 2000s and early 2010, people started realizing that using your relational DBMS's for time consuming machine learning computations was kind of wasting their licenses. They were paying for these RDBMS's. So Hadoop MapReduce became a big deal. HDFS became a big deal, and people no longer feared kind of exporting their data and copying their data to HDFS.

In the past people said, "Okay. Data has governance, compliance requirements. You should keep them in their RDBMS." But now with HDFS, things like Cloudera and all these companies came along. You could manage your HDFS clusters within your compliance boundaries. So they were no longer worried about copying the datasets.

Once you copy the datasets, now you have this read only analytics environment, like Hadoop MapReduce, and people started using tools like Mahout, which his basically a machine learning library that's implemented using MapReduce as the functional programming abstraction. More recently, Spark MLlib and other tools in the HDFS ecosystem. That sort of is the evolution of the tooling for scalable machine learning started in DBMS analytics, then Mahout, and then MLlib and other sort of HDFS-based ecosystem.

More recently, in the last three, four years, unstructured data has become a big deal. Even though a large majority of enterprise use cases focus on structured relational data. Text, images, audio, video, time series, these are also becoming important in many domains, and deep learning is the way to go if you want to process unstructured data. But many of these ML libraries and scalable ML systems that were built in the past were not tailored to building computational graphs of like very large kind of neural networks. That's where TensorFlow, PyTorch, NDK, MSnet, all these sets of tools started gaining popularity.

TensorFlow and all these systems are kind of standalone stacks. They have their own processing stack. They have their own processing kind of runtime. So that's another category of its own or what we call deep learning systems. These days people are starting to integrate these deep learning systems into HDFS-based ecosystems as well, Spark and TensorFlow, TensorFlow with something else.

Then the last category is what I call miscellaneous. These are like custom ML systems that are built as an end to end stack from top to bottom that do not kind of integrate with most of these. Like things like [inaudible 00:12:16] Boost for boost decision trees. That's very popular, especially on structured data. Then there's DeepDive, which is a statistical relational learning system that came out of Wisconsin and then Stanford. So these sets of tools are specific programming models for ML, not general purpose systems. That's what's also the fifth category.

Now, in the cloud, all these companies like Amazon, Google, Microsoft, they're launching their own APIs for easing the training process and also the deployment and the inference process.

How exactly they are building them in the backend, it will be one of these sorts of approaches. They'll either build their custom stacks or integrate them into their existing stacks, but they just expose the APIs so that now if you're a startup, you don't have to go and implement your own ML system. You can just call APIs available in the cloud services, or use one of these open source scalable machine learning toolkits and integrate them into your platform. So that's sort of a whirlwind tour of the evolution of the landscape of tools and systems for machine learning-based analytics.

[00:13:16] JM: There's such a bountiful array of problems that you could explore in this space, and you have focused on the area of joins and machine learning over joins. The problem, as I understand it from your work, is when you join two datasets, two tables, you're often joining on a foreign key, and when you perform that join, you often have an explosion of the number of objects that you're looking at. We'll get into this discussion in more detail if people don't really understand what a join is or why it blows up your data. But why of all the problems that you could focus on in this diverse space, why are you focused on ML over joins?

[00:14:05] AK: Yeah. That's a good point. So there is a lot of problems in sort of this whole landscape. You could focus on, say, scaling some ML algorithms and kind of building distributed implementations of it. You could focus on applying ML algorithms to various domains. There's a lot of exciting work that's going on there. But my own background is in database systems. That's what I did my PhD on. It was at the intersection of database systems and machine learning computations.

So interest was in kind of data-driven computations and building systems for data analytics, and machine learning was just the natural evolution of the kinds of analytics people do. Building systems is what excites me the most, but I want these systems to be principled to understand the theory behind them. To understand the data model, the data flow and the mathematics and the statistics behind what is going on.

The other direction of kind of applying ML to various applications, there, the focus is on understanding the application. Then you have to kind of characterize what the features are, what the business metrics are or what the scientific insights are. That is just very diverse.

Somehow, that doesn’t excite me as much as the building systems part. So that’s why I focus on data management systems for machine learning primarily.

One other reason could be I’m an academic. I like kind of identifying research gaps that are kind of gaps in our knowledge of the landscape of computing, and this is one of the most important research gaps. What are the principles of machine learning systems? What are the foundations of kind of building these sorts of systems that are, one, user-friendly, that are scalable, that are kind of easy to build, easy to develop.

People have studied such issues of usability, scalability, what I call developability and manageability. In the context of relational data management and SQL analytics for decades, but what does it mean in the context of machine learning analytics? That’s this huge open area, and I think there’s like a lot of open research questions in that area.

One of my main technical focus if you’d like to call it is query optimization for ML systems. So the learning over joins is a form of query optimization, like relational DBMS is one of the celebrated results in query optimization, is pushing selections, rejections, aggregates through joins.

The beauty of relational DBMSs has been called introduce relational algebra. People no longer needed to write very low-levels, kind of C++ code, chase pointers to get records. They write logic-based one line statements, SQL statements, or relational algebra statements, or dataflow language like Spark or Pandas. You write functional compositions. You don’t write low-level point adjacent code.

But under the covers, a middleware system called the optimizer, translates what you’ve written into a very efficient execution on the runtime. What does that mean in the context of machine learning analytics? That is the primary focus of the technical aspects of my work. As ML over joins turned out to be one of the key open areas in bridging this gap between data systems understanding and what machine learning computations do, that I was just floored that people really haven’t tackled before.

Because in my research, I like interacting with practitioners, like software engineers, ML engineers, data scientists. Even during my PhD days, I'm interacting with all sorts of people who use ML for analysis to understand what are the bottlenecks they're facing. I've spoken to like English professors, law professor. I've spoken to consultants in Deloitte, and Amazon, and insurance people at AmFam, a lot of really awesome people that I learned a lot of from.

Based on all of that, I understood that this data sourcing and feature selection part was a huge open bottleneck back then, because when you're operating on structured data, they have to interact with the data and understand how do you express the data to the ML algorithm itself. In that process, we kind of abstracted that out as a declarative dataflow process and created a tool called Columbus, that basically think of that as a declarative dataflow language for specifying feature selection operations. Now you can say, "I don't want these set of features. I want to drop these features, add those features." These sort of what they called dialogue with the algorithm.

So we formalized that process, and the reason that was different was in the past, like the data mining and machine learning community treated this purely as an algorithmic problem. You just throw all your features at the model, let it automatically figure out what it wants. The reason that was not satisfactory for many of these enterprise users was these features mean something. They come from different sources. They have different monetary cost. Not all features are made equal for them.

So that formalization led us to this system called Columbus, where we elevated the specification of the feature selection process to a declarative level, and under the covers our system would product optimized R code, or Python code. So that's sort of a multi-query optimization for the feature selection process. That paper sort of won the best paper award at ACM Sigmod in 2014. It's a top data management conference.

So then I realized that the community was really excited about this direction. I said, "Okay. There are other sorts of bottlenecks in this whole feature engineering process." One of those bottlenecks was one of the operators we supported in our Columbus system, was joins. Often, your structured data is spread across multiple tables. This is almost universal. Any company that manages structured data typically stores them as a normalized database. It's like databases 101. Recommendations, like ratings, users, movies. Insurance, you have customers, employers, area, weather and whatnot.

So there's this huge gap between how the data systems treat data and how ML systems treat data. ML systems, almost universally, expect the training data to be a single table. So this was this start gap that I noticed. The reason this gap arose historically was the separation of, "Oh! The ML is going to be handled by someone from a statistics background in a tool like SaaS or R," and the contract is that the data engineers who do all the data preparation and then build this massive single table to the data scientist. That separation of roles was starting to disappear. That's why this whole field of data science started emerging.

The statistician and the data engineers are no longer two separate people. One person gets to see the whole picture from the raw data to actually producing the insights. Then I realized that this is an opportunity that whole landscape of multi-table data and ML computations are putting on large amounts of data. If we put these two together, we can get even an order of magnitude gains and efficiency. In some cases, even two orders of magnitude. You could reduce the runtime by 100 times. You could improve the usability of the process. You could kind of reduce the burden on the data scientist who's doing the analysis. That's where this whole exploration of ML over joins started. That was in the focus of my dissertation work.

More recently, kind of generalize this whole space of data preparation and model selection for model building. What the source kind of build aspect has been the key focus of my research. Yeah, the ML over joins is what I'm going to be talking about today, later today, primarily because I have interacted with a lot of companies and they found this idea useful, and they actually use that in practice.

So some of them said, "Hey, you should talk about to more people. Maybe they will also find it useful." So that's why I came to Strata to spread the word among practitioners, "Hey, look. These are two interesting ideas that some people have found useful. Let me know if you have use cases that you could benefit from."

[SPONSOR MESSAGE]

[00:21:21] JM: Testing a mobile app is not easy. I know this from experience working on the SEDaily mobile application. We have an iOS client and an Android client and we get bug reports all the time from users that are on operating systems that we did not test. People have old iPhones. There are a thousand different versions of Android. With such a fragmented ecosystem, it's easy for a bug to occur in a system that you didn't test.

Bitbar is a platform for mobile app testing. If you've struggled to get to continuous delivery in your mobile application, check out bitbar.com/sedaily and get a free month of mobile app testing. Bitbar tests your app on real devices, no emulators, no virtual environments. Bitbar has real Android and iOS devices, and the Bitbar testing tools integrate with Jenkins, Travis CI and other continuous integration tools.

Check out bitbar.com/sedaily and get a free month of unlimited mobile app testing. Bitbar also has an automated test bot, which is great for exploratory testing without writing a single line of code. You have a mobile app that your customers depend on, and you need to test the target devices before your application updates rollout.

Go to bitbar.com/sedaily and find out more about Bitbar. You get a free month of mobile application testing on real devices, and that's pretty useful. So you can get that deal by going to bitbar.com/sedaily. Get real testing on real devices. Get help from the automated test bots that you have some exploratory testing without writing any code, and thanks to Bitbar.

You can check out bitbar.com/sedaily to get that free month and to support Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:23:31] JM: Tell me if I understand this correctly, but an abstract level it seems like in a database application, you want to break up your data into multiple tables because you want to spread out the data so that it's more efficient to access. It's put in kind of a separation of concerns, style situation. But in machine learning, you often want your data points to be represented in a super high-dimensional space, because you want to be able to do like feature generation, or like you want your data represented in this multi-dimensional vector space with as many dimensions as possible. It seems like there's a fundamental tension between those two data representations.

[00:24:13] AK: Exactly, and that gets to the root of why this whole space has a lot of open research questions from the standpoint of what is a programmability model? What is the computational model? What should the runtime look like? Because, like you said, in the ML world, the thinking always goes, "I start with a feature vector and then I apply my ML model to it," but 99% of the time goes into getting that feature vector in the first place. On structured data, that involves these sorts of operations.

But increasingly, like you pointed out, the silo-wide station of different data stacks is antithetical to this promise of big data and the promise of using ML on big data to derive new business insights or business outcomes or scientific insights. The existing tooling infrastructure that we have, the systems infrastructure that we have, there's just this huge gap. Because like I said, the landscape of ML systems, the five things that I outlined, in-memory tools, scalable ML tools, deep learning tools, none of them address this existing problem. This gap between what the data looks like and what the ML computations do.

I think bridging that gap would require new foundational principles and systems from the data management and the systems community to build this. Coincidentally, there was this recent conference that was started, SysML, that I'm actually going to after the Strata Conference, systems for machine learning. That's sort of the name of the conference. It's a new research conference that was started to be a home for this sort of research. Research that focuses on systems issues in the machine learning context.

So looking beyond just ML over joins, of course data is spread across multiple tables. You really need to get this holistic view of the interactions between these different variables for machine learning to yield maximum benefit. But sometimes your data may also be multiple modalities. You might have structured data, and then you might have text data, and then you might have time series data. You might have image data. Keeping the recommendation example, you have products, ratings, products, users. The products may not be just structured data, like brand, tags, and price and so on. You might have images of products. You might have reviews people have written about them. These are unstructured data sources.

Currently, if you want to exploit these sorts of unstructured data sources, you need to use deep learning tools, TensorFlow, PyTorch, these sorts of tools, to understand images, understand texts using cog nets and recurring nets. Coincidentally, yeah, the pioneers of deep learning won the tiering world if you saw the news yesterday.

These sorts of tools, how do we integrate them with these structured data processing tools, that is another open question, and that's the second focus of a lot of my research, which is making it easier to deploy deep nets for unified kind of multimodal analytics, where you want to pull together structured data and unstructured data for building predictive applications. There, it turns out to be an even more challenging problem, because the data processing and the data workflows of these deep learning tools are very distinct from what we know in existing data systems literature, which is primarily been relational and semi-structured and so on. Even scalable ML, people have largely focused in the classical ML setting on linear algebra computations. So very kind of predictable access patterns with different kinds of operators and behaviors.

Deep learning tools have very different footprints. Their computational cost is much higher. If you do this so-called roofline analysis that the computer architects recommend, most deep learning models fit under the compute bound regime, where the data access is no longer the bottleneck. I/O is no longer the bottleneck. It's raw computations that's the bottleneck.

So that fundamentally changes what matters when you're building this sort of a system. Where does most of the time go? What should be optimized for in terms of the computations, in terms of the memory usage, in terms of the data layout and all of that? So that is a wide open exciting space that we just started our foray into.

[00:27:53] JM: When I've got a table and then I've got another table, and I want to join those two tables. That results in a bloat of the dataset. For people who are very unfamiliar with database joins, describe why that occurs in more detail.

[00:28:10] AK: Oh, sure. Let's do that with an example. Like the ratings, users, movies, tables and recommendations, say like Netflix or Amazon product recommendations. The data is managed in a multi-table database. So you have a table for product information, like what's the brand, what's the name, what's the price? You have a table for your user information; user one, user two, whatever, hundred million users. Maybe you have a million product and a hundred million users.

Now the third table is your ratings where you want to predict is a user going to predict – Is a user going to purchase a certain product or not. This is how all these tools kind of give you prediction for ratings. Like, "Hey this movie, I'm going to predict that you'll really like it, or this product I'm going to predict that you might want to purchase." These are called recommender systems.

The way these are built predominantly is using content-based methods, where they pull together information about all of these kind of features about products, about users, about ratings, to build a predictive model. So now in the future, when you as a user log in, it will predict which of these products that you have not bought yet are you most likely to buy, and then give you that prediction.

Now in order to do that, the data scientist has to feed all these data into a predictive model to train a machine learning model. Now if you do this join of ratings, users and movies, this is a key foreign key join in database term. It's also called a star join, because there's one central table, called the fact table, that contains your prediction target, which is the stars, number of stars for your rating. Then you could have your timestamp of the rating and whatnot. Then you have these two foreign key attributes, the user I.D. and the product I.D. that point to this other tables. That's how these tables are connected. User I.D. points to the user's table. So it will be a record, which user gave this rating in the past? Then the product I.D. points to a product. What was the product that was rated?

So now, in practice, you might have tens of billions, even hundreds of billions, maybe even trillions of ratings that Amazon and Netflix would have accumulated over the years. Because they have hundreds of millions of users, each product for instance might be rated tens of millions of times. So now when you do this join, the record of the product from the product's table will now get duplicated that many millions of times after the join. Say, if there's 10 million ratings for that product, after the join, that product's record in the output of the join will be repeated 10 million times. There's a huge amount of redundancy that is introduced by these sorts of joins.

Same thing can happen to a user. If the user has rated, say, 100 products in the past. After the join, the user's record would be repeated 100 times. Because of this repetition of these data values, the output of the join could be enormously large. It could be even 10 times larger than the size of the input database. This build up can cause all sorts of bottlenecks. Obviously, storage is a bottleneck if the storage is large intermediate file somewhere. That will waste a lot of storage space. It could waste a lot of memory if you're doing in-memory computations, and you're wasting a lot of resources.

On top of that, in the ML over joins work, what we showed is because the data has so much redundancy, the computations of the ML process also could have a lot of redundancy. That means you're wasting a lot of time doing compute as well. It's not just storage and memory, but also compute wastage. That gap could be pretty tremendous. It could even be in order of magnitude, even two orders or magnitude in some cases.

Now, as more and more enterprise users want to use cloud services, like Amazon, Google, Azure, they have to do a pay-as-you-go kind of purchase of compute and storage and whatnot. Guess what will happen if your data is 10X larger. You'll have to pay 10X more money to Amazon, and that might be great for Amazon, but it's not great for all of these enterprise users. If your compute is going to be taking 10X longer, then obviously you're going to pay 10X more money to have this on.

So this sort of issue has become even more relevant these days because of the pay-as-you-go computational model, pay-as-you-go economic model of the cloud services. Even all of these cloud vendors, internally, they want to build better platforms for their ML toolkits. So they are also interested in kind of improving these offerings that they have, supporting newer functionalities. As far as I know, none of them have yet to adapt sort of the ML over joins approach, but this is from research and it's kind of from recent research. So who knows what will happen in the future.

[00:32:20] JM: One solution to the ML over joins problem that you explore is factorized machine learning. What is factorized machine learning?

[00:32:32] AK: So the first approach we took to attacking this problem was this inspiration from relational query optimization, which is let's say you're doing a selection over a record on the output of the join, relational algebra tells us that you can rewrite that query into a selection over the input of the join and then you can still get exactly the same result.

Based on that classical result as like decade's old, we introduced this technique of factorized machine learning, where the idea if we view the machine learning computation as an aggregate computation over the output of the join. Now, we can rewrite that aggregate computation into aggregations over the input of the joins.

Basically, instead of doing machine learning on the output of the joins, you kind of do partial learning over the inputs of the joins, stitch them together in a sophisticated way based on the algorithm and you can reconstruct exactly the same model that you would have gotten otherwise. That is the technique of factorized machine learning. Basically, you're decomposing the ML computations and pushing them down through the joins to the base tables.

[00:33:31] JM: And you've proved that's the same computation?

[00:33:34] AK: Yup. So the first work on this appeared in Sigmod 15. It's the first part of my dissertation. We focused on this large class of ML models called generalized linear models, things like logistic regressioning and regression and so on that are very popular standard simple techniques all using [inaudible 00:33:49] methods. We formally proved that the model that you get, the coefficients that you get, will be algebraically identical to the coefficients you would get if you do it on the output of the join.

Now, because these are all numerical computations, they might be some floating point issues and the tenth decimal place and so on. But in the machine learning world –

[00:34:07] JM: How dare you?

[00:34:08] AK: In the machine learning world, that may not be that big an issue, because what you care about is prediction accuracy, and we show that the prediction accuracy is pretty much unaffected. So that was the factorized machine learning approach.

Later on, we showed that, "Hey, this sort of factorized machine learning approach seems to be really useful for linear models. What other kinds of ML models can we do this for?" So that was the second research question, where we wanted to show instead of trying to understand the computations of every ML algorithm individually, can we can up with an abstract representation language to do this factorized ML approach? The inspiration again is from relational algebra. You have an algebra to express data computation relational algebra. What is such an algebra and machine learning? It turns out that there is none, because machine learning is just way too diverse. But the closest you have is linear algebra, which matrix manipulations.

You have matrix vector multiplications, matrix-matrix multiplications and so on. Things like people who use MATLAB and R are familiar with. These sorts of computations express many ML techniques. It includes supervised ML techniques, like the generalized linear models. It includes unsupervised techniques, like clustering algorithms, for example. Like k-means clustering. It also includes feature extraction techniques, dimension [inaudible 00:35:16] techniques, like non-negative metrics factorization and so on.

So not everything, so like deep learning for example is not expressable using these sorts of bulk linear algebra computations, but it captures a large fraction of machine learning computations. Many kind of domain scientist in biology, in political science, in all these other fields, they write new algorithms for data analysis in these languages using R, using these sorts of tools that expresses computations and bug in your algebra.

Then in the second work which appeared in VLDB in 2017, we showed that if you can express your ML computations in this formal language, what we can do is build this infrastructure called Morpheus. It's a middleware layer that can factorize this language itself. What does that mean? These linear algebra operations, there are only a finite set of them that matter for machine learning, perhaps like a dozen; matrix-matrix, matrix scaler and so on. We rewrite these linear algebra operators over the joins.

Basically, instead of doing a matrix multiplication with a vector, the matrix is the output of the join, we rewrite that matrix vector multiplication to computations over the input of the join. This is where we introduce this abstraction called a normalized matrix, where your matrix and your linear algebra computation is no longer just one matrix, but rather the output of the join of multiple tables.

So in the database world, we call this logical data independence. It's one of the celebrated aspects of databases, also called views. You don't have to physically create a data that is different from the data that you actually have. So, physically, the data is stored in one way, but logically you view it in a different way. That's logical data independence. It's one of the key benefits of productivity that relational systems offer. We were the first to bring it to this machine learning landscape, this linear algebra landscape with this normalized matrix abstraction.

So now, what Morpheus will let you do is pretend as if your data is just a single table. Forget about joins. Forget about multi-table data. Write your ML algorithm as if it's a single table. Give that to Morpheus. Give the schema of your data, "Here's the foreign key relationship and here's the multi-table data." Morpheus will automatically rewrite that procedure that you gave into a factorized ML procedure and execute it on a normalized data. Now the data scientist no longer needs to worry about how do I rewrite my ML implementation to operate on the multi-table version.

This is basically automating the data of factorized machine learning to any ML algorithm expressable in that language. We prototype this Morpheus middleware on R, on Python. We have extended this in various ways and we've seen some adaption from companies for these tools. Google and Oracle were also very supportive of this research. They gave me some research awards for this and they were also excited in kind of integrating this into some of their stacks. So that's one of the things I'm going to be talking about later today. I'll also do a demo of our library. It's open source. All my projects are open source on GitHub. So Morpheus is available on GitHub too.

[00:38:08] JM: So as you're describing, this results in dramatically less computation expense. Well, on the training process basically, right? So any benchmarks for how much it's saving people?

[00:38:20] AK: Yes. So one of the things that is interesting here is because we were the first to kind of approach this ML over joins space, we kind of gathered datasets that we can show results on, and I collected datasets from Kagul, from all these repositories and kind of instituted like seven benchmark datasets that are multi-table joins, like star schema joins. Things from like Walmart, like sales forecasting. Things from Expedia, like ranking of hotels. Things from Yelp recommendation, a bunch of datasets.

Each of these real-world datasets fits the key foreign key star schema that I told you about. If you are a technical data scientist originally, you would just join all of them and then treat them as a single table. But now we have this factorized machine learning tool. We could benchmark how good is the factorized machine learning approach versus kind of do the join and then do machine learning on that table.

What we found was the speed up depends on two things. One is how much redundancy does the join introduce. Two, what is the ML algorithm. Different ML algorithms have different speed ups depending on what the data kind of fan out of the join is, you will get different [inaudible 00:39:25] of redundancy. What we saw across these seven real-world datasets was you could get even up to like 35 times speed up. So that's like between an hour, we can reduce it to two minutes for your training process. That could have real productivity gains for a data scientist and like you are sitting in the loop. Instead of waiting for an hour for the training to finish, it will be done in two minutes. This was on kind of standard ML models, like logistic regression, k-means clustering, matrix factorization and so on.

Now, some other ML algorithms, the computations may not have that much redundancy depending on what the algorithm does. So we showed that there the speed ups could be like 5X or 6X. So that means instead of an hour, it will be kind of done in maybe 10, 15 minutes. So that's sort of – There is a spectrum based on what is the amount of redundancy the join introduces and what is the ML computation that is going on. Yeah, we could get even up to kind of two or one orders of magnitude speed up in real-world datasets.

Apart from the speed up, which is waiting for the front time, you also have to remember that the memory and the storage requirements also go down. Even if the ML computations are such that there is no redundancy. For example, we extended Morpheus to mini-batch stochastic gradient descent, which is needed for, say, neural network training. Your runtime for neural network training will not go down that much, because these neural network computations, the dominant fraction of the runtime is not the data access part, but rather the internal hidden layer updates during back prop.

So in Morpheus flow, what we showed was you don't have to materialize your data just because your neural network compilation doesn't have redundancy. You could save a lot of memory in storage if you do this join and what we call a lazy fashion. You construct the records on the fly on every mini-batch for the back prop to operate. So that could dramatically reduce the memory footprint of your data. Instead of maybe 10 gig, it will be just 1 gig or 2 gig. So there are these different benefits that we saw when we benchmarked these techniques.

[00:41:14] JM: I want to learn more about your interactions with these corporations that adapted or took a close look at least at your approach to machine learning factorization. So you said Oracle and Google. They both used it? Is it production systems?

[00:41:35] AK: So there are four companies. So LockBlocks was a database startup that actually used it in production. They have a retail stack and they have their own ML analytics system that they built, and so they kind of integrated the factorized machine learning –

[00:41:46] JM: They're just taking your open source stuff off the shelf?

[00:41:48] AK: I think they kind of just took the idea and re-implemented it on their platform. The open source tool that we released was adapted by this company called [inaudible 00:41:55]. It's an ecommerce company apparently. Some data scientist randomly from Russia spoke to me at this VLDB conference and then said, "Hey, we looked at this. This looks very relevant for our ecommerce use cases." I said, "Great! So then just try it out and let me know." Then we had some chats and then that data scientist also pointed out, "Hey, we also do feature interactions for linear models, because you don't want to just run out a linear feature set for logistic regression. Its capacity may be low [inaudible 00:42:22]. So you want to do parallelize interactions of features."

It turns out that's a not linear operation. I thought about it a little bit more and then formulated a new research problem. Then I gave it to a student and then that research problem appeared at Sigmod this year as another paper. So these interactions with practitioners can lead to surprising consequences. Apart from just adaption of my research on to their platforms, it can also inform my own research for new problems in the future.

This can have positive feedback loop that I experience in my work. I also interacted with Microsoft. So during my dissertation days, it was kind of funded by Microsoft Jim Gray Systems Lab, Professor David DeWitt at Wisconsin, and Microsoft gave me kind of no strings attached access to their resources. Their researcher engineers and code and all of that, and they had this internal tool called Cosmos, which is like this massively data parallel infrastructure and –

[00:43:13] JM: Now external, right?

[00:43:15] AK: Now it's available to the world. Yes, on Azure. CosmosDB. But back then it was internal. This was like 5 years ago. But I collaborated with this web security engineer there who gave me access to some of these multi-table datasets for predicting malicious user accounts, and I prototyped this idea on that dataset. It turns out that you could kind of save storage space tremendously on Cosmos as well. So it was kind of explored for production usage. I don't know where they took it afterwards, because Microsoft had a bunch of reorgs. I don't know what was going on inside, and I lost that team afterwards.

But then I started interacting with Google, and this was the ads backend. The data infrastructure and analytics team, the DIA team, terrific people. They kind of had interactions with me about how do you these sorts of computations on their database resident data. They are on this massively parallel kind of planet scale database system, and they are in charge of the data for Google Ads, which is like the cache co of Google.

So they were interested in bringing machine learning closer to their data rather than kind of shipping it out to different tools. That's where they were interested in understanding these sorts of techniques. They funded sort of this research [inaudible 00:44:18] Google Faculty Research Award. More recently they kind of migrated to Tensorflow-based setup. I don't know if they kind of integrated this lazy join approach yet with the Tensorflow. But it's now available on open source. We kind of used it.

Oracle more recently, the head of Oracle Labs came and visited us for the SoCal DB day, which was this research event that I ran last October in San Diego. It's like a bunch of companies and universities in SoCal. The database groups, we get together. Companies give talks. The faculty give talks. The student give post-presentations and so on. Oracle Labs kind of noticed this project, Morpheus, and they thought it was very relevant for their GraalVM polyglot data science engine that they're building.

So they wanted to integrate some of these optimizations, these algebraic rewrite rules that we talked about into the GraalVM engine itself. So that's a collaboration that's going to start this year. So Oracle awarded me another Faculty Research Award for that and they're excited about this, because one of their key use cases with their GraalVM engine is data science workloads.

So going from data manipulations, to machine learning computations, to deployment, and the data would come from a database. It could be an in-memory kind of file and so on. GraalVM handles bunch of different frontend languages and backend data sources at a unified manner. That's sort of a very interesting use case there. It's a second level of generalization.

Morpheus kind of generalizes different algorithms expressed in a formal algebraic language, but it has to be prototyped in R or prototyped in Python separately. With GraalVM, we can knid of build it once for the internal representation language of GraalVM, and then now it will be available in Java, JavaScript, Python, R, everything in one go. That's sort of the exciting part of it.

[00:45:57] JM: Yeah, we did a show on GraalVM. I didn't know about this application of it. I did realize it was a highly flexible tool.

[00:46:05] AK: Oh, interesting. Okay.

[00:46:06] JM: Yeah.

[00:46:07] AK: Small world.

[00:46:08] JM: Yeah, small world. Well, I mean, there were a bunch of people that were interested in GraalVM.

[00:46:11] AK: It's a pretty cool system. It's a unified runtime for polyglot data analysis. So that was the factorized machine learning part. The other part was the avoiding the join logically part. So the first part was pushing down computations with a join. The second one is even more radical, and that one is a little farther afield from the database query optimization style world. Because that one truly bridges this knowledge gap that exists between our understanding of data systems and our understanding of machine learning. That's the Hamlet project.

There, we actually applied statistical learning theory to show that in many cases, you can pretend as if the join doesn't even exist. What does that mean? Let's say we have this recommendation example, ratings, users, products. We showed that in certain cases, depending on the metadata about the data, you can pretend as if the products table doesn't even exist. You can use the product information in the ratings table as a representative of all of the products features. That's pretty radical, because now we can say, "Hey, I have a three table database, but because of the metadata, Hamlet tells me that this table is potentially not going to help improve my accuracy." You can actually ignore that table altogether.

The reason that happens is, like you said, the data exists in this multi-table form, but machine learning requires a single feature vector. Going from that data to the feature vector is a feature engineering process. That process today happens in almost kind of a blind fashion. Data scientists are very greedy about features. They want to throw as many features as they can from as many sources as they can into their machine learning model, and then let the model figure it out.

What we showed was this sort of kind of bring everything together and give it to the model approach, in some cases could be enormously wasteful, enormously wasteful of both the computation time, the resource and also the work of the data scientist. From my interactions with some data scientist in practice, we learned that not all tables are available to you as a data scientist. Your organization might have hundreds, even thousands of tables. But if you want to do, say, a sales forecasting or churn prediction application, you have to go and figure out what are the tables that I need. Ask those relevant teams with permission to access their data. Maybe get a copy of that data and jump through all those hoops to get that data.

Not everybody has access to all the data in the organization because of access control and governance restrictions. So there's this hoop that you have to jump through. Then once you get the data, then you enter this regime of factorized machine learning, right? You have this multi-table data.

So what we showed in the Hamlet work was sometimes you can actually avoid these extra tables altogether based on the properties of what those tables are. In very succinct terms, what we showed was that using the foreign key as a feature can actually capture the same information that you get by bringing these extra features in.

So in some sense, it's like using learning theory, you're telling that the query itself can be modified to avoid an entire join altogether. It's a very kind of novel setting that truly marries the world of databases and machine learning in some sense, because this is the first time somebody has looked into the learning theoretic implications of Oracle database dependencies as like these two distinct fields of inquiry, database dependencies and learning theory, and now we kind of married the two to show you can actually avoid the table altogether.

[SPONSOR MESSAGE]

[00:49:35] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why

DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[00:51:42] JM: Now, most people that are building machine learning models are not going to want to think about this. They're going to want it automatically making their machine learning training process more efficient. So do you have an idea on the insertion point for this kind of technology? How do you give it to the every developer out there that doesn't want to study relational databases and machine learning at the same time?

[00:52:12] AK: Sure. In this Hamlet case, there is actually almost no code that is needed. It's a very conceptual contribution, because as a data scientist, you get the tables from the –

[00:52:21] JM: It's almost like a compiler tool chain thing.

[00:52:24] AK: It could be. It could also be kind of a tooling thing where you're kind of visualizing the data or getting the schema of the data first. Once you load, you can have datasets, right? You can just get the number of topples in these datasets. Hamlet is basically a decision rule that says – What we call the topple ratio. If you have, say, for a given product, at least so many reviews, then the product record is not needed at all. So that's sort of the topple ratio rule.

What you need to compare is how many records do we have in the reviews table, versus how many records do I have in the products table. If that ratio, which we call the topple ratio, is above a model specific threshold, I can ignore the product stable. That can be done just to before you run the machine learning training. That can be done just before you do the join for the machine learning training.

It can be done even at the level of the data system that procures the data. So the data engineer can just do this count of the number of records and then kind of ignore this table [inaudible 00:53:18]. So it can be done in many settings. What we showed in the hamlet case was this sort of avoiding the join can potentially not hurt accuracy at all. So now the hamlet idea has also been kind of integrated into production in different settings. There was this company – Again, LogicBlocks kind of tried it out on their retail analytics use cases. But the most interesting anecdote that I have from academia is this company called Make My Trip. It's like India's Expedia.

One day out of the blue, I have this engineer from Make My Trip email me saying, "Hey, I read your hamlet paper. I tried it out on my dataset. I just kind of skipped some dataset," right? They're building the model, they had to do the join. I said, "I looked at the topple ratio and I decided not to join this dataset at all."

My machine learning computation was 8 times faster. The accuracy didn't change. Then he said he told his manager, and his manager was very impressed or something like that. Then that person applied to UCST as a master's student the next year. Came to UCST, did his master's and he was my TA last year.

[00:54:18] JM: That's preposterous.

[00:54:18] AK: I know, right? This is like the way things work. So that was the fascinating anecdote. Google, the TensorFlow Extended. Team inside Google Brain was also excited about this and they wanted to try it for some of their use cases. They asked the same question, "Where do we plug this in, this decision rule?" There are many points to plug this in. It could be in an ML platform like TensorFlow Extended, or Uber's MichelAngelo, whatever, where you have this data validation of the data sourcing phase where you say, "Here's the data that I want to build the model on." At that stage, you can inject this decision rule in and say, "Okay. This tool will say, "This table is potentially not going to improve accuracy. Don't bother joining it at all." Something like that. So there's like many entry points for –

[00:54:56] JM: Many angles. Do you think we need some new databases for applications of machine learning, or new abstraction – Like I look at this space and I'm just like, "Nobody has any idea how to build their "data platform". It's like a complete mess. There's no best practices, like I've got my data lake over here. I got my data warehouses. I got my streaming frameworks. I have no idea what to do with any of it. Or is this just like the beginning of some – As a new field and there's just no getting around the fact that we need all these different abstractions.

[00:55:31] AK: I mean, I've taught about this a little bit. One thing that is happening is there's this upheaval on the technology infrastructure side, the systems infrastructure side. Cloud computing has changed a lot of things. People no longer just want a siloed RDBMS within their walls. They want elasticity. They want fault tolerance. They want decouple compute and storage and all that stuff. That is affecting the way you're using machine learning techniques as well. You might want to do it on large amounts of data. You might want to do it on small amounts of data. You don't want to buy a giant server and then underutilize it. So on that side, there is this upheaval.

On the other side, there's also the programming models for machine learning are inherently diverse. There is no one unified algebra for machine learning, like relational algebra is. That's just the way machine learning is. So I don't know if you know Pedro Domingos at Washington. Calls this the five tribes of machine learning, there's [inaudible 00:56:19], there's a connectionist, there's a symbolist, all of that.

All of these techniques of machine learning are popular in different settings. Like logistic regression still remains the most popular classifier on the planet. Even though there's all these deep nets and stuff like that, because that often acts as a baseline for everything else. CNNs and RNNs are not that popular on structured data, because they just don't operate on structured data.

So deep learning tools, although there's this huge amount of buzz, are not going to kind of take over the whole machine learning landscape. On the other hand, the data systems infrastructure in influx. Consequently, the ML systems infrastructure will also be influx. But in my research, I kind of focus on a more abstract level of what are the processes that are fundamental regardless of what the systems infrastructure is. This avoiding the join stuff that I told you about, the hamlet, matters no matter what infrastructure is, whether you're doing it in-memory, Python, or you're it on TensorFlow, or you're doing it on Spark. A join is a join no matter where you do it.

So that's sort of my research days, just focusing on these abstract problems that are almost fundamental to the process itself, rather than kind of coupled to a specific systems infrastructure. We can prototype this on different systems infrastructure with different systems concerns, but the contribution of the idea itself is rather abstract and kind of longer lasting in some sense.

Now, concretely, where is this whole landscape of machine learning system is heading? I ran this panel discussion at Sigmod team workshop. So there's this workshop at [inaudible 00:57:41] Sigmod Conference called the Data Management [inaudible 00:57:43] Machine Learning. It's a new workshop that I started a couple of years ago. I kind of gave a talk there once and then I ran the panel, and the workshop last year helped fronted. The panel discussion had a bunch of leading researchers. Matei Zaharia was there, Joe Gonzalez was there. A bunch of others. I asked this question, "Is the systems landscape for machine learning going to remain fragmented as it is?" and it seems like the consensus was probably it's going to remain fragmented.

I don't think there's going to be kind of a consolidation of everybody is now going to use only TensorFlow and deep learning. That is probably never going to happen. Because the data types are just so diverse. There's structured data, there's time series, there are texts, there are images. The interpretability concerns are so diverse.

In many domains, you simply cannot get away with throwing everything into a black box model. But in many other domains, you can get away with throwing everything into a black box model. That is kind of a societal concern. It's kind of a legal concern, and that landscape is very different. Maybe in 10 years from now, web companies will start getting regulated very heavily, like already GDPR has come along and –

[00:58:42] JM: 10 years?

[00:58:43] AK: I know, right?

[00:58:44] JM: Keep dreaming. More likes 6 months.

[00:58:46] AK: 6 months? Okay. So it's already here. Then they'll start kind of facing – They'll get kind of regulators and auditors breathing down their neck. The reason why many enterprises are very slow to adapt in new technology is because they have legal concerns, because they have auditing concerns and so on.

In healthcare, for example, they are even more conservative in many cases, because every cost of miss prediction is potential human life. So there's just this whole spectrum. On the other hand, like clicking an add on Google.com is very low kind of consequence for getting a miss prediction. There's this whole spectrum.

Because of all these fundamental reasons of what the data types are, what the business concerns are, I don't think there will be kind of a consolidation of everything around one system. But, like I said, there are fundamental principles and processes that are inherent across all these system landscape, and that's sort of what I go after in my research.

[00:59:38] JM: Which do a lot better than we're doing today is what you're saying, right?

[00:59:41] AK: Exactly.

[00:59:42] JM: We could have some consolidation, or at least an option to consolidate.

[00:59:47] AK: Exactly. I mean, I would think that a lot of cloud vendors are betting big on AI and ML services as a future for their cloud services. I think that could see a lot of uptick across a lot of companies. In some sense, that is a form of consolidation. Instead of running your own kind of platforms individually, like every company go build your own data science platform. The cloud vendors are kind of consolidating that. Of course, there're a slew of startups that are also building data science platforms. So at that level it might be consolidated. It's more at a macroscopic level that there's a consolidation rather than at the microscopic level of what is the execution engine for my training process.

[01:00:26] JM: Right. Right. We're at Strata. Let's wrap up by just talking about a little bit what have you learned – I mean, I know you're only here yesterday. We saw Ben Lorica's talk. I'm actually interviewing him next. So maybe you can give me some spoilers on his talk, because I didn't see it. But what are you learning at Strata? What kinds of revelations are you having?

[01:00:45] AK: This is the first time I'm showing up to Strata. This is the first time I'm giving a talk a practitioner's focus conference. I've been to a lot of Sigmod and VLDB, research conferences in the database and data management world. I've also attended SysML. I will be attending SysML next week, which is this new conference. Cider, a bunch of data management conferences. But the reason I wanted to show up for this practitioner's conference was to learn from the data scientists and to learn from the companies what is the landscape today like. Where is it going? On that front, there's been a lot of interesting kind of conversations and interesting things that I've learned. I went to a lot of talks, some of which I learned new things from on the explainability and kind of model interpretability stuff. There's this new thing called Shapley values that I learned about.

[01:01:27] JM: Shapley values?

[01:01:27] AK: Shapley values. This was a day before yesterday at the tutorial. It's called Shapley Values. It kind of generalizes and unifies a lot of model explanation primitives that people have started building.

[01:01:36] JM: It sounds useful.

[01:01:37] AK: Yeah, that's pretty interesting. Then yesterday, I was interacting a lot with some of these companies, H20, Salesforce, Amazon, Google, some startups, determined AI. So people that I know, companies whose work I'm following and kind of got to understand what, say, Salesforce Einstein is up to, for example, because that is quite relevant, the kind of ideas that we're exploring. This process of feature engineering and model selection and all of that stuff, you want to make it a platform so that you can commoditize it. Salesforce is applying it to 100,000 customers or something like that. You cannot afford to throw one data scientist at every customer even. So you have to create a platform that automates this whole process for all of those customers.

H2O also has some sort of what they call driverless AI, although I'm not [inaudible 01:02:19]. So I just want to understand –

[01:02:22] JM: Self-driving.

[01:02:22] AK: Yeah. I know, right? Self-driving. Everything is self-driving these days. So I just wanted to understand what the landscape is among the cutting edge firms and see what sort of techniques and tools these data scientists and practitioners are looking for. What sort of things are still open, and just talking to them about my research and getting some kind of their [inaudible 01:02:40] and their thoughts on that. Because, like I said, I like interacting with practitioners. It creates this kind of virtuous cycle of, one, you could potentially see your research getting adapted by them, and that will give you the satisfaction of your research having real-world impact, which is quite rare. Often, research takes years, even decades to have impact. These sort of varies, it can actually have immediate impact sometimes even before publication.

The second is the other way around, kind of like talking to them kind of exposes new open bottlenecks that I might not have thought of if I was just sitting in my room and academia. So that kind of informs new research questions. It leads to new research outputs, new systems, new concepts that we want to kind of formulize and build up on this. All of these, I can have like interactions with practitioners and kind of thinking about this process has also helped me characterize this whole space of ML systems into a book. I don't know if I mentioned this, because Morgan and [inaudible 01:03:31] kind of invited me and a couple of my coauthors to write a book about this whole space. So we wrote this data management and ML systems textbook. The first textbook in this area that characterizes everything that I described, all the way from what's the history of this ML in databases, to scalable ML systems, to deep learning systems, to ML lifecycle systems. We kind of laid out the landscape. What are these systems? What are the concerns? What are the systems issues? What are the research questions? What is the existing landscape? What are the open questions and all of that? That book came out just a few weeks ago, a couple of weeks ago.

It is quite unusual for us, system professors, to be writing textbooks. But this is a research oriented textbook. It's targeted at practitioners, advance data scientists and ML engineers who are looking for what is the state of the art if I want to do X. What are the existing things out there? What are the open questions? It's also targeted at researchers and graduate students who are looking to kind of enter into research in this area, because this is a hard area of research. A lot of people are starting to work on this area, but they need to understand what was done, what worked, what is not done, what did not work and all of that. So all of these was born out of my interactions with the practitioners. So that's why I want to keep that contact alive and come to these sorts of conferences that will allow me to interact with more of them.

[01:04:40] JM: Arun Kumar, thanks for coming on the show. It's been great talking.

[01:04:43] AK: Thank you, Jeff, and thank you for all the questions and the opportunity to talk about my work. Let's enjoy the rest of the day at Strata.

[01:04:48] JM: Agreed!

[END OF INTERVIEW]

[01:04:53] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]