EPISODE 1081

[INTRODUCTION]

[00:00:00] JM: Many data sources produce new data points at a very high rate, and with so much data, the issue of data quality emerges. Low-quality data can degrade the accuracy of machine learning models that are built around those data sources, and ideally we would have completely clean data sources, but unfortunately that's not very realistic. One alternative is a data cleaning system which can allow us to clean up the data after it has already been generated.

HoloClean is a statistical inference engine that can impute, clean and enrich data. HoloClean is centered around the probabilistic unclean database model, which allows for two systems, an intention and a realizer to work together to fill in missing fields and fix erroneous fields in datasets. HoloClean was created by Theo Rekatsinas and he joins the show to talk about the problem of fast, unclean data and his work with HoloClean. We also talk about other problems in machine learning and the engineering workflows around data.

If you want to reach 30,000 unique engineers every day, consider sponsoring Software Engineering Daily. Whether you are hiring engineers or selling a product to engineers, Software Engineering Daily is a great place to reach talented engineers. You can send me an email, jeff@softwareengineeringdaily if you are curious about sponsoring the podcast. We're also looking for writers and a videographer. If you are interested in working with us, you can send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:46] JM: SAP is a company that touches $23 trillion of consumer purchases around the world. You've probably heard of SAP, but you may not know about SAP's open source investments, such as SAP Data Intelligence. SAP Data Intelligence connects and transforms data to extract value from the distributed data landscape and embraces the best of open source technology.

SAP Data Intelligence brings together data orchestration, metadata management and powerful data pipelines with advanced machine learning enabling close collaboration between data scientists and the rest of your infrastructure teams. SAP Data Intelligence runs on Kubernetes. SAPs contribution to Kubernetes includes the Gardener Project that helps to operate, monitor, manage and keep Kubernetes clusters alive and up-to-date. To learn more about SAP Data Intelligence, you can visit sap.com/sedaily. That's sap.com/sedaily.

[INTERVIEW]

[00:02:57] JM: Theo Rekatsinas, welcome to the show.

[00:02:59] TR: Nice being on the show, Jeff. Thank you for having me.

[00:03:02] JM: Yeah, it's great to have you. Today we're talking about HoloClean, which is a data quality system that you built. There's a broad set of data sources that produce new data points at a very high rate. Describe some of these sources.

[00:03:18] TR: There are tons of sources that we are consuming today to make decisions. The easiest that you can think of from a business perspective is internal catalogues. It could be structured data that if you are a big corporation, you could have information about inventory, your transactions. It could be – So you have these types of proprietary information. It could be sources that have to do with sensors. IoT is a huge trend and the speed of collecting data there is unprecedented. We also have information that might be relevant for downstream applications that can appear in terms of structured data, videos, texts that we might be collecting. You really have a diverse type of information that is out there and contains useful data for decision making and we really need to have a unified view of all these information to make the most accurate decisions and timely decisions as well.

[00:04:22] JM: Why do data quality issues emerge from these high volumes of data?

[00:04:28] TR: Data quality problems have always been there. In fact, as I like to say, even when the first relational models came out in the 70s by code, one of the biggest considerations was how to handle null binds, so missing data or measurements that do not appear. There are

many different reasons today why you have low quality data and these might be because of noisy measurements or sensor failures. Think of wearable devices that go wrong. If you are processing structured information, for instance, streaming video or natural language, there is ambiguity. You can have uncertain extractions that are the results of other AI-driven or automated processes.

Of course, outliers is really a big issue as well. These outliers could be either because of human errors or they then correspond to machine failures and so on. You really have a diverse array of sources of error in the data that you want to process today.

[00:05:38] JM: What are the ways in which low quality data can degrade the quality of an end application?

[00:05:46] TR: It can be that the impact of erroneous data and low quality data can be severe. Let me give you an example of a very simple decision-making process that an industrial partner has. For instance, you're collecting information, demographic information about your clients and you have the information containing data about the ZIP code where your clients reside. Now, you have this information and you want to identify, recommend for instance, politics for insurance. If you have the wrong ZIP code attached to the file of your client, you cannot even perform a join between the information, the demographic information that you have about your client with external data sources that might give you information, let's say, for instance about pricing of houses and information about the neighborhood where your client resides. It can really cripple the quality that you get out of your analytics from early on.

Of course, you have also heard of cases like right now there are more and more prevalent of attacks against automated decision-making pipelines that rely on machine learning. You can either have errors that come in. You have a self-driving vehicle and your camera predicts the wrong class for turning instead of right, you're turning left. You have all these cases that can be catastrophic for your downstream applications.

[00:07:17] JM: That said, we've got high volumes of data that are being collected and we want to be able to clean this data in an ideal world. How would it be possible to clean data in an automated fashion?

[00:07:33] TR: Cleaning the data means that you have some information about what was the intended dataset that you should have received, you should have access to? You need this type of notion of correct or clean data to be accessible to you and it can be accessible in many different forms. For instance, you might have curated labeled data by experts that you can now compare and contrast with a data that you are receiving and you can potentially perform this comparison on the fly as you're receiving your data.

Of course, this is an expensive process that requires a heavy involvement of humans, but it is something that is commonly used in practice. One way to solicit this type of information for instance would be really a manual approach to cleaning your data where a user really eyeballs different values and tries to identify mistakes.

There are other ways as well where you can identify this source of – This hint, if you will, about what clean data should look like. This type of context really comes into the form of business rules that you may have or logic that describes your dataset. It could be redundancy that you have in your dataset or you might actually know that certain types of errors can also be tolerated by your downstream analytics. It's really a rich – There is lots of rich context that a data cleaning system should take into account to be able to make these decisions and identify what errors look like, how can I clean errors, and so on.

[00:09:10] JM: Do we have reasonably useful data cleaning systems? What are the options for data cleaning systems?

[00:09:18] TR: Data cleaning systems currently focus on specific problems that have to do with data quality. Let me tell you quickly what are the biggest problems in data quality and how existing systems and technologists try to address those. The very first problem that you have is that of missing values. There are lots of algorithms and there are a lot of approaches that data scientists take towards imputing these missing values. This could be as simple as performing a statistical profiling of your data, trying to impute information. Pandas offers such services for instance, and it's widely used in practice.

You may have other data quality problems that focus on transformation. For instance, you may have a date that follows the European style where you first write the date and then the month and then the year versus a US-based style of writing dates, which is first the month and then the day. You kind of want to perform normalization and fix these formatting errors, or you may want to be able to standardize really this bias and you have great tools there, for instance, Trifecta, the data wrangler tool that Trifecta is offering is the state of the art in that space. You also have other tools, for instance, like another big problem in data quality, in fact I would say one of the biggest problems is that of redundant information in terms of duplicates. You want to be able to identify mentions of the same entities in your dataset that have been duplicated. Identify these duplicates and potentially remove these duplicates.

This problem of entity resolution is one of the biggest problems in data quality and you have very successful tools there both on the open source domain. From the University of Wisconsin, you have the Magellan project. It is in the Python environment. From a commercial perspective, you also have companies like Tamer that have really taken these types of tools to the next level to industrial scales and so on. That is for really taking actions to try and fix errors.

In the data quality, you also have more recently tools that are focusing on detecting data quality issues and detecting quality issues especially with respect to machine learning pipelines. Google and Amazon actually have two open source projects. One is TFX. It focuses on data validation in TensorFlow pipelines and they are really what you want to identify as kind of outliers or inputs that do not adhere to a certain format of your dataset so that you can detect these quality issues fast.

Amazon's project is called DQ. Again, they focus on ML pipelines, and really what they're trying to do is data quality validation on the fly. This is the realm of successful open source projects. As you can see, you have projects both from industry and academia and there are more and more projects that are coming out as data quality is becoming a more prevalent problem.

[00:12:28] JM: Where does data quality fit into the overall machine learning life cycle? We've got IoT devices that are collecting tons of data. We've got these large video streams and then we've got machine learning models that are consuming that data. Where in the life cycle are we inserting the data quality applications?

[00:12:55] TR: Excellent question. In fact, I will argue that it's everywhere. The typical ML lifecycle, you can think of it as developing a model. So this is part of the design process where we want to identify what are the best features that we will use. What is the best architecture we will use? Then we move on to training this model over online live data. This involves huge volumes of data so that we get our end product, which is the trained models, and of course we deploy them afterwards to answer queries of users and perform inferences.

Data quality appears in all three stages. For the very first one, if you have erroneous data, the model that you will develop and the type of architecture and features you will select will have lots of biases. For instance, if you have lots of errors on a specific feature that you knew it was informative. Maybe the model will not pick it up as something that is helpful and that feature will be ignored. This will really influence the quality of the model that you will develop.

For training, pretty sure you've heard of what is called in federated learning specifically, backdoor attacks. If I keep giving you training data with wrong labels, the model that you will learn will be, again, completely biased. At the end, you will be deploying something that will be taking the wrong decisions.

Finally, during inference, again, that's where really you want to validate that if I give a query to a model and I could detect that many of these features that are present in this query point are wrong, then I can prevent from this catastrophic decisions being made because that point was basically significantly different than the points that the model was trained on. It is kind of an outlier detection problem that we want to solve in this phase. Really, data quality is I think – Of course, this is a biased opinion. I think it is one of the most important problems across the ML lifecycle from development to deployment of machine learning models.

[00:15:03] JM: I develop a machine learning model and I train that machine learning model and then I deploy that machine learning model to production, where it could be queries for inference. What are the points along that process where data quality management can play a role?

[00:15:24] TR: Excellent. Let's start with the training, the first part of developing a model. Imagine your initial dataset has missing values, right? Before you even start training your model,

you really want to include these values because – First of all, machine learning models are not inherently designed to handle missing values. You need to fix that problem for compatibility purposes. The way you fix this problem can have a severe impact to the type of model that you will get after training. For instance, if you do a naïve approach where you – Let's say I have a dataset that has purely numerical values and for its feature or coordinate I just go and compute the average and I use that average to impute and have a complete dataset that I will perform training on, I am injecting a bias in the final model that I would learn. I'm reinforcing the fact that more and more data points should have the smooth basic value in the feature. This might lead to suboptimal models, and in fact you can actually – There are lots of theoretical results that show that this is the wrong decision. But in fact, many practitioners perform while trying to clean data.

Formatting is another big issue during training. You want to be able to remove outliers. You want to be able to standardize your results. Normalize potentially your measurements because of different reasons. Imagine you collect this data over a period of time and something changed in the backend code. Some new release of your code was deployed, so your measurements might be sifted. You want to take into account these scenarios and try to normalize the information that you have.

This is during training, really. It's basically you're trying to guarantee that the model that you will learn will be learned over a clean data so that you minimize the type of the biases that are introduced to this model. Of course, machine learning models themselves are robust to noise. However, they tend to be robust to random noise.

If you have errors that are not systematic. If you have errors that do not come because of – Let's say you have a sensor that fails completely and starts producing lots of measurements with biased values. Then in this case, you won't have a problem if your errors are really random and sparse.

During inference time, on the other hand, you want to safeguard against really wrong predictions. One classical example is in deep learning, for instance. Deep learning models tend to overfeed the training data. The world changes, so the distribution of the data will change. If you feed new examples to your deep learning model that are a little bit away from the

distribution or the data that the model saw during training, you might have arbitrary decisions, predictions from the model that may not be correct. Depending on your downstream application, this decision can be catastrophic. Again, the typical example here would be before I even ask a model, I want to make sure that my data point over which the model will perform the inference is correct. This typically guarantees that I will get kind of a correct prediction out of my model.

[SPONSOR MESSAGE]

[00:19:01] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:21:17] JM: Let's say I'm recording a stream of location data points. I have latitude, longitude, state, ZIP code, closest business address. Give examples for how that data could be cleaned.

[00:21:36] TR: Let me start with a simple example, and let's focus on ZIP code and city. What are characteristics of a clean dataset that you would perform and then you want to imitate these characteristics in your system?

As humans, we know that because of the design of ZIP code, chances are that if I observe the ZIP code across different measurements, I should have reported the same city. In many cases, if I have the same combination of ZIP code and address, I should have the same city again or I should have the same state and so on and so forth.

As you can see, you have this inherent structure. This kind of dependencies and hidden correlations between these different measurements that are surfaces as statistical aggregate properties basically of a dataset as you keep collecting more and more samples, right? Especially, you mentioned this case of neighboring businesses. I can leverage that to identify potentially the business or the location I'm residing in, and every time I keep seeing that combination of neighboring businesses in my measurement, this provides a hint. It doesn't necessarily mean that I'm in the same location, but it provides a hint that I'm in the location that I saw in the past.

You would expect basically clean data to have this type of canonicity or structure in your data stream. This is exactly what data cleaning tools, and HoloClean is one of the tools that does that. This canonicity is exactly what data cleaning is exploiting to identify mistakes or correct mistakes. For instance, if I keep measuring – I'm in Madison, Wisconsin. If I keep measuring the ZIP codes and the state and the city for businesses in Madison, I will start finding the special pattern that appears in ZIP code. I should start seeing it in my data and I should be able to pick it up and be able to identify wrong measurements if I have for instance a city being Chicago. For

a business that is in Madison, I should be able to find it immediately. It's this canonicity that clean data has and data cleaning systems exploit.

[00:23:55] JM: I see. For that canonicity, do you need a fundamental source somewhere out there that is an oracle for your data? How do you find that canonicity?

[00:24:09] TR: There are different ways. The easiest way as you said is to have a reference dataset. For instance, for ZIP codes and cities, I may have a curated catalog that they can go and query. Apart from that, that we saw in HoloClean is that in most cases you would expect, especially as you're collecting measurements overtime, you would expect that most of these measurements will be accurate. The amount of noise that you have in these measurements will be limited, let's say, to less than half of your measurements. If you are in such a scenario and you know that you have a lot of measurements from – Let's take the example of these businesses, local businesses. If we have a lot of measurements from the location that we're interested in, this canonicity should just surface itself automatically like in the samples you have.

By using tools from statistics, you actually can discover this type of canonicity, this type of structure and you don't even have to ask external datasets. You don't even need to have external datasets to be able to discover this canonicity and exploit this canonicity for data cleaning.

[00:25:18] JM: Okay. I've got my dataset. I've got some straightforward processes for, I can imagine, manual processes for cleaning it. I can look at manual – Manually compare my dataset to an Oracle dataset, if I had an Oracle dataset source of truth. But if I want automate the data quality process and I want to add some more sophisticated ways of cleaning up my data. What can I do?

[00:25:50] TR: What you need to do is – Let me start from an example and then I will move to data cleaning. Let me draw an analogy between cleaning a dataset and that of correcting typos while we're typing information in our text editor. While I'm typing information in my text editor, and let's say I have a typo where I misspell – For quality, I remove you basically, because I was typing fast. I spell qality or something. Immediately, the text editor will suggest that, "Oh, probably what you were trying to ride was quality." So how do we get this information?

This information, you can get it by analyzing patterns between what out of the correct words in a dictionary and mistakes that humans have performed. If you see qality with U missing, really, by exploiting the fact that you have this specific co-occurrence of the other letters, you can treat this as a problem of recovering the hidden value of a random variable and try to predict basically back – You use this evidence, the fact that you had A followed by L and followed by I and so on, so you have this specific pattern. That reveals information as to what that hidden value should be.

All these spell takers that we have really rely on very simple machine learning models like naïve Bayes try and pick up statistical information from what you have observed to try and predict what you should have observed. Try to predict what is really the hidden world that you should have seen.

Exactly the same thing is what you can do for data cleaning. Imagine I'm giving you a dateset. It's cell in this dataset, meaning its value in an entry, I can treat it as a random variable. What does this mean? I can treated basically as something that it should have been generated by a random process, and instead of me seeing the correct value, I'm seeing something distorted.

Formally, what I can really do is I can try to – Given the evidence that they I have, which is the other values in that sample, for instance, and the other values in other entries in my dataset, I can say that data cleaning is nothing more than a prediction task where I want to predict, but use all the information that I have access to and predict back what the value of that cell be given access to all these information.

Really, data cleaning is nothing more than this problem of solving a statistical prediction task. You have a model that tries to imitate, it tries to learn and estimate how the clean data is generated. What are the rules really that the dataset should obey, and then you use this information to try and fill in the mixing values, if you will, where missing here is really you assume that for it cell, I didn't see it. Am I predicting back the value that I saw or am I predicting back a different value?

If I predict back a different value with high confidence, it means that I had a mistake and I potentially should consider that suggested value as the correct value. It's really what data cleaning is. It's nothing more than an instance of this type of noise. It's a model that is very well known in spell checking and other applications with a natural language processing.

[00:29:19] JM: How can we be sure that a statistical guess at what data should be is correct and how sure do we need to be?

[00:29:34] TR: Excellent question, and this is exactly that the name of the game in data cleaning. You don't want to have a tool but will always go and perform changes in your data. Nobody will trust this tool and nobody will want to use this tool. What you want in data cleaning is to quantify the uncertainty around a prediction and then give the user the option to accept different predictions with respect to the confidence level that they're willing to tolerate.

Now, how do you increase your confidence? Let me go back to the example of ZIP codes and cities. If I see – Let's say I take the ZIP code here in Madison. One of the ZIP codes is 53703. I give you an entry that says 53703 Madison, Wisconsin. If you see this entry together with another entry, that would say 53703 Chicago, Wisconsin. You're not confident about what is happening, but if you have access to a rule that saying the ZIP code, the same ZIP code should have the same city. You know that something is wrong. You're very confident with respect there is a mistake.

You're not confident with respect to what the correct value should be, because let's assume you don't have any access to external catalogues. However, if you observe a thousand topples, let's say 53703 Madison, you now have – Think of it as a voting system. You have a thousand topples that say that's 53703 should co-occur with Madison. Then you have this one single topple that is telling you 53703 should co-occur with Chicago. Clearly, your confidence increases now based on the assumption that most of the topples, most of the entries that you have seen should not be wrong.

Based on that, the more redundancy, the more repetition you have in your dataset, the higher your confidence will be. This is one of the premises of automated data cleaning. The more

redundancy you have, the higher your confidence of that prediction will be and the higher the chances that you can automate that specific repair.

[00:31:48] JM: You are the creator of HoloClean. Explain what HoloClean does.

[00:31:54] TR: HoloClean is one of the first tools to introduce a probabilistic approach to data cleaning. Traditionally, in databases, where I'm coming from the research study of databases. Traditionally, we reason about the correctness of data using logical constraints. Constraints like the one I described to you that if I see the same volume for a ZIP code, I should always have the same value for state.

For a database, these are constraints that one would specify during the definition of the schema and that all they serve is really to try and ensure the consistency of new data points that people are introducing and storing in the database. All you will do is raise a flag if you see a violation of one of these constraints.

Traditionally, people would try – In the database world, they will try to use these constraints over a given dataset now to try and reverse engineer basically this process of ensuring quality and they would try to use these constraints to fix errors in a dataset. Again, this approach would rely on considering repetition of different values together with the constraints and try to take kind of the majority vote if you will.

You have this isolated approach, and then you have the standard approach that we talked about, which is that of comparing my observations with that of a canonical dataset, the catalog dataset that I have. People were treating these two different. In addition, I have this other idea of I may have a constraint, but there are different statistical properties of the dataset, co-occurrence if you will between values, that I sued again satisfy when I'm trying to detect errors or repair errors in a dataset.

You have these different types of contexts, signals if you will, that guide data repairing. In the past, before HoloClean, people were treating these signals in a piecemeal fashion. HoloClean was really the first system that said, "You know what? Why don't we combine all these signals and try to have a unified framework that will take really any background knowledge that we have

about our data into consideration and try to both detect and repair mistakes and errors in a dataset?"

Because we wanted to be able to combine logical rules, business rules, together with external catalogs that basically act as biases together with statistical information, the approach that HoloClean follows is to put everything into a probabilistic footing, if you will, and try to cast data cleaning as a statistical inference problem.

This was the key idea behind HoloClean. When it started, it was really acting as a compiler if you will, of all of these signals into a unified probabilistic model that would allow you to take decisions, cleaning data quality decisions over your dataset. Since then, HoloClean of course has moved on beyond the compiler into a more end-to-end engine for performing data quality operations.

[00:35:07] JM: The goal of creating a statistical inference engine to impute clean and enrich the data is one description of HoloClean. You mentioned that there are these external data catalogs. Can you tell me about what are the external data catalogs that you're using?

[00:35:30] TR: This is an optional input to the system, right? If a domain expert, for instance, wants to use HoloClean with a dataset, but they do know that they have an external catalogue. For instance, in the case of ZIP codes and addresses, I do have an API that I can query to get a list [inaudible 00:35:46] information from Census. If I have a connector to that dataset, I should be able – Any data quality analysis tools should allow me to use this information.

The way that HoloClean allows users to link on demand, basically, such datasets with the inference engine of HoloClean is through what we call matching functions. Really, you want to say that, "Okay, if see an entry for a certain ZIP code, the ID that I will – The key for a query to an external dataset should be the ZIP code, for instance, or should be the street address. By allowing this type of matching key, this type of functionality for matching dependency really, we enable users to really specify how to connect these external catalogs on the map.

The catalogs vary with respect to different cases that you have. For instance, if I want to clean, let's say a database of commercial products that I want to put in my inventory, I may have a

curated catalog where I do have a certain ID for some of the products for which I have kind of golden values. You can think of it as performing a join in a database sense. Performing a join between the noisy dataset that I have a clean dataset that the user has specified.

[00:37:08] JM: Could you give an example for how HoloClean could take a set of erroneous inputs and transform it into something cleaner?

[00:37:19] TR: Think of the process of interacting with HoloClean as follows. As starting point, all you need to give the system is a dataset. We assume no access to business rules. We assume no access to external datasets. I'm just giving HoloClean my dataset.

What I can do with this, the first thing that HoloClean will do is kind of an analysis to find, to discover kind of constraints on pick up the statistical signals if you will with respect to the dataset, and it will really try to learn a model as to how the clean data should be generated, this probabilistic model that describes the clean data.

Given that, the output of HoloClean is a probabilistic version of your input dataset that for each individual cell in your dataset will tell you it is correct because I predicted the same volume with high confidence. It is potentially wrong, because I predicted a different value with high confidence. Now, if that different value itself has super high confidence, then I may suggest a repair. Really, what the user is getting back as an output is kind of a probabilistic version of the initial dataset with suggestions and confidence intervals with respect to what the correct values for each individual cell should be.

As you can imagine, given that data product, you can build a bunch of data quality services on top of this output. For instance, I can perform – If I learn this model and I have now streaming data that is coming in, I can just perform a simple check to see if my topple are valid. If I start predicting different value for different parts of my topple, this stopple means that it's not valid. I can try to correct errors on the fly. For instance, if I can detect something is wrong and I can perform a very confident prediction, I can fix this errors and then pass it to my downstream analytical engine.

The output of HoloClean, again, think about it as a probabilistic view of your input dataset. Now, if you have access to additional information, additional context such as external catalogs and domain constraints or business constraints, these are additional optional inputs that you can give to HoloClean, and these inputs will basically give kind of guidelines to the probabilistic model, additional biases if you will, as to what this clean model that HoloClean will learn should. That's how things are combined. Combine the statistical information that you have from the role dataset with optional additional guidelines as to what the clean data model should look like.

But in the simplest form and how we have seen people using HoloClean in practice, they really don't specify anything. They just give the dataset into HoloClean and they get back this probabilistic version of the dataset that they later use to either detect failures or get suggestions and recommendations as to what the correct values should be.

[00:40:31] JM: How would HoloClean be used in practice? Could you give an example of maybe a dataset and the cleaning process and the usage of the data afterwards?

[00:40:43] TR: There are different ways and that are different parts of the data analytics pipeline, but you can use HoloClean. The easiest use case to think about is upstream. I acquired a dataset before putting it into my data lake or before putting it into my collection that I would use for data analytics. I just want to get a better version of it. How do I do that? I can use HoloClean, and that's one, the most typical use case that we see.

We use HoloClean, and now you get really an assessment over which attributes of these datasets are the most clean? You get the prioritization over different cells and different topples of the datasets. Which ones are clean? Which ones are believed to be wrong? Which ones can be fixed of those that are wrong? You really can either ask HoloClean to repair everything for you so you get a better version of this dataset and you store it and then you can use it for whatever task you want, or you can even start building pipelines where you can prioritize how you can get human experts to be involved in the data validation process.

I can build a pipeline where I pass a dataset through HoloClean, and now I'm not asking experts to label and validate all the topples as it's being done in many cases in the industry, but I can just surface only the subset of topples for which HoloClean is not very confident, but is confident

there is a mistake, but it's not confident as to what the correct value should be. It allows me to build this nice prioritization basically interfaces.

You can also use HoloClean or really the model that HoloClean is learning in machine learning pipelines, for instance, before you perform inference before I issue a query against the machine learning model. If I had trained the dataset, I'd have my clean training data. I can train a HoloClean model and keep that aside and ask the HoloClean model, "Is the topple correct before passing it to a downstream machine learning model to get the prediction." You can also use it as a tool for data validation over streams, really, of data. There are different ways that you can use this technology that HoloClean is offering in the data analytics pipeline, and these are two of the most common uses that we've seen so far.

[SPONSOR MESSAGE]

[00:43:13] JM: Scaling a SQL cluster has historically been a difficult task. CockroachDB makes scaling your relational database much easier. CockroachDB is a distributed SQL database that makes it simple to build resilient, scalable applications quickly. CockroachDB is Postgres compatible, giving the same familiar SQL interface that database developers have used for years.

But unlike older databases, scaling with CockroachDB is handled within the database itself so you don't need to manage shards from your client application. Because the data is distributed, you won't lose data if a machine or data center goes down. CockroachDB is resilient and adaptable to any environment. You can host it on-prem, you can run it in a hybrid cloud and you can even deploy it across multiple clouds.

Some of the world's largest banks and massive online retailers and popular gaming platforms and developers from companies of all sizes trust CockroachDB with their most critical data. Sign up for a free 30-day trial and get a free T-shirt at cockroachlabs.com/sedaily.

Thanks to Cockroach Labs for being a sponsor, and nice work with CockroachDB.

[INTERVIEW CONTINUED]

[00:44:36] JM: Okay. Let's talk a little bit about how HoloClean works. I think some of these things will be hard to explain because I am not an expert at all in machine learning and I think these are just difficult things to explain over a podcast, but there are some – Well, first, I guess we could say HoloClean is centered around what you call the probabilistic unclean database model, and you explored this a little bit in an earlier answer. Could you explain what that is?

[00:45:07] TR: Yes. This is really a formal model that is trying to provide guarantees to how we can handle errors in a dataset. It is based on this type of noisy channel. Really, the noisy channel says, "Okay. There is a process in the background that a random process, a physical processes that I don't know and I cannot really measure that is generating clean data. This clean data is going through what is called a noisy channel. What I observe is only this noisy version of the dataset.

Just to give you an example, we have applied HoloClean and datasets from the City of Chicago where you have mentions of restaurants, and this is after inspections of like health inspectors, basically. What would be the clean model there? The clean model is telling us it has to do with how they lay out, the special layout of the City of Chicago is, the ZIP codes, the areas, where the restaurants are located and so on, and this has nothing to do with humans and has nothing to do with the collection of the data. It is the process that is generating really the clean view of this world.

However, when you have an inspector and goes to a certain address, then they transcribe the ride down in a form, the address, the ZIP code, location of the restaurant, and that the process can be noisy, because a human can introduce adders. What we get to observe is the noisy version of that. The probabilistic unclean database model is trying to characterize these two processes. We refer to the first one as the process, the intension process that generates the intended database. What we should have seen. Then we have a realization of these intended database, which can be a noisy process. What we observe is this unclean dataset and we only have access to this unclean dataset.

This view of the world allows us to formally describe many data quality tasks such as error detection or data repairs or data validation as all of them, put them into a single framework and describe all of them as probabilistic inference queries. The motivation behind this probabilistic unclean database model is to unify, if you will, all these data quality operations and bring data cleaning closer to the standard machine learning basically approach where I have to ask an automated probabilistic model about inferences.

Really, this model – Again, the main idea is to unify and help us understand how data cleaning should be tied with probabilistic models. It also allows us to get guarantees and answers as to when can we even perform these operations. For instance, if I don't have enough redundancy in my dataset, I cannot guarantee any type of automated cleaning. This analysis is something that this model, the probabilistic unclean database model, enables.

[00:48:17] JM: As you mentioned, there are some generation systems in this model. There is an intention, which is a probabilistic data generator and a realizer, which is a probabilistic noise generator. Can you describe these two generation systems in more detail?

[00:48:34] TR: Yes. The intention really captures the intrinsic characteristics of a clean dataset. If we go back to our example of businesses and demographics, if I keep sampling information of local businesses, I should really recover the co-occurrences of states, and ZIP codes, and cities that exists on the map of the United States.

We don't know exactly this process. What it is? We don't know it by design, but what we want is to actually use the noisy information that we have now and try to recover these processes that explain what is the correct structure that our clean data should have, the correct canonicity, the correct redundancy and so on.

The intention generator, the probabilistic date generator, really encapsulates all these ideas of how should the clean data look like. That realizer tries to capture and model something different, which is that of how errors are introduced. In my previous example with health inspectors and restaurants, errors are introduced in a random way because I confuse the certain ZIP code with the neighboring ZIP code, or I just did a typo because I was typing too fast and I miss the letter.

This has nothing to do with how the clean data is generated, how the word looks like, but it's just describing how errors are introduced.

This second process, if you couple the second process with the first process, you can explain really how you get to see the noisy version of the data that you have. If you learn a description of the intention model and the realizer, view it as a pipeline where I can backtrack and try to go now from the noisy version back to follow my realizer model that explains how errors were introduced to undo basically these errors. Whenever I commit to another value for my data point, this should adhere to the constraints that the intention, the intended model described. This is how these two connect basically for data cleaning.

[00:50:51] JM: All right. Tell me a little bit more about how the whole framework fits together.

[00:50:58] TR: Really, this again is kind of mathematical description of what we should do. If we go back to the example of how somebody would use HoloClean, all I'm giving you is a dataset. That dataset is really from this probabilistic unclean database model. It's really the noisy version. What the framework does internally is to first use this noisy version and use a bunch of advanced deep learning-based models to try and learn both the intention and the realizer model. The learning operations that would perform are first to use this noisy data without asking humans about any labels or any training data just to –We're trying to estimate the intention and the realizer.

Once we have learned those, then you can offer a bunch of services with respect to data quality operations, which is what HoloClean does. Data repairs that HoloClean started as a data repairing engine. Data repairs, for instance, if I know the intention and the realizer, all I need to ask the system is given the noisy version of the world that I have seen and the model that you learned with respect to the clean data, what is the most probable assignment to a cell in your database? You ask a bunch of these queries against the learned models and you get the final predictions of the system. Of course, this entire process that I'm describing is happening in the background. The user never gets to experience any of this intermediate product. All the user basically sees is give us input, this noisy database, and you get back this prediction as to what that clean intended database should have been.

[00:52:49] JM: Got it. There was a show we did recently on Snorkel, and Snorkel is I would say in a similar category to HoloClean. I guess Snorkel is more labeling. HoloClean is data quality management, but Snorkel is very much about applying the correct labels to a system, but do you have any points of comparison between Snorkel and HoloClean? Do you consider them entirely different categories?

[00:53:22] TR: Yeah, this is an excellent question. Both of them try to attack data quality problems. The data quality problems that they attack are kind of orthogonal, if you will. Just to draw an analogy think of HoloClean as trying to fix the feature values in a training set while Snorkel is trying to fix mistakes in the labels, right?

Snorkel also uses this idea of I am observing noisy labels, and these noisy labels come from the labeling functions that they have. You are observing noisy labels for your examples, and you want to find the most probable label together with a distribution for your input training examples.

HoloClean does something, as I said, orthogonal to that, which is it doesn't necessarily focus on machine learning pipeline. It is focusing on general data quality management problems where even if I have labels anywhere to define this type of redundancy, you can recover it, but it gives different types of APIs to users, and the target that we have here, the goal of HoloClean, is to reason about quality of all the individual entries that you have. Even if you have noise in your features, for instance, not just the labels, we can actually give you an analysis as to how clean this input dataset is.

Of course, if you have a machine learning model that can consume probabilistic features, and there are actually – Recently, there are a bunch of machine learning models that are coming out that consume probabilistic features and not exact. You can feed these into HoloClean. From the get go, basically the goals of the two systems are different. It's just that there is technology and ideas actually, not technology. There are ideas that are served between the two systems. The core ideas is that of reasoning about noisy information to get a better prediction and more accurate prediction about what the clean version should have been of your data.

[00:55:23] JM: How do you envision HoloClean evolving? Do you expect it to get – Or has it been deployed into production systems or is it more of a research project?

[00:55:33] TR: HoloClean started as a research project, and there is an open source version of it in holoclean.io. There are different deployments of HoloClean in both the industry and academia. One that is quite close to my heart is that of the Census Bureau in Germany, the Federal [inaudible 00:55:50] of Statistics in Germany. They started using HoloClean. That was a while back actually to harden their analytical pipelines and perform data quality analysis.

Since then, HoloClean has been tried out in a bunch of different commercial settings where the goal was to minimize human involvement in data quality pipelines, and we have seen extremely successful deployments and there are cases where we've seen that we can really, for instance, in market research, there are deployments where HoloClean was able to reduce the involvement of humans and replicate basically work that was done in months. Bring down into just really hours. It can say – Yeah, we have a bunch of deployments that the quite successful in both settings, industry and more research-oriented settings.

[00:56:46] JM: Okay. Well, Theo, do you have any closing thoughts on the space of data quality management and predictions for the future?

[00:56:55] TR: I just want to say that data quality is I think one of the biggest problems right now. It manifests itself in a bunch of different ways in different research fields. For instance, in machine learning, you see people start talking about robust machine learning against adversarial attacks. Machine messing learning when you're training models in IoT devices and personnel wearable devices that you don't want to share your information or you don't want to say are noisy and run this information. Machine learning has really put data quality into the center I think of modern analytics, but it's an old problem that only now we see successful solutions.

One thing that I would like to mention that exactly from at least the researcher perspective and from a commercial perspective, these modern advances in ML with deep learning and auto encoders, I think they hold the promise of successful commercial systems for data quality and I think it is a problem that is becoming more and more prevalent in industry and you can see that with the efforts in big tech companies like Google and Amazon, I think recently, Twitter as well,

where they really focus on making their analytical pipelines more robust, and the key problem there is data quality.

[00:58:15] JM: Okay, Theo. Well, thank you for coming on the show. It's been great talking to you.

[00:58:17] TR: Thank you very much.

[END OF INTERVIEW]

[00:58:28] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

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

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

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

[END]