EPISODE 772

[INTRODUCTION]

[0:00:00.3] JM: DoorDash a food delivery company where users find restaurants to order from. When a user opens the DoorDash app, the user can search for types of food or specific restaurants from the search bar, or they can scroll through the feed section and look at recommendations that the DoorDash app gives them within their specific local geographic area.

Recommendations is a classic computer science problem, much like sorting, or geographical mapping, or scheduling resources in a computer system. We will probably never solve recommendations. We will adapt our recommendation systems based off of discoveries in computer science and software engineering.

One pattern that has been utilized recently by software engineers in many different areas is the word2vec style strategy of embedding entities in a vector space and then finding relationships between those entities. If you have never heard of word2vec, you can listen to the episode that we did with computer scientist and venture capitalist, Adrian Colyer, or you can listen to this episode in which we will describe the word2vec algorithm with a few brief examples.

Store2vec is a strategy used by DoorDash to model restaurants in vector space and find relationships between those restaurants in order to generate recommendations. Mitchell Koch is a Senior Data Scientist with DoorDash and he joins the show to discuss the applications of store2vec and the more general strategy of word2vec-like systems.

This episode is also a great companion to our episode about data infrastructure at DoorDash. It was a lot of fun to talk to Mitchell. He knows a ton about data science and specifically the word2vec style algorithm strategies, which seemed to be quite flexible.

[SPONSOR MESSAGE]

[0:02:10.2] JM: Triplebyte fast-tracks your path to a great new career. Take the Triplebyte quiz and interview and then skip straight to final interview opportunities with over 450 top tech companies, such as Dropbox, Asana and Reddit.

After you're in the Triplebyte system, you stay there saving you tons of time and energy. We ran an experiment earlier this year and Software Engineering Daily listeners who have taken the test are three times more likely to be in their top bracket of quiz scores. Take the quiz yourself any time, even just for fun at triplebyte.com/sedaily. It's free for engineers. As you make it through the process, Triplebyte will even cover the cost of your flights and hotels for final interviews at the hiring companies. That's pretty sweet.

Triplebyte helps engineers identify high-growth opportunities, get a foot in the door and negotiate multiple offers. I recommend checking out triplebyte.com/sedaily, because going through the hiring process is really painful and really time-consuming. Triplebyte saves you a lot of time. I'm a big fan of what they're doing over there and they're also doing a lot of research. You can check out the Triplebyte blog. You can check out some of the episodes we've done with Triplebyte founders.

It's just a fascinating company and I think they're doing something that's really useful to engineers. Check out Triplebyte, that's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bytes. Thanks to Triplebyte and check it out.

[INTERVIEW]

[0:03:58.9] JM: Mitchell Koch, you are a senior data scientist and machine learning engineer at DoorDash. Welcome to Software Engineering Daily.

[0:04:06.5] MK: Thanks for having me. Glad to be here.

[0:04:08.5] JM: DoorDash is an application for ordering food and getting that food delivered to you. We did a show a while ago with one of your colleagues and that show was quite interesting, because we delve deep into the data infrastructure of DoorDash. I think today we'll

talk a little bit more about data engineering, but more on the side of how that data infrastructure is actually used to write algorithms that are useful to the end user.

A user logs onto DoorDash and they find a restaurant to eat at and they order that food to be delivered. If I'm using the DoorDash mobile app, if I'm looking for a restaurant, what are the different ways in the app that I can find a restaurant?

[0:04:55.6] MK: Sure. Yeah. I mean, first of all, yeah, I mean, this problem it's definitely an important problem that we're looking at. When you as a customer come to the DoorDash app, we want to show you the best selection we can. The first thing you'll see will be carousels essentially or clusters of related restaurants related to some theme. There will also be just a list of stores that are recommended to you. Of course, we have search as well.

[0:05:28.2] JM: Right. I can search to find food. Let's say I make a request to the search engine for burger. What happens if I just search burger on the app?

[0:05:40.4] MK: Yes. With that, what's going on is we'll recognize that burger is a cuisine type in that case and display burger restaurants. That would be different than if you were searching for maybe a particular type of item or another type of item, something like that, or healthy, those would be treated differently.

[0:06:06.3] JM: There are some elements of the DoorDash search engine that are processed offline in batch jobs. There are other aspects of the search process that are in real-time. When I enter in that search query burger, obviously the search results are being calculated and served to me in real-time, but some aspects of the search index are processed beforehand. Tell me more about the offline versus online computations of responding to a search query.

[0:06:39.3] MK: Yeah, it's a great question. We want to compute offline different attributes of the consumer based on previous orders, as well as compute offline attributes of the stores, things like how popular a store a store might be, what types of items it has and aggregate levels and talk a little bit more about that later in terms of other ways of representing stores too. Once we have those attributes of the consumer in the store, everything else is done online, because if you think about it, computing – if we have so many consumers have used DoorDash, we

wouldn't want to compute all that information for all the consumers ahead of time. It would be a little bit wasteful on the data side there.

[0:07:35.2] JM: How are my previous orders being factored into the results of the search query? Because obviously, if I've gotten a burger from a certain place before, it should impact what I'm searching for today. How does that factor in?

[0:07:51.2] MK: Yeah. Right now, so I guess it separate out the problem of the core search, we're actually inputting something, searching for something versus the recommendations that we will surface to you directly on the home page, which that's what we call the store feed. Right now, the store feed is personalized to you. Search is not personalized right now. I think that's generally a big question that people have when they're approaching the search problem is do we want search to be personalized or not? We currently do not, but we may decide to in the future if we can incorporate personalization in a good way.

[0:08:30.8] JM: Cool. Okay, so that makes for an interesting point that we can revisit later. Generally speaking, there are these two types of finding a restaurant that we'll approach in this episode. The first is that search query where you are entering in an intent of a place to find to eat. Another is this feed. If I scroll down in the DoorDash app, I'm looking through this feed of restaurants that are being recommended to me. This is more personalized.

In both of these cases, you have a matter of location, because you only want to recommend restaurants, or you only want to return search results about restaurants that are within a vicinity of that user. That leads to a sparseness in the data that you're returning to a user. Explain what that means in this context, the sparseness of the location-based recommendations or search results.

[0:09:34.7] MK: Yeah. I mean, that really comes to really the core of the problem that we're dealing with. With DoorDashing this technology platform to connect you with local businesses, it's a very different problem than the typical e-commerce problem, which is what people tend to think about in terms of recommender systems.

As we are dealing with – as you can only order from stores that are near you, that when you think about okay, typically the way people think about – the first thing people tend to think about when they're approaching a recommender system is something like collaborative filtering where you have a matrix of in this case, consumers and stores, more broadly users and items.

In this case, that matrix would be very sparse. We don't approach it like that directly. Instead, we use the technique called a knowledge-based recommender system, where instead of thinking about it as a matrix, we're instead taking in our case a consumer and a store at a time so these consumer store pairs. Based on the attributes of the consumer and the store, even common attributes, like has a consumer ordered from cuisines that the store is a part of? All those kinds of attributes enter into our models.

What we're doing is we are at a high level the way our recommender system works and there are other pieces as well that do capture related information, but at a high level how it works, it is a supervised learning system where the input is a consumer store pair and the output is a score, and that we rank once we compute for all the stores what those scores are for you, then we can rank order those.

[0:11:33.2] JM: In giving people recommendations, you want to be able to surface what is known as latent information. Can you explain what latent information is?

[0:11:45.1] MK: Yeah. Latent simply means hidden. Think about it in terms of latent states in a model. In our case, what it really comes down to is if you think about if you don't have latent information, then what you're dealing with are just the direct attributes that you can see. We might have a particular restaurant labeled as a pizza restaurant and maybe it's a pizza and American, something like that, but that's not really all the information that we want to know about such a restaurant. Maybe it is a particularly trendy new kind of pizza and somebody who likes a similarly trendy Mexican food might also like that pizza restaurant.

With that latent information, we're trying to capture some of these additional taste factors, something that has sweet things, or it could be anything. The way we do that is through vector representations.

[0:12:46.0] JM: Okay. We'll get into that in a little bit more detail with the discussion of store2vec; a little bit later. Can you just tell me more generally about how the recommendation strategy in feed works compared to maybe other recommendation systems that you've seen around the industry? How does a recommendation system for food differ from content recommendation systems, or recommendations of people you may know on LinkedIn?

[0:13:21.1] MK: Yeah, yeah, definitely. I think it really comes down to what is the product purpose of such store feed. What do people really want to get out of a feed? In some ways, it is similar to content recommendations, or any of these other types of recommendations. I think one big thing here is you will actually tend to order from the same restaurants over and over, at least a lot of people do. It's not necessarily the case that we only want to show you new things, whereas with content it's always new.

If you're thinking about Twitter for example, everything's brand new, they have to refresh their models all the time. For us I mean, I think one big piece here is even just assessing what is the overall quality of stores independent of the personalization aspect is very valuable as well.

[0:14:14.4] JM: Okay. Well, let's start to talk about store2vec. In order to talk about store2vec, I think we should first talk about the algorithm known as word2vec. This is a very flexible family of algorithms. Let's just start with word2vec. What is word2vec?

[0:14:35.1] MK: Yeah. Word2vec is this very interesting algorithm came out of the natural language processing community in around 2013 that has to do with representing words as vectors, which people have done some before but this was really the first place where people saw a lot of success with this. What they were able to do is in this way of representing words as vectors, create vectors that have very semantic representations. If you have something – you can do something like where you have the vector for the word King and you do king minus man plus woman, you can end up with queen, which I think was surprising to them as well, but this is really the basis for a lot of these other kinds of similarity vectorization techniques.

[0:15:27.8] JM: In word2vec, we can model sentences and words within an embedding space. Can you explain what it means to use an embedding space?

[0:15:39.2] MK: Sure. It's simply a specified and dimensional vector space. A good way to think about it is really as part of a neural network architecture. Lots of neural networks will have these types of vectors in them. Essentially, you start out with inputs to your neural network that might be many, many inputs. You'll want to reduce that down to something that can then be used for the rest of the neural network tasks. Essentially with word2vec, the neural network you can be predicting what is the context of a word given the word. It's this word similarity task in this case.

[SPONSOR MESSAGE]

[0:16:34.5] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify life cycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/hpe to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's enterprise software product of the year.

To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineeringdaily.com/hpe to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[0:17:57.4] JM: We can use word2vec to model these words in an embedding space. We can for example, create a vector of numbers that are associated with how close other words appear to a given word. For example, the word king might in in a large corpus of different sentences that are modeled, you might have king having a frequent adjacency to the word royalty, or to the word man, so that you can learn over time that contextually, king is related to both royalty and a man.

Similarly, you would see queen be related to royalty and woman. Just because if you take in a bunch of sentences and model the words in terms of their relationship to other words in given sentences, you can build a spatial model that gives a mathematical representation to where words are relative to other words in a sentence on average. Once we have modeled all of these different words in space and we can have spatial relationships that relate these words to one another, what can we do with that? Why is that useful?

[0:19:15.1] MK: Yeah. In the case of word2vec, it's really about if you have these – it's really about being useful for other natural language processing tasks, being able to say what word would fit in a particular context? For us with store2vec, this provides the foundation for store similarity, which we can then use to improve our personalization.

[0:19:41.0] JM: Yeah, okay. Maybe in the word2vec situation, maybe you could use it to for example, if you're building a model of natural language, you could use it to be able to process new sentences and derive the meaning from the words in those sentences and you could eventually train over time to get better at those relationships between the words.

[0:20:06.8] MK: Yeah. You can think about for other natural language processing tasks, whenever you might be having a language model or something like that where you're generating text being able to use these trained word vectors as part of those – that overall neural network can be helpful.

[0:20:26.5] JM: Word2vec can be generalized to other types of entities. We may not want to just model things in terms of words having relationships to one another, we might want a model for example stores and stores having relationships to one another. Explain why word2vec is generalizable to other types of entities.

[0:20:46.9] MK: Sure. Yeah. It really comes down to what is – to this concept of a vector embedding. Essentially, that's what word2vec is about. It's saying okay, we have these words, we want to represent those vectors and then that allows them to as you said, view them in this dimensional space, and so we can see what is similar to each other once we have these vectors.

That same concept, we'd want to apply that to other types of entities in general other than words that are important to us and how we do that might differ from the way word2vec processes text corpuses. Generally, we should be able to find some way for different tasks to achieve this.

[0:21:34.9] MK: You worked on this system for DoorDash called store2vec. Can you explain at a high level what store2vec is?

[0:21:43.8] MK: Yeah. It's really about the same conceptual issue. In our case, we have these stores on our platform. We know certain attributes about them, but really we want to be able to capture this additional latent information. Is it store of sweet things, things like that, that wouldn't be captured anywhere else in the data that we have? We want to represent these stores as vectors. That's really the high-level goal here. That allows us to then, once we have those vectors say, “Okay, what stores are similar to other stores?”
Also based on a consumer’s order history, we can say what stores are most similar to the sorts of consumers ordered from. What are the stores that are similar to a consumer essentially as well?

[0:22:36.8] JM: Right. In this embedding space, we want to make every store a vector, an in-dimensional vector. What are those dimensions of a store vector? Because we need to put these stores into this modeling space, so that we can do spatial similarities and equations between them. What are the dimensions numerically speaking?

[0:23:01.4] MK: Yeah. It’s an interesting question. Yeah, the dimensions on their own, sometimes they'll mean something looking at a particular element of the vector, sometimes not. Really, they are just what is most helpful overall in terms of determining whatever the task is that is being achieved. In the case of word2vec, it's placing these words in these in-dimensions, such that similar words are close to each other in this space, but what those particular elements are is not specified.

[0:23:38.5] JM: Just to help understand this, when I've looked at some recommendation systems in the past that are doing vector-based analysis, they oftentimes take dimensions like category of movie, or a number of stars from a given user, these things that are numerically

defined elsewhere. Or is this a movie that is in the horror category? Then if you have these numerical dimensions that have been defined, then you can derive relationships between these different vectors.

Here, we're taking more of a word-based approach. It's not like these are – it's not like we're taking categories or geographical and information. This is related to the words that are – or other stores that are related to, or other restaurants I should say that are related to these restaurants. Explain a little bit more about the corpus that is being used to model these stores, the information that is being used to create these vectors.

[0:24:51.2] MK: Yeah. I think it helps to think about what is the corpus that we're dealing with here. In the case of word2vec, the corpus is natural language text. We're looking at words that are for a particular word, we're looking at a window around it and those are the context words. For us, what we're doing is saying what stores are in the same session? What stores has a consumer viewed in the same session as this store? That is our context that we're using. Stores that are used in the same session, those will be closer together in this X space.

[0:25:32.2] JM: Okay. Let's go through the process of taking these stores and putting them into the embedding space. Describe a store for example, let's say a burger restaurant. What goes into the process of making that burger restaurant able to be put into an embedding space and what does that look like once it's in the embedding space?

[0:25:59.0] MK: Yeah. The process is that the data set that we're taking in is has to do with session. Sessions that consumers have engaged in on our website or app and all it contains is okay, these stores were viewed in this session. What we're doing is we are – we're going through those stores. If they were viewed in the same session, we're creating these essentially sentences that we'll then use really the word2vec algorithm directly as it is. That allows us to –

As we go through these sentences of stores essentially, along the way for each store, we are maintaining a vector that we'll be essentially move closer together to vectors of stores that appear in the same session as other ones as we go through this algorithm.

[0:27:01.5] JM: Let's say there's a user that sits down and they're flipping through the DoorDash app, they're looking at 4505 Meats and it's a burger place. Then they look at Osha Thai and it's a Thai restaurant and then they're looking at Smitten Ice Cream and it's a ice cream place and then they end up ordering from let's say 4505 Meats. What would that session – how would that map to a vector that we would put into our embedding space?

[0:27:32.7] MK: Right. It wouldn't map to a vector on its own, but rather that session represents essentially a single sentence that we're using in the entire training process. An entire training process, we're looking at millions of these sessions and the algorithm based on these is saying – based on this one – based on this particular session, what we're saying is okay, 4505 and Smitten Ice Cream and Osha Thai, those are all related in some way. It's essentially this particular session is making those vectors a little bit closer to each other, but there are probably some other sessions that will make those restaurants closer to other restaurants as well.

[0:28:15.3] JM: Each store has its own vector in the embedding space?

[0:28:20.2] MK: Yes.

[0:28:20.7] JM: Is that right?

[0:28:21.2] MK: That's correct.

[0:28:22.4] JM: Okay. The dimensions of each store vector are based on the frequency of other stores in user sessions.

[0:28:33.5] MK: Yes. The values of the vectors, yes.

[0:28:36.7] JM: Right. Okay. 4505 Meats has its own vector in this embedding space and it might have a high-ranking for Smitten Ice Cream, because people – you have latent information there that people like to get ice cream after they get a hamburger. Then maybe Osha Thai is a lower number, because people who like Thai food don't like 4505 Meats for some reason, or other that you have another latent piece of information. If I understand correctly, you have all

these user sessions and these user sessions are used to train the vectors of each restaurant in this embedding space.

[0:29:16.1] MK: That's right, Jeff. Yes.

[0:29:18.0] JM: Okay, great. Then you have to have the consumer vectors. Let's say I'm a user. I have my history of things that I've eaten at. Where do I fit into this embedding space and why do you need to put users into the embedding space as well?

[0:29:33.6] MK: Yeah. Yeah, really first bringing it back to what we talked about before with the recommender system overall, now we have this at the high-level there. It's a supervised learning system, but we want to capture this additional latent information. Really, we want to be able to say, okay there's – we have this input of you the consumer and a particular store and we want to know, well how similar is the store to you based on its latent information. That ends up being one feature that's brought in to the overall model as a whole.

Well what' really trying to be back here for is to say, okay what are these restaurants that are similar to the other restaurants that you word from, but may not necessarily appear that way just based on the attributes that we explicitly encoded in our database like it's a burger restaurant. To do that, the way we do that is we need to have a vector for you the consumer. The way we do that is we take your previous orders, what stores you ordered from previously and we have vectors for those. We do is average those vectors together. That vector becomes your consumer vector. Then we can say what stores are similar to your consumer vector.

[0:30:59.2] JM: If you take my consumer vector and you now have that information, you now have some degree of closeness to other stores, to restaurants that I may not have eaten at, how can you use that information to create recommendations?

[0:31:14.8] MK: Yeah. Okay, so what we're doing is in the recommendation algorithm is we are looking at all the stores that are available to you and there are features that are part of that based on attributes of the consumer in the store.

One of the features here is store2vec distance; the distance from that particular store to your consumer vector. Yeah, I mean, you can think about it. I guess, maybe a better way to think about it first, maybe easier to think about it first is to say, what if you're ranked just based on store2vec? Just based on store2vec, what would do is we have your consumer vector. We would just show you the stores that are closest to your consumer vector. Does that make sense, first of all?

[0:32:01.3] JM: Yeah, absolutely.

[0:32:03.0] MK: Yeah. Essentially, what we're doing is we're taking that ranking and we're combining it with other factors as well, the quality factors about the store and other factors based on your order of history, like price ranges you like and things like that and all that together becomes what are the recommendations that we show.

[0:32:20.9] JM: Okay. How did you validate that this was actually a useful data point? Because you don't want to roll out a recommendation system completely to all of your users and then find that it actually doesn't have any significance. Explain how you put the results from the store2vec tests into an experimentation process.

[0:32:48.5] MK: Yeah. Yeah. This is definitely very important to how we do machine learning overall at DoorDash. Really, the overall process first, just lay it out for you is when we're developing a supervised learning system like the recommendation system is, when we're developing such a system, what we want to do is be able to test it offline, so to be able to get some metric offline for how it performs. That enables us to iterate on the algorithm offline, so not actually running tests on users, but just based on data that we have offline and determine when we're making progress, we're making progress.

Then once we've selected a version of the recommender system of the machine learning system that we think is the best, or maybe even a few of those, then we will run AV tests, or multivariate tests to actually measure how it performs in the real world.

[0:33:47.1] JM: Okay. What's your bar for how well it would have to do there? How do you know that it's improving the results?

[0:33:56.3] MK: Yeah. With the recommender system, what we're doing is we want – so we need some performance metric. We need to be able to say how well these can perform? To do that, we need an offline training and test set. What we're doing there is we need to have labeled examples that say in this case, so what is the recommender system – what is this recommendation algorithm doing? What it's doing is it's saying for a particular consumer and store, will the consumer order from that store or not?

If we show consumer store, will they order or not? That's really what this algorithm is doing, so it's optimized for conversion, showing you stores that you will order from. What our training set needs to look like is it needs to show examples of these consumer store pairs and the features from them and yes or no as to did they order.

You can think about this as being a binary classification algorithm. The algorithm takes in some input and it's outputting, will they order or will they not order? Now we don't actually use the – what we're actually doing is using the probability as our score. We're saying what is the probability that the consumer orders from the store? It still falls under this binary classification framework overall.

Okay, so the metric that we will then use to evaluate – well I guess, okay. First of all, we train an algorithm. Okay, that's how we have the positive examples. I should also mention how we have negative examples of how do we get the examples of when people not order. For that, we use this concept of noise contrastive estimation. We're bringing in source that you could have ordered from, but you did not order from. Those are our examples of non-orders. The idea here is that the algorithm needs to be able to distinguish the true orders from the noise and that's what makes a good algorithm in this case.

What we do here is we can then train a machine learning algorithm based on this on the training set and then we can test it on a hell doubts, development test set or test set and that will and what we can do is we can evaluate. We have the labels for did they order, did they not order and we can evaluate things like what is – so of that your order is how many did they actually ordered from that's the precision, or how – of the, we can get the recall. Well, the metric that we actually use is the – is what's called the area under the precision recall curve and that is – that

is a metric that will not necessarily change with the magnitude of the score, so it allows us a metric to use to evaluate the ranking as a whole.

[SPONSOR MESSAGE]

[0:37:16.8] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

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

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

The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW CONTINUED]

[0:39:24.7] JM: I think it's worth stepping back here and examining the word2vec, or I should say store2vec, or maybe you could generalize to x2vec, let's say store2vec approach, in

contrast to this other approach that you might take if you were just looking at for example, the spiciness of food, or these other things that you could just get from a knowledge base. The spiciness of food or the number of star ratings that the restaurant has on Yelp, or some sentiment analysis around how good the ratings are from a store on Yelp, like these written sentiment analysis reviews. How do you test those kinds of algorithms versus store2vec? How did you come across the idea that store2vec was going to be a better source of recommendations?

[0:40:20.2] MK: Yeah. It really comes down to how do we capture latent information. Because our existing recommender system without store2vec, because of that sparsity problem that we talked about earlier, that was based mostly on these attributes, this particular attributes that we had. We really want wanted to integrate latent information in some way. These word2vec vectorization, vector embedding techniques, that provides a way of incorporating latent information. Whereas yeah, I mean, would love to hear if you have ideas about other algorithms for doing that. Yeah, that's what we found to be a path forward there.

[0:41:01.5] JM: Right. Yeah, I guess that is a pretty good source in if you think about the – what is your largest source of engagement of a user with the with the DoorDash platform, I suppose the longest period of time where the user is focused on information that would be relevant to DoorDash and relevant to that user's recommendation is as they are scrolling their feed and consuming information about that feed.

Obviously, if you could track their eye movements and have more granular information about how they're consuming information during a DoorDash session, you would be able to train something on that, but in reality the best data you have is which stores are they viewing and. It's more of a consumptive experience, they're reading it. Really the only action that they're giving you is which stores they're looking at and in what order they are looking at them.

[0:42:03.1] MK: Yeah, yeah. I mean, yeah. That's what we're doing. It's based on what do they actually tap into and spend their time viewing.

[0:42:10.5] JM: When you talk about these latent features, after you're done training this model, are you able to zoom out and say, "Okay, here is a concrete feature that we can say this is was

a latent feature, but we can actually describe what this feature is," or is it just some vague, numerical representation that you can't actually talk about with confidence?

[0:42:32.9] MK: Yeah. It probably is possible to identify some of the particular factors if that is useful for the particular application. For us, what we care most about is just the similarity itself, so we're just using the vector as a whole and looking at the distance between the vectors.

[0:42:51.4] JM: Okay. Got it. Now we did this previous show with Raghav about the DoorDash data infrastructure. We don't need to rehash that stuff in detail, but can you talk about how you used the DoorDash data platform and just what your process was for getting the information you need and how you're able to take advantage of some of the tooling that DoorDash has built?

[0:43:15.6] MK: Yeah. Yeah, so DoorDash – some of the big things that we used was our overall ETL extract transform load system, where we are able to then run these larger machine learning-based jobs. For this particular application, we're taking advantage of that to do the large-scale data processing necessary for training the store2vec vectors, training the consumer vectors as well. Then that data that is computed offline on our data platform is brought over to our search service, which then live will be able to compute the recommendations and execute the recommendation algorithm, which has been trained offline.

[0:44:11.6] JM: Were there any unexpected challenges in working with the tooling, getting this from testing into production?

[0:44:19.0] MK: Yeah. This was one of the first use cases around our large scale, rather high-compute machine learning setup on ETL. Yeah, as we as we develop this, we work closely with our data infrastructure and ML infrastructure teams on being able to establish the best practices around these types of high-compute jobs.

[0:44:46.7] JM: Cool. There is one other area of recommendations that I thought might be worth discussing, which is e-mail. Is there anything about e-mail recommendations that because we discussed how search and feed, our two environments where you could give recommendations, but these are different formats than an e-mail. How does e-mail factor into a recommendations process?

[0:45:11.2] MK: Yeah. We've from time to time we'll send different types of e-mails that we also need to display the stores that are most relevant to the consumer. Part of that is being able to run our recommendations algorithm in batch over all the consumers and so we can generate those recommendations and then could be surfaced in e-mail. It's a very similar process, except in this case the recommendations are not computed live, but they actually are computed in batch.

For that, we actually currently use spark to do that because at this point, we're really – it is absolutely a big data problem, and so it makes sense to use this distributed computing to run these types of jobs.

[0:46:02.0] JM: You've been at DoorDash for four years and you've seen a rise in machine learning tooling and obviously the volume of data. What has changed in terms of the machine learning infrastructure and how has your job changed at DoorDash?

[0:46:19.3] MK: Yeah, yeah. Yeah, I started at DoorDash in 2014; joined as the as eighth engineer. Now we have well over 100 engineers. At that point we were in just four major markets, I believe. I think we we're launching in Chicago around that time. Now we're in over 6,000 cities, almost every state in the US and much of Canada. The scale of what we're doing is definitely a lot bigger than when I started. Similarly, our technology and our systems have improved quite a bit since then.

Yeah, when I started the one of the first problems I was working on was actually our problem of ETAs, quota times, how do we estimate how long a deliver will take and be able to display that to the user. At that time, we didn't have separate machine learning systems for doing that. It was really part of our overall back-end system. Part of the evolution of our architecture is being able to break out these separate microservices, like I think Raghav might have talked about our prediction service that is now used for those types of – serving those types of machine learning predictions.

[0:47:46.2] JM: Okay. Well Mitchell Koch, I want to thank you for coming on the show. It's been really fun talking to you about store2vec.

[0:47:50.3] MK: Yeah, it's been a pleasure.

[END OF INTERVIEW]

[0:47:55.6] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source, it's free to use and GoCD has all the features thaht you need for continuous delivery. You can model your deployment pipelines without installing any plugins. You can use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your cloud native project.

With GoCD on Kubernetes, you define your build workflow, you let GoCD provision and scale your infrastructure on the fly and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It's great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you're investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]