SDS PODCAST EPISODE 669: STREAMING, REACTIVE, REAL-TIME MACHINE LEARNING

Show Notes: http://www.superdatascience.com/669

Jon Krohn: 00:00:00 This is episode number 669 with Adrian Kosowski, Co-Founder and Chief Product Officer at Pathway. Today's episode is brought to you by Posit, the open-source data science company, and by AWS Cloud Computing Services.

00:00:17 Welcome to the SuperDataScience podcast, the most listened-to podcast in the data science industry. Each week, we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today. And now let's make the complex simple.

00:00:48 Welcome back to the SuperDataScience podcast. Today, the positively brilliant researcher and entrepreneur, Dr. Adrian Kosowski returns to the show to give us a taste of what the future of machine learning looks like. Adrian is Co-Founder and Chief Product Officer Pathway.com, a framework for real-time, reactive data processing that is based in Paris. He has over 15 years of research experience, including nine years at INRIA, a prestigious French computer science center, leading to the co-authorship of over 100 articles in a range of fields, theoretical computer science, physics, and biology, for example, and he's covered topics in those papers, like network science, distributed algorithms, and complex systems. He previously co-founded and led business development for Spoj.com, a competitive programming platform used by millions of software developers, and he obtained his PhD in computer science at the ripe old age of 20.

00:01:38 Today's episode will appeal primarily to hands-on practitioners like data scientists, machine learning engineers, and data engineers. However, we do our best to break down technical terms and provide concrete examples of topics so that anyone can enjoy learning about the cutting edge in training machine learning models. In this episode, Adrian details what streaming data processing is and why it's superior in many ways, to the batch training of machine learning models that

historically dominated data science. He talks about how streaming data processing allows highly efficient real-time model training, how reactive data processing enables data applications to react instantly and automatically to never before seen input data, potentially saving firms vast sums. He talks about when it makes sense for computer scientists to become a product leader like he did. He talks about why Pathway selected the particular programming languages they did for their platform, and the big up-and-coming opportunity for data and machine learning startups. All right, you ready for this mind-blowing episode? Let's go.

00:02:40 Adrian, welcome back to the SuperDataScience podcast for your first full-length episode. You were here...we met at the Open Data Science Conference West in San Francisco, back in the northern hemisphere autumn. And you recorded an awesome episode on Liquid Neural Networks - that's episode number 632. Fascinating technical topic. Adrian, where in the world are you calling in from today?

Adrian Kosowski: 00:03:09 So, I'm based in Paris. I'm calling in from just outside Paris, France, from a place which used to be the countryside, but is now meant to be the Silicon Valley of France.

Jon Krohn: 00:03:19 Oh, yeah. And it's the Pathway Office, is that right?

Adrian Kosowski: 00:03:22 It is. It is.

Jon Krohn: 00:03:23 Nice. And, so you're the co-founder. You're a co-founder, and you're the Chief Product Officer at Pathway, which is a reactive data processing framework that allows people to create real-time data products much more easily. So, I know that we're going to get into a lot of what Pathway is, but before we even get into that, I want to let our listeners know that you very kindly offered, you're offering 10 free hoodies to the first people that respond. So, when I, when we release this episode, it'll be, it's always on Tuesday

mornings from a North American perspective, it'll be the morning, and I post on LinkedIn from my personal account, a big post about what the episode's gonna be all about. When I make that post, I'll include in it, to say, the first 10 people that ask for a Pathway hoodie, get one, and you're offering to ship them anywhere in the world. So, it's very kind. Thank you, Adrian.

Adrian Kosowski: 00:04:20 It's our pleasure entirely. These are really good hoodies. We hope you'll be satisfied.

Jon Krohn: 00:04:24 Yeah, apparently they're hoodies so good that you'll want them, even if you're in a very hot climate.

Adrian Kosowski: 00:04:31 That's what they say. We also do software, but we do hoodies most of the time.

Jon Krohn: 00:04:39 Yeah, so when you guys aren't designing hoodies tell us about the software that you make. So, you have a reactive data processing framework. Yeah. Tell us what that means and what can you do with it.

Adrian Kosowski: 00:04:49 Yeah, sure. So, reactivity is all about the art of dealing with changing data in such a way that you don't have to worry too much about the processing part when data changes. I think if you want to be formal, there's probably some dictionary or encyclopedic definition of reactivity which will tell you it's about being declarative, declarative in a programming sense, like explaining the logic without imperatively saying what to do at every step of a data transformation, without explaining vocals in like a functional programming sense, explaining what the transformation should be. And that's about the essence. So, it's really combining the ability to be declarative with the ability to process data changes automatically in an efficient way. So, that's a notion known as incrementality.

00:05:48 It's the idea that when data changes, you don't have to do a full recomputation of over models, of over things that you've designed in your data pipeline or in your data

science project. You just do a minimally computation to react to the way data changes. So, I guess the most best known example of a reactive system out there is your spreadsheet, call it Excel sheets, whatever you prefer, software, you define the rules on the data, and when the data changes, the cells update. So, this is like one example. It's a data processing example. It's not one that scales very well, but it's an example of data processing. And actually, since spreadsheets came in, I think nobody has been able to fully replicate the success of this type of approach at scale. And we come with, with our attempt. I should say that reactive reactivity is a concept that's very well known, very familiar to frontend developers, if you've worked with JavaScript TypeScript.

Jon Krohn: 00:07:02 Yeah. It's even probably the most famous framework right now for front end development is called React.js.

Adrian Kosowski: 00:07:09 It is, and the others that don't have React in their name only active, nonetheless. All of them are. And like the, the kind of place this has got front-end developers to is that you when designing a front-end system you don't have to do as much event handling as you would do 15 years back. So, some of you may remember having to write things like on-the-click events to describe the state change of a button, you know, when you click, you have to do it, and so on. And these days in front-end development, you don't do it that much. Surprisingly in data processing, even data processing at scale, if you want to work in a real-time setup, you want to work with data that changes or with streams of event data, a lot of the time you still find yourself doing the equivalent of on-the-click-do or something like this.

00:08:06 The back-end equivalent is on-data-change event or on-arrival of a certain packet of data do. And this is something that has to be done behind the scenes, no question about it. It's just that we don't necessarily want the developer, be it the data engineer or a data scientist to be exposed to the pain of doing this type of on-something

event processing after they've already had to put a lot of effort to design the system just to get their job done, to create a model or something.

Jon Krohn: 00:08:44 So, it seems relatively straightforward to me to understand in the case of a user interface, we have a browser app that is reactive to somebody adjusting how wide the browser is, or whether they come in on a mobile browser or a tablet or a desktop, that the website automatically adjusts, or as you're saying, to behaviors to somebody clicking on something and the application reacting to that. In the case of data changing, what does that mean? Like how do the data change that are flowing into a machine learning system? Like, the machine learning system it could be handling different kinds of data types, or what does, yeah, what does it mean when the data changes?

Adrian Kosowski: 00:09:35 So, the most straightforward setup, I'd say, is when the data type does not change, you just have to deal with new data, the same type, just data that you haven't seen before. If you're a data scientist, like the ideal world is when the data sample that you're working with is the actual data that will be, that has to be analyzed. When you're in the online ever-changing world where new data comes in, this is never the case that the data sample that you look at, at the time of designing your model is for one, for which you need to do the insight. So, where, in some sense, where at least for the testing data, the real-world testing data, is not known to you. Sometimes even in real-world scenarios, it may be the case that the training data, so to speak, is not known to you. So, your model is retraining itself or adjusting to new incoming data.

Jon Krohn: 00:10:39 Online learning.

Adrian Kosowski: 00:10:40 Online learning and, and things like this. So, this is for the general setup. If you like diagrams, you can picture data inputs on the left, data outputs on the right, and your data pipeline in between, and whatever fresh events

come in from input need to be taken into account. If you want to make life fun, fun in, in a architecture sense, you can also put a human in the loop, somebody who's providing feedback on how your model is performing and saying, actually, we should tweak this parameter. For example, it's like, you know, it's, today we need to adjust because it's a, we have a special day, and something like this. Some parameter for your forecasting prediction, anomaly detection model. Or the user can say, actually in the training data, the was a mistake, and we need to pull out the training data point and say, look, this should never have been obtained like this, or the label should have been changed. Or a certain class of automated inputs which entered the system may have entered with incorrect values. For example m, there was some confusion between m denoting meters and miles. Of data that input needs to be rescaled, and you have to sort of unlearn the data that came in previously and relearn with the new data. So, anything, anything is kind of possible in the sense of data changes for the system.

Jon Krohn: 00:12:17 So, in the past, on the podcast, we've talked about issues around things like feature drift, where, you design a machine learning model to be able to handle the kinds of training data that it's encountered in the past, but then the real world changes. And so the inputs, the features that are coming into the model, so you know, you described a flow from left to right. So, on the left-hand side, in those data inputs, the inputs are fundamentally changing, the structure of the inputs is changing. So, even though, as you say, it's the same data type, you know, it's still a 16-bit float value, or it's a, you know, it's an integer, whatever, the features are now in a range that are outside of your training data because the world has changed. So, is what you are describing, this reactive data processing, it's designed to allow your machine learning models during online learning to be able to adapt to this feature drift automatically.

Adrian Kosowski: 00:13:25 So, this is part of the story. I think reactive data processing should be treated very broadly and if you start implementing it in the larger system like data pipelines in enterprise, which are processing event data, the start of the story is in data engineering. The end of the story is in analytics. In order to benefit fully from this type of framework, from real-time data processing in general, it has to be put into place like end to end, or at least it helps to put it into place, end to end. And the, the models, the analytics models, the machine learning models that come in are kind of the cherry on the cake, but the one that allows it to [inaudible 00:14:13], a lot of value. So, we make sure to make this possible.

00:14:17 This is just to say that in a strictly machine learning context this would be a very, let's say, a good application and at the same time, an ambitious one when you get to models which are sufficiently advanced to be very much aware of problems like feature drift versus, let's say, an intermediate class in somewhere in between engineering and more advanced machine learnings of models which have a certain time horizon, a time window in which they learn, and they are updated as this time window moves ahead. I'd say this is like first more natural example in the real world project that you'll be looking at for last few months of data, some kind of moving average on the data and trying to adapt to it. So, indeed it may be the case that we get into questions of models getting outdated and needing updating, model versioning, and so on. But this is some heavy machinery which comes in relatively late in the project. A lot of the time it's actually possible to just adapt to the structure of the data itself by having a model which knows how to adapt to the structure of the data, which has this capacity to encompass horizons to be somehow scale-free with respect to the nature of the data.

Jon Krohn: 00:15:54 Nice. Every company wants to become more data-driven, especially with languages like R and Python. Unfortunately, traditional data science training is broken. The material is generic. You're learning in isolation. You

never end up applying anything you've learned. Posit Academy fixes this with collaborative, expert-led training that's actually relevant to your job. Do you work in finance? Learn R and Python within the context of investment analysis. Are you a biostatistician? Learn while working through clinical analysis projects. Posit Academy is the ultimate learning experience for professional teams in any industry that want to learn R and Python for data science. 94% of learners are still coding 6 months later. Learn more at Posit.co/Academy.

00:16:40 So, let's try to make this example a bit more concrete by going into let a specific use case. So, you and I were planning on later in the episode talking about global supply chain networks and how the pandemic broke these, and how reactive data processing combined with IoT (Internet of Things) hardware could make, help make rather global supply chains more resilient to abrupt delays and shocks. So, maybe let's dig into that specific use case now, so that as we kind of address other questions around how reactive data processing works, we can like tie into the specific concrete example.

Adrian Kosowski: 00:17:18 Yeah. So, actually, we started Pathway working closely with actors in the logistics industry, working to improve global supply chains and global transportation patterns. Logistics is a pretty fascinating area because if you look at the importance for value in the scale of the industry, it's about something like 10% of a world economy. It's, so it's really big. It's highly concentrated, and a lot of a value is in international trade, trade that goes on containers, tucks, large vehicles. And it's in some sense from a data processing perspective, when we were starting, this was largely terra incognita. It was of a new world of analyzing this type of data patterns related to logistics assets.

00:18:12 What IoT gives in this setting is the ability to trace moving assets, be it containers, trucks, parcels, you name it, end to end. That is you attach a sensor and you have a whole trace, the whole tech of an asset that's moving. And this

leads to an interesting situation in which something like 10% of world industry has some of its most important data lying in one data schema, one data format, which is essentially a big table of events related to moving assets. That's the table whose columns are something like timestamp, asset ID, location x, y, or GPS latitude, longitude, and for type of event that happened, whether it was a kind of just a measurement of a location, whether if it was a measurement performed by a IoT, for example, of temperature, pressure, door opening, some kind of alert.

00:19:18 And this kind of table actually captures a lot of other things in logistics as well. For example, if there's the facility where your parcels are being scanned, all such scans events also enter in this type of table. So, you have one kind of input data table, data schema, which seems to capture everything that's happening, which is quite unusable from the point of view of business intelligence analytics and process monitoring and observability used directly, just because it's super hard to query. It's hard to express in the language such as SQL, a query on the data, which would extract what's important. And the important questions are related to process, things that are happening. So, for example, a logistics client may be interested in knowing what are the risks of anomalies of given type, like shocks happening to your sensitive pharmaceutical shipments in the next two days on a given route, let's say, Rotterdam to New York.

00:20:32 And, if you look at the data inputs, all the information is there. If we've given a lot of man years and a lot of patience, you could probably get it in the end by hand. But, it's not all that easy to extract and automate over global processes. So, that's where we started. We started working with a process of enriching this data automatically, converting the schema to add structure to it in such a way that is actually possible in real-time to get an enriched data schema, which is queryable, and which reveals information about the process. A lot of aspects to it related to, first of all, trajectory mining,

understanding how things flow, a uncovering automatically the key locations, so, this is like automatic Geofence detection in the [inaudible 00:21:28] of the sector. It's about understanding anomalies, congestion, delays as they happen or even before they happen, and putting into place predictive models.

00:21:42 So, there are many steps here. And already getting all of this done in a batch setting, a setting where you have all the data available, like just historical data, is a challenge to express it cleanly and to get an analysis of a snapshot, so to speak, of the data. And it gets extremely complicated if you, on top of this, you'd would want to manually create logic in a way which would take into account changing data. It becomes a task which is both tedious [inaudible 00:22:24] requires a lot of duplication of logic between the, let's say, the offline case and the online case and so on.

00:22:30 So, our effort was on the one hand to automate this, and on the other hand also to figure out what parts, what models in machine learning, what transformations of data, are actually amenable to this type of approach. Basically say, forget what cannot be done, focus on what can be done here now, and make it possible to make this robust. So, just to give you an example of the types of data processing routines that we have. We've designed to work robustly across different modes of transport, be it ship, truck, train, container, or vehicle even working for small assets like parcels, sometimes with animals, sometimes with public transportation. So, basically to have models which work with very little or minimal awareness of what is actually being traced, what type of acid is being traced to allow changes to this process, to allow new modes of transport to be introduced, to allow changes to the logical process.

00:23:40 If you follow like what your couriers and delivery people are up to before Christmas, it's actually amazing how the whole logistics network adapts, there are new depots

being opened, temporary depots. There's changes to the process. Things are happening completely differently in peak season. And if you want to make sense of it, you have to have a system which is able to take into account these changes as they happen. New depots open? Okay, it's not something that you'll want to manually introduce. It's something that you have to kind of capture from the data.

Jon Krohn: 00:24:17 Wow. Okay. Yeah, so, that is very concrete and crystal clear. So, these real-time data products, reactive data processing frameworks like Pathway allow data models to be applied to complex systems like a delivery system, like the global supply chain network. And it's reactive automatically to things like vastly greater volumes in peak season at around Christmas time, heading into Christmas time, including things like hubs coming online that previously weren't there. And so you don't want to be going to your data scientist and saying, "Hey, a new hub opened up yesterday. It's December 1st, and a new hub opened up. We need you to retrain all of the machine learning models that we had to be able to account for this new hub. And then the very next day, two more hubs open up, because we're one more day closer to Christmas, and you're like, sorry, data scientists, we got two more hubs. We need to retrain that data model again". And every time the data scientists are like, oh, this is gonna take like a week. And so, instead with a reactive data processing system, it's flexible to these kinds of changes automatically.

Adrian Kosowski: 00:25:36 Yeah. And that's exactly the [inaudible 00:25:39]. It's, it's kind of also changes the whole workflow for way the non-technical user can interact with the application. So, you can have an expert in the domain who is working with a system. And rather than asking data scientists to update models, they either get, at a case of a simple model you can actually redeploy the model as it happens, or in some case, you get a question whether you want to update your model, you update your data. So, if we talk of these hubs

that open at the beginning of the day whoever's managing the dashboarding part of the solution gets information that "we have detected, 20 new hubs, which opened since yesterday, please approve or correct errors". And essentially the work of this person is more in fine tuning or fixing things that didn't, that were not detected with, let's say 100% accuracy or performing small collections to the automatic labeling that's being done, rather than actually introducing [inaudible 00:26:50] the change process manually.

00:26:54 Just to say from our perspective this is where we started out with. So, we started with one concrete data product in logistics, and this is our flagship data product. It's being used by major companies, freight forwarders like DB Schenker, which is third worldwide largest freight forwarder in the world by the French Postal Services, which also have a wide international network on this operating in other countries as well. So, the kind of use there is what we see and we feel, at the same time what we are delivering is the visibility of data scientists and data engineers to work in the same way as we do. So, we want this, we want to share the experience, we want to share the experience both with everybody, with a wide community, and also with the data teams of our clients. To work closely with them, to allow them to modify the data pipelines, to include new data sources. So, it's really very much about giving this full development experience and having it work like having it work as a developer tool.

Jon Krohn: 00:28:12 Nice. That's crystal clear. And so, now that you have expanded beyond that initial use case, your most developed product around global supply chain networks, what other kinds of use cases are people using this reactive data processing for?

Adrian Kosowski: 00:28:27 So, this is interesting because like the use cases expressed in our terms, so, I'm assuming we are discussing data here, and data like data audiences are extremely horizontal. They're not like tied to industry-

specific verticals. So, many of, many of us change jobs, you know, between different companies, like going from, from healthcare to, let's say whatever, transportation or manufacturing. But for the kind of use cases that exist from a, you know, from a data science perspective are pretty consistent [inaudible 00:29:07] anomaly detection in real-time, predicting anomalies, detecting fraud, which is the type of anomaly detection, and recommendations online updated recommended systems. Some further ones, which do appear in some cases of [inaudible 00:29:25] forecasting, and time series forecasting, plus let's say some, that are further down the stream. But in some sense, it's also a question of the most immediate value, for most immediate, most immediate pain point is really for one where you need to act quickly. And the time horizon related to anomaly detection, to alerting is just much, much shorter than the time horizon related to updating forecast models typically.

Jon Krohn: 00:30:01 Right? So, I can imagine financial applications, for example, where you're detecting fraud would be, a really great use case.

Adrian Kosowski: 00:30:07 Yeah, financial applications are a nice one. It's also interesting that you have several horizons in the financial application. Let's say you are doing real-time transaction processing, be it more on the major card processing actor side or on the DeFi side. Either way, you have a window of opportunity of two to three seconds to block certain types of transactions. Those who have the user is still not getting impatient. And then a post-processing like window where you can still undo some transactions of try to fix things. But things get worse from in the horizon of seconds to minutes. So, this time horizon is actually very, very short in this case.

00:30:52 One that many of us in data know is related to monitoring of health of systems of processes that are going on. So, things around observability in process of server monitoring in because the system reliability field. This is

an interesting use case, which is very special, but it also gives, I think an idea of the kind of alerting that we are looking at. If there's a human operator involved, you want to react within your SLA time window, which will typically be something like 15 minutes. This is the number that comes up most often given guidelines of major companies. So you have like 15 minutes to react, and the kind of data that you have is how many minutes of this 15-minute window that you as a human have to react, are eaten up by the system being slow to give you the information. If it's more than five minutes, you're really, really angry, but maybe you could go from five minutes to three minutes to 30 seconds. Sometimes you increase for value there, and you can actually do way, way better if you get this alert faster.

Jon Krohn: 00:32:16 Cool. Are you stuck between optimizing latency and lowering your inference costs as you build your generative AI applications? Find out why more ML developers are moving toward AWS Trainium and Inferentia to build and serve their Large Language Models. You can save up to 50% on training costs with AWS Trainium chips and up to 40% on inference costs with AWS Inferentia chips. Trainium and Inferentia will help you achieve higher performance, lower costs, and be more sustainable. Check out the links in the show notes to learn more. All right, now back to our show.

00:32:53 All right, so now that I have a clear idea, and probably our listeners have a clear idea of applications of this reactive data processing, let's dig into it technically in a bit more detail. So, a big contrast that comes up a lot in the context of reactive data processing, and that even came up in a recent episode number 661 with Chip Huyen. So, there's this idea of batch processing versus stream processing. And so, what's the difference from a machine learning perspective between these two processing modes, batch and streaming, and how does that tie into reactive data processing?

Adrian Kosowski: 00:33:34 So, actually the, the answer that would be taken from a reactive data processing perspective is that if you look at time, time as something that appears in your data, if time is an important feature for you, then it's an important feature for you. And whether you are in batch or in streaming, you should handle time. If time is not a feature in your data then you should have a privilege of not looking at time and the system will be able to handle things for you. So, just to give a very concrete example, let's take-

Jon Krohn: 00:34:15 Yeah, please. You just blew my mind.

Adrian Kosowski: 00:34:16 Let's take-

Jon Krohn: 00:34:17 If time, if you have the luxury of time, not mattering-

Adrian Kosowski: 00:34:21 Yeah. And it's actually something like time not being a feature means for, for example, look at spam. What's a spam message? A spam message is the same, like you see spam, you recognize it, you'd read the same thing as spam today, as in the week or in a month, right? It's spam is, spam is like, there's no time aspect related to spam. However it might be the case, that depending on how the spammer is behaving, you may be able to detect a spam message at some point or not. For example, when somebody is sending a message for the first time, the spam filter may still not be aware that it's spam. But after a message has been sent 10,000 times, 100,000 times your model for spam detection or a spam detection filter will be able to figure out that something is amiss because of a behavior of a spammer.

00:35:18 Your messages are classified as belonging to a kind of cluster component somewhere, which is spamish. And all of this can be classified as spam, right? So, in this sense what's happening is that the incoming data allows the model to improve its classification decisions over time. However, the logic of the classification process as such is largely not tied to time. It can be...time doesn't have to

play a role in it. You can do a lot of things without thinking about time. So, what the answer is in the reactive setting, is that you could imagine that you have access to all of your data in batch mode. Basically, you have all of the input, you process it, you give the answers, and these are the best answers you can have because they are answers given with all the knowledge.

00:36:20 And then if you launch the same code in a reactive system, it'll be doing its best to maintain answers, which are as good as possible given the current knowledge. So, at the end of the day, it'll converge to the same outcomes, which you would've had if you had had all the information initially. And on the way, it'll be doing kind of a best-effort classification based on whatever partial knowledge it has. So, it may happen that a message comes into your inbox, call it Gmail, for example. Suppose Gmail has a reactive spam filter. It's classified as non-spam, but five minutes later when new information arrives it's, it becomes automatically reclassified as spam and can be pulled from your-

Jon Krohn: 00:37:07 [crosstalk 00:37:07] Oh, wow.

Adrian Kosowski: 00:37:09 So, this is, this is the idea that you don't have to worry about the deployment for way how things are going to be you know, run, rerun. You don't have to worry about how the data streams unfold over time. You just designed the logic and you put it in the system, and you somehow you were released of from the worry about of the streaming data.

Jon Krohn: 00:37:32 Cool. All right. So, I know we're getting there, but, so how does this, we haven't, like concretely defined this difference between batch and streaming?

Adrian Kosowski: 00:37:42 Yeah, so batch is the concept that your computation is run and scheduled. I think batch, orchestration, scheduling, ... These are concepts that all go together, they are part of the same mindset, it's one where you look

at the data as it is now, you run a computation on it, or you run some processing on it, something happens eight hours later, 24 hours later, the scheduler says, rerun for batch, you rerun for batch, and things get updated. So, it's this type of mindset.

Jon Krohn: 00:38:18 To give maybe an extreme example that gives a clear sense of this, when people have been using ChatGPT since it came out in late 2022, it has a modal that comes up and gives you all kinds of warnings. Like this is experimental, but one of those warnings is that the data hasn't been refreshed since 2021 or something like that. And that's because the underlying model, so up until very recently at the time of recording the most common natural language model the people were using under the hood of ChatGPT, was GPT-3.5, and this GPT-3.5 had been trained on a batch of data that was current up until 2021 at some point. And so there was this big batch of data, and then it took, who knows, maybe weeks of training, maybe even months, I don't know.

00:39:12 GPT-3.5 is so large that the whole processing pipeline could have taken months to do, particularly when they want to add in all these kinds of safeguards around using it ethically. And so, that model is not streaming. It's very much the opposite. And you could take that same model architecture and update it in 2022 at some point, but it's only getting like this annual update on a batch basis. So, you have these big batches of data, and so you're describing the situation where the batches could be much smaller, where it could be every eight hours or every 24 hours that we have in machine learning model in production. That is where the model weights are being updated from new data. So, that things like a new hub in the delivery network that came online in the last 24 hours is now going to be handled. You know, we have some data regarding this hub and we can be handling it. Yeah. So, that, so this kind of gives the sense of batches and how we can have big, big gaps between model refreshes or small gaps. And then streaming is a completely different

kind of perspective where it's like data point by data point in real time being updated.

Adrian Kosowski: 00:40:31 Yeah. Streaming is exactly this perspective where the data flows in a manner where when the new data point comes in, you handle it. So, if I think like, to be perfectly, like to take a, maybe a human perspective and batch and streaming for the inherent preference for, for looking at batch or looking at streaming depends very much on what, what world you come from. If you're more in the mainframe or static database type of thinking, you probably have a natural preference for looking at computational systems as batch systems. If you are more in the microservice design APIs, things communicating, data flowing, your natural preference, your first approach is more around working with streaming data. And each type of system, whether it's around batch processing or around event-driven stream processing has very different characteristics.

00:41:52 In general being event-driven has the advantage, the obvious advantage of lowering latency because you can react to new data as it, as it arrives so your models can update. The difficulty is in making non-trival logic work. And by non-trivial, I mean actually doing something like a database join which is already impossible in most data processing frameworks to maintain a join of two data tables, join as you'd see it in SQL and Pandas or whatever. This is something that's hard to do in real-time, let's say, and that requires a special framework, which has to know how to handle the joint to be able to do it. So, back in 2019, it was messy to do. Now more and more frameworks are catching up, but this is about the forefront in terms of tooling as what's possible, what kind of difficulty of processing is possible in real time.

Jon Krohn: 00:42:59 Cool, cool. Yeah, so it sounds from, you know, I come from this background of being a scientist. And so, we ran discrete experiments and you get a batch of experimental results. And so, I'm used to this idea of having like, this

specific table we're like, okay, experiment one is done. I have my set of data, it has this many rows because that's exactly how many human participants we had in this first study. And then it's just kind of static indefinitely. And I train a model on it, might then publish the results of the model. I might even make the data open-source, make it available freely online, put it into the public domain, and then anybody can use my perfect table of data from experiment one that never changes. And so, yeah, so I come from this batch background. It's probably the case that people that are used to the kind of streaming situation that you're describing, people who are used to microservices, they're probably more likely to come from a computer science background or a software development background.

Adrian Kosowski: 00:44:12 So, I'll, I can give you an example because right now we are doing some experiments on our side. We are, we're running benchmarks of Pathway against other frameworks. So, we are doing an experiment and in, if in some sense these experiments, let's say last weeks, they take weeks to run, and you need multiple repetitions of an experiment to get to liable data to be able to take, let's say the medial of the data points or some average or whatever. But imagine you just wait for the first run to complete, the first time of your experiment, and you already have a data source, which can be like, let's say Google Sheets or Excel. Google Sheets is a better idea because it's kind of more online, more live, where you've put your experimental results there.

00:45:03 And at this point you can use whatever plotting software you like let's say Tableau to get some first charts out of it. And now suppose you continue running the experiment and as new data points arrive, you get larger and larger sample and your dashboards are live, they get updated as the sample goes. And for example, once you've got five runs, a line, which was a bit noisy at the beginning, has kind of smoothed out because you've lost the statistic noise level has been reduced. So, your dashboard is a

kind of life dashboard on top of a scientific experiment. So, in some sense, in this way, you can look at a scientific experiment being done in streaming mode with new data is coming in, it's updating, fixing the dashboards and at some point you press stop and this dashboard is production ready.

00:46:02 And this time is a good point to say that the fact that you are in streaming mode does not mean that you cannot look at data back in time. It's a bit like with most of our productivity tooling that we're using these days, we are used to the fact that you can look at the version history, you can go back some number of edits, you can go back to a past snapshot or past version of the system. And this is especially important in distributed systems which are serving answers to requests. And you have to be absolutely sure that you are serving answers based on a consistent version across machines. Meaning that you are referring to one snapshot, let's say from a few seconds, few minutes back. But if there's a problem with the snapshot, you could also roll back to a past snapshot, maybe 20, 30 minutes back or even further. So, there a kind of notion of snapshotting of a past. But you're working inherently with a kind of timeline of, of things that move, move forward.

Jon Krohn: 00:47:10 So, is this, this constant movement, this constant timeline, is that maybe the most challenging aspect for machine learning engineers when they're trying to implement streaming applications?

Adrian Kosowski: 00:47:23 If you were using an API which has streaming in the name, then it's an added challenge. It's an, it's a second challenge. I wouldn't want to say it's the biggest challenge in terms of some kind of ingenuity being conceptually the most difficult that would be undermining the, the effort that's needed to actually get something going in machine learning, which is enormous, but it is an enormous challenge in terms of system deployment, system maintenance, making sure there are no bugs, making

sure this is actually feasible to them. So, the, the industry standard for now is that going from batch prototype, that means static data scientist showing a dashboard to a live streaming deployment of the same dashboard, updating in real time is roughly 10 times the effort.

00:48:38 So, I'm not saying it's necessarily 10 times the same ingenuity needed, it's just 10 times the effort at 10 times for timeline. Likewise. So, it's a question of cost. It's a question of some R&D risk involved, some risk that that things will go wrong or will not be moving fast enough and afterwards there's a question of maintaining this and making sure the system is going. So, at this point, the system, if it's a streaming system, it has essentially hired somebody on the MLOp side on the data and engineering/reliability side to make sure that the pipeline is being properly maintained. So, it's a challenge to a degree, but many systems never end up going from batch to streaming or from prototype to production with realtime data. Because this challenge is just too much to overcome. And depending on the, depending on the setting, it's sometimes maybe the case that the project actually brings 10 times more value if put into place in real-time or even more. But it just never happens because the cost side is too prohibitive, or the time management is too prohibitive. And the approach, what we're taking is basically to automate the second step, make it possible too.

Jon Krohn: 00:49:57 So, streaming can historically be 10 times as complicated but it can offer more than 10 times the value once it's implemented in production. And real-time reactive data processing frameworks like Pathway are designed to dramatically decrease that 10 times complexity in getting it set up and allowing you to realize that 10 times value,

Adrian Kosowski: 00:50:22 Well put Jon. The cost aspect is actually quite fascinating because as we do this transition from batch systems to work more and more with steaming data there are a lot of dimensions on the cost side, which come in with a

steaming system, which are not obvious. So, one aspect is that the structure of expenses related to infrastructure, cloud infrastructure changes. Traditionally streaming systems had a higher hot storage component requiring a lot of state to be maintained in hot storage. And this is only changing now. This is a cost factor which has been reduced, and this is the counterpart is actually that batch systems have an enormous computational cost associated with them because you are doing a lot of essentially useless computation every 24 hours. You're recomputing, recomputing, recomputing, the same things, even though maybe in the horizon of one day, 1% of your input later changed, you're doing a 100% recomputation,

Jon Krohn: 00:51:40 Right.

Adrian Kosowski: 00:51:40 So, in the case of, you know, of actors, when you, when you look at your cloud bill and you take the free components, which is storage data processing, and networks communication, let's say if a data processing clusters, your, your spark clusters or whatever else you're using to, to churn the data, are generating a significant part of a bill. It may be the case that moving to streaming-like systems or reactive systems which allow you to transition through from micro batching into this, this online world will actually cut the bill of a-

Jon Krohn: 00:52:18 [crosstalk 00:51:18] Cool. Yeah, that's really interesting because to me it seems inherently that if you have this continuous learning, oh, of course that's gonna be more expensive because it's always running all the time. But I hadn't thought about it from this perspective that with batches, you're retraining the entire model every time. Yeah, and so that is very computational expensive. Very cool. Hadn't thought about that.

Adrian Kosowski: 00:52:41 There are many like practical aspects to it. And one is just, you know it's, it's for reality around us for ad hoc instances are often much more expensive than instances [inaudible 00:52:53]. So, the rule of thumb is that if your

computation is running for four hours, five hours, six hours during a day, during 24 hours, it already makes sense to have an instance just reserved for this like 24/7 without asking for a new instance every time. And if you're in this space by making this like batch computation spread out over time, you are already better allocating new resources because you get something like a factor 4, 5 in computational resources for free just by having more hours in the day to use. And, there are a lot of possible gains which just come from this better like spreading of computation over time.

Jon Krohn: 00:53:40 Very cool and crystal clear. So, to allow these reactive data processing operations to happen efficiently over this 24-hour data as opposed to doing it in batches at Pathway, you talk about transformers, and so these aren't to be confused with transformer architectures that have become really common in large language models like GPT-4 and going all the way back to the early transformer architectures like BERT, it's a different kind of transformer. So, it's kind of like maybe how the word Kernel is used in computer science. It means so many different things, and so transformer here means something different. Although from our chat prior to beginning recording, it sounds like there is a kind of a common thread to the etymology of why a transformer architecture is called that and why your reactive data processing operations are called transformers. Do you want to fill us in on these transformers?

Adrian Kosowski: 00:54:38 Yeah, I guess this is a great question actually. And just to say that the name transformer is kind of controversial in the sense that obviously transformers of a T in GPT and basically one of the more commonly used words on the as an architecture in deep learning and kind of backstory from our perspective is that one of our fans in Business Angels is a co-author of the original Transformers paper. So, [inaudible 00:55:13] attention is all you need. And our CTO also comes from the attention/transformers world. So, it took a lot of internal discussion if we wanted to use

the word transformers in a pure data processing sense just there's not much depth to it. A transformer is something that transforms one kind of data into another kind of data. In our case, it's a box which transforms tables into other tables. We are not the first to coin this term, and I think it's one of them, like on the data engineering side, it's pretty unambiguous and I think for the season we-

Jon Krohn: 00:55:57 Oh, really? So, in data engineering, this use of transformers as an operation that transforms one kind of data structure into some other format that's, it's quite common?

Adrian Kosowski: 00:56:12 It's used by some other frameworks. I'd say as like, we didn't want to coin new terminology for it. There's actually an interesting point which also helps to take the path between data engineering and data science is that in data engineering, a lot of the time you think of your data tables, data sources as assets meaning that there, there's certain, like they have a physical representation somewhere in the data warehouse, and when you combine them, you create a new physical representation. And to do it, you need to run a job. If you are in batch mode or something like this, it's, there's a very physical feeling to data flows, whereas in the data science world more often than not you are designing a kind of block, a building block, which just it's like a function which takes certain input parameters and has certain output parameters, which is much more flexible to use.

00:57:16 It's not tied to specific inputs, it's more composable. It can be used inside other functions. In our case in the case of Pathway, we have support for iterations. So, for example, iterate a given transformation until convergence. So, you can have a transformer, which is like one iteration of let's say gradient descent, and you put it into a block which says iterate until convergence, and then you get a new transformer, which is like for, for looped version of the first one. And, and in this sense, something

that, that has these like data table interfaces in and out that's pluggable and, and moldable in the, in the data flow is how we use the term transformer.

Jon Krohn: 00:57:59 So, cool. So, now I understand what transformers are in the context of data engineering and if transforming some data format into another data format as part of this data flow. So, why is that so critical as a part of streaming and reactive data processing?

Adrian Kosowski: 00:58:17 So, the kind of place which is crucial here is actually being able to express logic easily, clearly, and in a way which is accessible. And we are always somewhere on the boundary between declarative and imperative populations. So, if you say you want to filter a table leaving only values in which a given column is larger than 10, you're defining a kind of block which says, let the data in. And you get one table at input and output table with the same schema, but with fewer lows. So, it's as if you were wiring together blocks and the connections between these blocks you can, you can feel a bit as if you were fiddling with, I don't know, with circuits or whatever. You're just pinning them together and you get the kind of data flow.

00:59:20 And this is like the outmost perspective. And somehow when you look inside each of these blocks, so when you're running a filter, which just leaves values with the larger than 10 you're probably doing a built-in, which is, which is like a filter, select something like this. But you could be doing something a little more advanced, which is like applying a lambda function to every row of your table in the MapReduce paradigm. So, you could be doing something more powerful. And then you could also be doing some transformations which are specific to multiple rows of the table. So, this comes in a lot when the data is interconnected. For example, when your data tables represent a graph, a network, and the connections between nodes expressed by edges, which are pointers to other nodes, and you want to perform some kind of local

operations, for example do a search of a given neighborhood of a graph, and there you can switch to a, it's more convenient to switch to a programming paradigm where your main actor is not the table, but it's really the data row, the data element, the data node that you're working with.

01:00:39 And you act around it somehow. So, you are, you're not working with tables, you are, you are working with, with individual roles and this is what's happening inside for transformers. So, if you peek inside, you have this possibility to work with individual data elements, it's a bit like it, it's a bit like to some extent, if you define a kernel when you, you mentioned the word kernel. So, kernel is very, very much about designing a transformation, which is around individual wires. And the whole thing the whole deployment has to perform has to run for multiple kernels in panels. So, it's the counterpart of this in the world of incremental reactive process.

Jon Krohn: 01:01:24 Cool. And there are instances in the context, certainly of reactive data processing where these transformer operations are themselves, machine learning powered. Right. And I think you've referred to those as smart replacements.

Adrian Kosowski: 01:01:38 Absolutely. Yeah, that's the objective, and from being a product person, I'm very much like on the developer experience front. This is one of my hopes here is to provide a seamless experience transitioning from what you could call relatively mundane data operation which is defined in SQL to one, which is in some sense smart or fuzzy or machine learning powered. To give you an example, if you take the group by operation, which groups row of the table according to a value of a given column. So, standard group by. This returns a certain number of groups of rows, if you take a clustering operation on the table, clustering understood in a machine learning sense, clustering also performs a grouping of data points, right? So, from an interface

perspective, if you use a table or data frame API to express the operation group by, as you see it in SQL or Pandas and clustering with arbitrary custom logic have the same API.

01:02:57 So, in some sense, for us, it's just an interchangeable box. If somebody is, for example, grouping, let's say points in space by the x and y coordinates, they put a group by type of box, but then they change group by to spatial clustering. And then the internals pass from an SQL like database operation to a machine learning operation, which performs a clustering of points in space. So, this is like one example for group by versus cluster. Another is about, for example, join versus smarter fuzzy join if you're joining two tables by name, okay, versus pure join operation. If you are joining, but there maybe typos in your names or some other inaccuracy, so, you're not sure which column you're joining with which column, then you're getting into some kind of fuzzy filtering. And the API again, stays the same, but for implementation is completely different. And we switch, like we make the transition from the data engineering to the data science side.

Jon Krohn: 01:04:05 Crystal clear, thank you for those examples. They make it very easy to understand how transformers can be machine learning powered and be these smart replacements in a reactive data processing framework. And you mentioned in your response there, how you are a product person. I think this is interesting. So, we mentioned right at the onset that you're the chief product officer at Pathway, but if I dare say your background strikes me as the kind of background that somebody would usually have as a CTO as a Chief Technical Officer. So, you have this very technical computer science background. I know you're still hands-on today writing code, and you're in this CPO role. And, so I think you might have given us a bit of a clue to the answer as to why it makes so much sense that you're the CPO and it's

because the Pathway product is designed for highly technical people. It's designed-

Adrian Kosowski: 01:04:56 Absolutely, absolutely. I think, the main all goal of a products person is to understand the end user and to be like the end-user. It makes things simpler, but at least to understand the needs of the end-user and to be able to have a certain empathy for these needs. This is why I guess for a technical person, it's easier to be a CPO of a developer product. I would say if it were not a developer product, I'd be completely disqualified. Just not kidding. I mean, it's like it's too tempting to switch size and be part of a creative process and say, I mean what if we could add to our data processing engine something else that, you know, when starts doing, doing a tech push and so on.

01:05:50 Yeah. I agree fully this temptation exists and it's kind of a bit of a complication, always knowing how the internals work. However, being technical also means that I can actually test the product in action. I can add five lines of code in Pathway. I can see it. I can see if I'm able to showcase the things that we are promising ourselves, I can review some showcases or pieces of open-source run by others using Pathway to see if it's all meeting the expectations that are made of it. Especially given that the state of data processing frameworks as it is, is such that the developer experience and the experience of maintaining them, scaling them is one of the bigger issues. Cause things just are either a follow apart or don't work 100% of the time. So, when the experience of actually interspecting debugging is not optimal. And this is one of the bigger pains of data teams, and having trying to experiment with how we can resolve these issues and work on the experience front is a major challenge and major, I think, opportunity also for this space to provide some improvements here.

Jon Krohn: 01:07:12 Nice. That makes a lot of sense. And yeah, I can, it's crystal clear to me now. When I was preparing for this episode, I was like, oh, this is kind of interesting. And

then it started to become really obvious. Okay, he is the perfect product person because he is the ideal user of this product as well. So, speaking of your technical background before Pathway, you were a computer science researcher for over 15 years, including at a renowned French institute called INRIA in French, and it translates into English as the French Institute for Research and Computer Science and Automation. And when you were that researcher for all those years, you published dozens of peer-reviewed papers, you specialized in things like network science, distributed algorithms, dynamical systems, and the transport optimization that Pathway specialized in initially. So, do you have like a particular area from that time, from your research time that was, that you were really passionate about, maybe you're still really passionate about today? It seems like complex systems, for example, were a recurring theme for you.

Adrian Kosowski: 01:08:21 Definitely. Definitely. If you zoom out, you know, and look at a large system where things are moving, it can be in nature with ants cooperating in the task, it can be a transportation system. Then you look at how it's built, how it works, how the local interactions drive a system, and it's fascinating to observe this from also a computational perspective and ask why it's working like this. So, why is the why is it working and what does it achieve in this way? And can we learn from it? Can we learn things about it? Can we learn certain approaches? Can we try to transfer them into computational paradigms or vice versa? Can we use computational paradigms to explain complex systems? I think one of the recurring themes which exist in complex systems is visibility of distributed control and coordination, which is fascinating.

01:09:23 Another, which is probably closer to the, let's say for big challenges of both machine learning and computer science for the next decade is one of low energy computing or energy optimization. If you look at distributed systems, they actually, for the fact that they

are so distributed, they do things locally, they learn things, they improve things with relatively little interaction, little communication, little computation, and little cost. And for an ant, it's obvious that it cannot, you know, use up more energy than it has because it has to eat to get this energy. So, it's kind of optimizing as well for the computation effort. It cannot have a bigger brain because that would eat up its sugar. So, there are things that [inaudible 01:10:19] in nature that are driven by energy. In world, let's say computing world, especially in for deep learning world, versus something that's kind of acknowledged that we are not optimizing for energy and in, and that a lot of these computations are done in a way which well just gets things done with by scaling resources up scaling cost up.

01:10:50 But, since it's so important, we don't think about the, the energy impact and for actually the incremental way of computation and also streaming computations are more energy efficient. So, it's somehow more natural for me to be in this space, which cares about the amount of data updates that are happening in these systems, and doesn't just recompute everything from the beginning,

Jon Krohn: 01:11:18 Right? The ant brains in a streaming processing system don't need to be as big because they're just online making decisions, one little piece of food at a time.

Adrian Kosowski: 01:11:33 Fascinating, it's, quite fascinating with ant brains for the more external storage an ant can rely on, like leaving pheromones the less it needs to store in its head if it cannot rely on external storage like, because it's too hot in, let's say [inaudible 01:11:52] desert, then it has to do more computations, and they do rely more on internal storage and [inaudible 01:11:58]. So, these surprising tradeoffs between like communication storage and computation happen in nature, and they're very like neatly captured by a lot of those models. So, so it's kind of.

Jon Krohn: 01:12:13 That's so cool.

Adrian Kosowski: 01:12:14 So, yeah, I guess there are many challenges that nature and that complex systems are remarkably good at, but we haven't quite grasped. One of them is the ability to forget. This is one of the things that many natural systems have, which is naturally forget things they've learned or they unlearn things. This is something that's not super easy with deep learning models. So, it is more the, again, the area where lightweight models or models which have some kind of ability to add, delete data points that they, that have an edge in this area somehow.

Jon Krohn: 01:13:01 Rich, rich opportunity for more research there, for sure. And so, you made this leap from all these years of research into being a startup founder. How did you make that leap? And is there anything that you missed from being a researcher? It sounds like you still get to do a lot of fascinating research in the role that you're in.

Adrian Kosowski: 01:13:19 Definitely. I get to miss the frustrating part of being a researcher. So, like one thing that I've discovered is that it's actually gives more, more joy to deliver code or all like small showcases around code, which is open-sourced then to focus on the full effort of writing papers. Maybe GPT-4 will change that and actually, for paper writing part will be taken care of. We just have to focus on delivering the essence for now. But there are many exciting, exciting challenges in which we are doing now, that's true. I should say I'm not completely new to the enterprise/startup world. Given that something like almost 20 years back, I started a programming community called Spoj.com, Spoj com which would be in its day one of the largest competitive programming communities.

01:14:22 It was, we really did it for the excitement of actually getting people to use, to learn competitive programming, to boost their skills. But we also needed the lifeline for it. And interestingly enough, the lifeline, the revenue

channel there was through enterprise who were interested in putting into place a similar framework for their enterprise staining programs in what was known as algorithms at the time. And then started being called a data science but with a, with a gentle migration around 2010. But this was also an interesting experience for me, like drafting my first contacts and doing my first sales, and at the same time, having this enormous opportunity, which was to drive a community, be part of a community, take feedback from a community, and to learn the things that I sort of need to make for product better.

Jon Krohn: 01:15:20 Nice. Very cool. I, yeah, I actually, I was hoping to speak about Spoj.com so I'm glad that you managed to tie it in there. And so, given that you are still hands-on today, so despite being in this CPO role you know, very senior role in a fast-growing startup, you still manage to make time to be hands-on day-to-day. I'd love to hear what your weakest, kind of, like, what kinds of roles, what kinds of hats you have to wear over the course of the week when you're, you know, there's product design aspects, there's programming aspects, you end up, you make podcast appearances, conference presentations, that kind of thing.

Adrian Kosowski: 01:16:05 Yeah, there's multiple hats versus the hat of the product person. And that is already many shades of one hat, because the product person is all about collecting input about features, which comes from outside that could be from outside, meaning from users who are happy, unhappy, or expect some prioritization or just have some comments. Comments from clients who, which come in through, through sales channels and so on. And somehow aligning these needs with what is actually feasible and what is proposed by the CTO, by the development team with a longer term roadmap, which is kind of originates from us and from what we would see in the system. So, somehow being able to make an informed prioritization decision about the different features, elements, aspects that come in is part of my role.

01:17:05 It's an interesting place to be because when part of your product is essentially a framework which has an API, your features are like code in a way, and then this means it's actually way, way closer to code. Another hat is about actually being able to work with the community, the users who are excited about the product, and to be able to either help them or at least understand better where they're excited and see if we can make the product meet expectations. So, this is very much that both making reach outs which are maybe not too overwhelming just to say basically, "Hey, we are there, if there's anything we could do better. We understand that the framework is relatively new still, so it's robust, but you don't have 10,000 answers on stack overflow to, to guide you, but we are here for you. We are here on Discord, you know, you can exchange with us freely and like, get answers probably sooner. And actually also have like 10 excited members of our development team who are able to help guide and potentially come up with new ideas together." So, this is, this is pretty exciting as well to be part of animation of this. One thing which I've done for Pathway and which I know some organizations, we are not the first to do it, so I know some organizations have done it. One organization that has done it is GitHub. We've pushed a lot of the workflow into a combination of GitHub with pool requests with content creation, for example, through markdown.

01:18:54 And if you imagine that you have a monorepo base, which includes your code, your website documentation, and your content pieces, this means that no barriers in the workflow between team members who are more on the marketing side, content marketing side, or on the side of actual creation. Anybody can contribute on a fair basis using the tooling. There is a certain onboarding effort, which is perhaps higher than just with platforms that are not meant for developers. So, you have to devote two hours at the minimum to onboard every new team member, but once the process is flowing, anybody can contribute, and it's kind of transparent whether you're

contributing to the code base, to the documentation, to the content around it. So this, it's much more natural that the key developer will say, "Hey, we are not explaining this site on our website, let me fix it".

01:19:56 And likewise, somebody is not just writing an article or a content piece about it, they're delivering a full executable piece of code, which can be compiled by during a ci cd process into something that goes live as an article on our website, and also tested in the process, whether the latest version of a framework doesn't, like is it's not broken, or that there's no issue with things executing properly, so, that we can be absolutely sure that our content meets developer standards and we deliver quality.

Jon Krohn: 01:20:32 Yeah, a lot of different hats that you have to wear, of course.

Adrian Kosowski: 01:20:35 Yeah. I try to reduce this one. It's something that you can pull off probably only in developer-product oriented team, but one that I highly recommend, if you happen to be doing one just make markdown the language in which people talk to each other, technical and non-technical, you can do basically everything there from like websites in some sense, code as well, or code that generates markdown. And so drawings, anything was possible. So, so you start feeling like one big Wikipedia and one web with knowledge and everybody's kind of interconnected.

Jon Krohn: 01:21:13 Cool. Yeah, it sounds like a great way to work. And I have some kind of experience with that in a smaller scale, where the first book that I wrote, Deep Learning Illustrated, it's in LaTeX, so not markdown, but we did everything in GitHub. So, I had Grant Beyleveld, who works on my data science team at my machine learning company, Nebula. He was a co-author on that book, and we were able to push updates and be able to very easily comment and be able to track changes through that kind of system. And I thought it was really intuitive and straightforward, and gave a great record of how things are

changing in the system. So, what kinds of, other than this GitHub trick that you're suggesting for teams to work with, what kinds of tools do you use daily? Like, what's your programming stack like, personally?

Adrian Kosowski: 01:22:06 So, I'm, you know, I'm, I try to be hands on, but I'm already like in between the two worlds. I don't have a super advanced stack just to say, I use one laptop screen, so this says something about me. If I'm working on one screen, it means I'm like a developer, but not, I wouldn't be able to aspire to certain circles.But for one of the things we have in our setup, it's a remote setup, which means that the develop machine develop everything is set up for, for remote development. So, everything is working on the same developer machine that's shared in the team, which means that I'm using the same setup as everybody, which is a stack meant to be productive. We do have some preferences in terms of IDs, I'm usually a VS Code person myself, like most of the team.

01:23:04 But this is just for, this is just the front and the rest of it is what comes through with our tech setup as, as defined by our CTO. So, I get to benefit from this, and actually having a remote work setup is something that comes through and having all of the team on board with it including for example, the persons who are doing sales demos to be able to, to have this workflow in which you can provide a demonstration on a remote machine that everybody has access to in a specific place, it kind of makes it much smoother to have a the workflow going all across the team.

Jon Krohn: 01:23:46 Nice. And so that tech stack that your CTO defines, what is, what are the, what's the core programming language, for example, a Pathway?

Adrian Kosowski: 01:23:54 So, Pathway is a Python front to Rust engine. We are neatly between the two. In general, a user of Pathway stays on the Python side, has the ability to use SQL, but which is compiled to Python. And then Python is meant

as the language for expressing most of the logic. It largely, how do you say it compiles out the equation in the sense that the Python code disappears, disappears, disappears. Sometimes some function calls survive, but to the extent possible the Python operations are replaced either by [inaudible 01:24:32] or by Nuba, depending on the case. So, it gets low level and GIL lock, like lock [inaudible 01:24:40] and so on. So, there are no, there are no bottleneck there. And the, the computation is taking care of on the Rust side.

01:24:47 So, the team is largely either on the Python side or the Rust side. We do have an enterprise offering, which also includes dashboarding layer to be able to present nicely interconnected dashboards which allow for dataset exploration on top of what's made available by Pathways. So, Pathway does for data enrichment, puts forward tables of data that are ready for business intelligence for business analysts to work with. And we demonstrate this through a SQL layer with several hundred dashboards depending on the data model and logistics. It's several hundred that we can propose to clients or to even data engineers, data teams on the client's site to customize for their own needs or that we can customize. So, there's a certain SQL layer to it as well.

Jon Krohn: 01:25:48 So, the Python makes perfect sense to me as the, what you describe as the front-end to your product, because if this is designed primarily for data scientists, machine learning engineers to be using Python is the lingua franca of data science and machine learning. So, it makes perfect sense. When you were deciding to work with Rust behind the scenes, so it's a functional programming language, it's one of the most popular functional programming languages today. But how did you make that particular decision that this would be the right language in the back-end?

Adrian Kosowski: 01:26:18 I would say that the decision took itself because at the same time, several people with similar mindsets, some of

them on our team, some of them are also involved in open-source projects decided to make Rust the language of choice to it has a certain number of advantages also from our perspective in terms of of the quality we can deliver, we can be sure to deliver. But just, just to say that let's say theR rust ecosystem has gone to a degree where it's actually just as easy, if not easier, to find a data stack with a Python front-end, which has a Rust backend or Rust data representation. So, Polars, for example, as a Pandas replacement, this means that we are not, we are not losing anything by having to interface just a bit more to the C part of the world.

01:27:40 It's still possible. It's just not a big deal for us to have a slightly more complicated glue to see C libraries. It's not that it would be a major argument in the discussion, and other than that, if we have folks who are on board who are willing and happy to use Rust and deliver better quality in Rust, it's a big gain. There are some anecdotal places where last safe typing is causing us some extra work, but at the end of the day, everybody's happy with it as well, so-

Jon Krohn: 01:28:19 Got it. Nice. And so we know from this episode, and also from your previous appearance on the show back in episode number 632, that you're a brilliant person. You can go really deep into the technical weeds on a wide variety of topics up and down the technical stack from the low-end code, all the way to having a product work well for a user. So, I think a really interesting question for you is if there is some kind of approach that's up and coming, it could be maybe if it happens to be data science, that'd be ideal, but some kind of approach or technique or tool that you think is emerging that our audience should know about and be excited about in the years to come.

Adrian Kosowski: 01:29:12 This is actually a big question, and now you've made me feel like that I have to take care of a prediction, I like, because like 10 years now-

Jon Krohn: 01:29:27 Well, yeah [crosstalk 01:29:25] It could be something, just something that's happening now, that our audience should know about, that maybe they wouldn't hear about elsewhere.

Adrian Kosowski: 01:29:36 I think one of the things that I should say as a product person is that productivity is important. And the kind of drive around productivity that's been the Silicon Valley DNA, essentially, it's like everybody reorienting towards productivity is here to stay. And this is something that we see as well also with the let's say for B2C applications of artificial intelligence, they're very much oriented towards increasing the productivity of people. So, it's actually like B2C or B2B from the sense, from the point of view of of being a productivity tool at work or a productivity tool in our personal lives. And this is still there. And I would personally also want see the aspects of growth of learning methods that kind of counterbalance this with for example, energy efficiency of the ability to do low energy computing and to take into account certain other of machine learning models beyond let's say what, what LLMs are capable of delivering.

01:31:22 So, I think I'm one of those who will be pushing for the so-called niche in the sense of the number of applications that you see, but one which has an enormous value tied to it from the point of view of both data processing and enterprise, and behind the delivering value behind the scenes, and helping guide people, organizations towards more informed decisions, towards better insight. So, it's more of the data insight part of the world. That's kind of, I don't know if this makes sense.

Jon Krohn: 01:32:05 Right, right. So, so your, I think your main point is that there's a lot of focus on productivity. So things of course, like ChatGPT, this is seen as a productivity tool and you made the joke that made this makes it easier to write papers as an academic. But what you're saying is that there's still a big gap in being able to get insights automatically from data.

Adrian Kosowski: 01:32:26 I'd say this is for case, and I'd say we are missing something. We are missing some blocks. We are missing some bricks to be able to bridge the two worlds. For the next five years, I'd say that the type of insights that we'll be getting will still be based on pre-deep learning era models, predominantly that is the ability to put these models, which into place in the right way to deploy them, to make them work in an, from an operations perspective, MLOps, [inaudible 01:33:05], that's the big effort. And once we've bridged that effort we will be left with this this question whether we can get, you know, human level insight out of a system that's performing too much data analysis for human to actually to churn through. So, here we are still very much in the range of decision support tools on the enterprise side. We have not made this transition from decision support to like something further than this. And I think there's, it's because the ingredient's missing. It's not just with nobody's tying, it's, it's something's missing.

Jon Krohn: 01:33:52 Nice. Cool. I love that insight into the missing insights in automation. All right, so Adrian, I've already taken way more time than I promised you I would. So, thank you very much for being generous with your time today and sharing so many of your insights with us. At the end of every episode, I ask for a book recommendation. Do you have one for us?

Adrian Kosowski: 01:34:17 Book recommendation. Wow. For book recommendation, I yeah off the top of my head actually, if you mentioned Rust and we were started talking about like learning Rust. If you want to learn Rust, then Rust Documentation is kind of like a book. And I think it's actually an amazing experience. So, this is just something just to say spontaneously that the Rust book is where, it's where you, the difference between a documentation and a book has blurred itself. That's, that's one of the reasons why I like this. But in terms of like looking at my mini bookshelf in this office, I'm as you said, I'm a complex networks person. So, anything related to complex networks always

is a good good thing to read. Complex networks are everywhere from real world systems, social networks, our brains, everything here as a complex network. I'd say for An Introduction by Newman, or Newman and co-authors is always a good thing to do if you haven't read it. And some of the more recent works out. Jon, I think you may have actually a better overview here.

Jon Krohn: 01:35:40 Yeah, I don't know if I have a better suggestion on complex networks books but yeah, I don't really know that space at all though. Sounds fascinating. I do want to be thinking about the world more in terms of ant brains, so that sounds like one I should be picking up.

Adrian Kosowski: 01:35:56 One, one thing I, I'd say is actually when you take any networks book, complex networks, so whatever, they have these pretty covers, you know, there's always this-

Jon Krohn: 01:36:03 Network diagram.

Adrian Kosowski: 01:36:03 Yeah. If you have one of both, it usually makes for a good read. It explains like anything-

Jon Krohn: 01:36:12 You can judge a networks book by its cover is what you're saying. Awesome. And so Adrian, how do people follow you after this episode? If they want to glean more brilliant insights from you after the episode, where's the best place to track what you're up to?

Adrian Kosowski: 01:36:29 So, I'm, I have no, like, ambition, aspiration to be like an influencer. I'm happy to chat exchange with anyone. You'll find me regularly on our Pathway Discord, so as discord.gg/Pathway, of Pathway. I'm there. I mean, I'm happy to exchange with anyone from the community who has like an interest in any topics which are broadly related to reactive data processing and so on or to network science topics and how they've been treated. You can find me on LinkedIn. You can find me on Twitter, at last.

Jon Krohn: 01:37:09 Nice. All right, sounds great Adrian. Yeah, so that's a very nice offer for listeners. You can reach out to Adrian on the Pathway discord and it sounds like at this time he has bandwidth for individual questions, pick his brain. Hopefully that's still the case when this episode is published. And yeah, so thank you so much Adrian for making that offer as well as the hoodie offer going all the way back to the beginning of this episode. And thank you for being so generous with your time as well. And yeah, we'll have to check in with you on your journey and the Pathway journey again in the future.

Adrian Kosowski: 01:37:46 My pleasure entirely, Jon, thanks for having me and hope to see you at a conference sometime soon.

Jon Krohn: 01:37:51 For sure. That was an incredible episode with a brilliant guest. I hope you enjoyed our conversation as much as I did. In today's episode, Adrian filled us in on how batch processing is associated with training machine learning models at discreet intervals. This could be daily or monthly or whatever, while streaming processing allows for computationally and cost-efficient real-time ML model training. He talked about how reactive data processing allows an application to react to data it hasn't encountered before handling it seamlessly and potentially saving firms vast sums such as in financial fraud detection situations or with complex evolving systems such as the global supply chain network. He talked about how the transformer operations that transform data during data flows can be dynamic or fuzzy when they're powered by machine learning. He talked about how Pathway elected to go with a Python platform interface to be easily usable by machine learning practitioners while they chose Rust for high performance behind the scenes.

01:38:52 And he talked about the big commercial opportunity of filling in the missing bricks for extracting useful insights automatically from data. As always, you can get all the show notes, including the transcript for this episode, the video recording, any materials mentioned on the show,

the URLs for Adrian's social media profiles, as well as my own social media profiles at superdatascience.com/669. That's superdatascience.com/669. I encourage you to let me know your thoughts on this episode directly by tagging me in public posts or comments on LinkedIn, Twitter, or YouTube. Your feedback is invaluable for helping us shape future episodes of the show. And if you'd like to engage with me in person as opposed to just through social media, I'd love to meet you in real life at the upcoming Open Data Science Conference East, ODSC East, which will be in Boston from May 9th to 11th. I'll be doing two half-day tutorials. The first one will introduce Deep Learning with hands-on demos and PyTorch and TensorFlow. And the other tutorial which is brand new, will be on fine-tuning, deploying, and commercializing with large language models, including models like GPT-4. In addition to these formal events, I'll also just be hanging around and grabbing beers and chatting with folks. It'd be so fun to see you there.

01:40:04 All right. Thanks to my colleagues at Nebula for supporting me while I create content like this SuperDataScience episode for you. And thanks of course to Ivana, Mario, Natalie, Serg, Sylvia, Zara, and Kirill on the SuperDataScience team for producing another mind-blowing episode for us today. For enabling that super team to create this free podcast for you, we are deeply grateful to our sponsors whom I've hand selected as partners because I expect their products to be genuinely of interest to you. Please consider supporting this free show by checking out our sponsors links, which you can find in the show notes. And if you yourself are interested in sponsoring an episode, you can get the details on how by making your way to jonkrohn.com/podcast. And thanks of course to you for listening. It's because you listen that I'm here. Until next time, my friend, keep on rocking it out there and I'm looking forward to enjoying another round of the SuperDataScience podcast with you very soon.