EPISODE 695

[INTRODUCTION]

[0:00:00.3] JM: Search is part of almost every application. Users search for movies to watch. Engineers search through terabytes of log messages to find exceptions. Drivers search through maps to find a destination. Search remains an unsolved problem with lots of room for optimization. Many search applications have been built on Elasticsearch, an open source distributed search engine.

Elasticsearch is the code that powers some search as a service products offered by major cloud providers. After eight years of open source development, Elasticsearch is excellent at core search functionalities, such as indexing data, sharding and serving queries. With improved access to machine learning tools, search applications can advance in new and interesting ways.

For example, an incoming search query can be sent to an API for natural language processing before being served by the search engine. A natural language processing API can derive additional meaning from the query, adding metadata to a search query. Machine learning can also be applied to better understand how people are searching across your search index and to optimize the search index to incorporate those user preferences.

Liam Cavanagh is the Principal Program Manager on Azure Search. He joins the show to talk about the architecture of a search index, how search queries are served by an index and how machine learning APIs can be used to improve queries. This was an interesting show at the intersection of search and machine learning. I hope you enjoy it.

[SPONSOR MESSAGE]

[0:01:39.6] JM: We are running an experiment to find out if Software Engineering Daily listeners are above average engineers. At triplebyte.com/sedaily you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast.

I will admit that, though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself. If you want to take that quiz yourself, you can help us gather data and take that quiz at triplybyte.com/sedaily.

We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz scores.

If you're looking for a job, Triplebyte is a great place to start your search, it fast-tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself. I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8-bytes.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW]

[0:03:38.0] JM: Liam Cavanagh, you are a Principal Program Manager on Azure Search. Welcome to Software Engineering Daily.

[0:03:43.3] LC: Thanks for having me.

[0:03:44.7] JM: We're going to talk about search as a service, but before, I want to talk just a little bit about search more broadly, more historically, because then it'll be easier to describe the architecture of a web service built around search. Let's start with the idea of a search index, because this is the database that backs a search engine. How do you define a search index?

[0:04:09.1] LC: Yeah. The way that I think of a search index is actually very similar to what you might find with say a table in a SQL database, where just like you say, it's a place where you host your data, is the thing that you do your queries against, and unlike SQL where you would do a SQL query, typically what people are doing against in an index is they're doing what's called a full-text search, which says here are the words that I'm interested in, find me the rows or what we would call documents that are contained within that index and then we bring them back. We're very, very powerful at enabling this type of a full-text search over your content, but then allowing you to really quickly and efficiently categorize it and group it and drill down into it.

[0:04:53.8] JM: When a search index gets built, it will give different ways to access certain information that is in that search index. For example, if you're searching over images then the search index is going to be built one way. If you're searching over text documents, it's going to be built a different way. How would the implementation differ between an image search engine and a text-document search engine?

[0:05:21.7] LC: Yeah. When you look at something and specifically in the case of an image search, quite often there are multiple pieces of content that you want to leverage when you're doing that search. There is the actual image itself that you might want to display as a result, but traditionally there's a lot of metadata that's associated with it that is very, very effective, enabling you to provide that search over.

For example, maybe there are tags that describe that image, maybe it's the key colors within that image, etc. The nice part of Azure Search is that you have just like in the case of tables, you have these fields or columns where you can store all of these different attributes with this metadata that relates to that image, so that you can search over it effectively. For example, maybe that image describes the people, or the actions that are happening within that image.

You can do the traditional full-text search, yet bring in that image as a result as needed. That's actually one of the advantages of going for something like a cloud platform like Azure, is that you can leverage the best of each of the data stores. In Azure Search, we're really, really good at doing this type of full-text search as I just mentioned, but then you can persist or store that image into say blob storage and then just bring it in as needed.

One of the things that we find as you move into the cloud is that it's no longer just one store. Before where maybe you might have done full-text search maybe in one system, now you have all kinds of different options where you can pick the best of each store and bring them in as needed.

[0:07:02.8] JM: We'll talk a little bit more about the ingestion process here. If you have a set of photos that you want to be indexed over within a search engine, assuming those photos have not been annotated, then you're going to need to have some annotation system that's built into your search index ingestion process. If I've got a collection of photos, there's no metadata around them, such as labeling them as containing cats or not, there needs to be some step in the search index ingestion process where text metadata around those entities needs to be created. Describe what happens in that text metadata generation process.

[0:07:51.4] LC: Yeah. One of the capabilities that we have within Azure Search is something called an indexer, which allows you to point Azure Search at different pieces of content or different stores. What it will do is it will crawl those stores and ingest that content. What we've done more recently as part of our – what we call cognitive search capabilities of Azure Search is we've enhanced that ingestion, or that crawling process just to leverage various different cognitive services, and I'll actually explain that a little bit in Azure or through our cognitive services. These are these APIs that allow you to use machine learning to go through content and basically enrich it.

Some of the cases might be here are some text, find me the key phrases from that content, or the people mentioned, or the locations, or organizations. It actually has the ability to extract a meaningful meaning from text. Now if we look at your example, which is images, we can also leverage a lot of the vision-based cognitive services.

As we're ingesting, or as we're crawling that store images, we bring it in, but then we can use the integrated cognitive services to say, “Take this image and tell me what you find in there.” We might say, “Okay, here are the objects we found in that image. Here's a description of what we think that image is about.” All kinds of other things that cognitive services can do to understand that image that then gets loaded or stored into your Azure Search index to then be searchable. This is all part of this whole ingestion pipeline that we offer as part of Azure Search.

[0:09:36.2] JM: If we go back to just talking about text, even agnostic of images, there is still some metadata extraction that you can do if you have some machine learning APIs, for example. If we're talking about unstructured documents like long log messages, if you have a long log message, that's not something that is easy to build an index around. I mean, you can build naive indexes around, but if you have some natural language processing that runs over those long log messages during the index process, you can build more rich metadata to be able to access that. Like if you have – if you want to do some synonym mapping, or some other things.

This can differ from structured data, like a list of movies where you have maybe 10 fields for each of those movies and you have a little bit more structure to the data. Could you talk about the difference between building a search index for unstructured documents versus building a search index for structured data?

[0:10:39.1] LC: Yeah. That's a really great question, because that is one of the challenges. It's actually a lot simpler if you're coming from something that has structure, like a database or some other store. Where it gets more challenging is this case where you're saying where you have unstructured content. You could be in PDFs, or HTML files, or office documents, but you want to build out an effective search on that.

It's simple enough to crack the text on it and allow you to full-text search, but really what you want to do is get structure out of that unstructured content. As we're bringing in content, we talk through some of the things that cognitive services can do to build out structure, like or example, extracting out the key phrases, the people or organizations. As we get in to more specific examples, let me give a medical as an example; think of all the medical content in there. What is really important is if we bring that in and we say, “Oh, you know what? It’s really important if you find out what are the diseases mentioned, the parts of the anatomy, the other factors.”

As part of our ingestion, what you can do is you can leverage some things that we have built in, but you can also plug in your own machine learning capabilities. Maybe you create them all yourself, or maybe you use an open source project such as cTAKES, Clinic Text Analytics and Knowledge Extraction, which is a machine learning model that can do this entity extraction for you and plug that in. It's very extensible how you do it. You're absolutely right that a critical aspect of this is as you're bringing in content, trying to do your best to get meaning, or structure from that structured content.

[0:12:24.2] JM: When I create my search index and I want to make it available to users to access remotely over some network connection, do I want to store that search index in-memory, or on disk? How does it get accessed?

[0:12:42.2] LC: Yeah. With Azure Search, we are a platform as a service, so we try very hard to take a lot of those complexities out of the picture. We are, because we're an Azure service, there is an API that developers can use to programmatically access in search and query the index. It's also important, because it's accessible that we protect it.

We have these authentication mechanisms, these API keys that are part of that, so that whenever you query, you need to pass in the key so that you can authenticate and get the proper results back as needed. That is all part of that. To your other point of things such as memory, that's also part of the platform. We do certainly as you can imagine persist that index to a persistent store, but is also important to query that data into memory wherever as possible to optimize it as much as possible.

We tried really hard as a platform to make it simple, so that users don't have to really think about those challenges, but that doesn't mean that you can't say add in some cache layer, if you wanted to on top of that. Some of our customers do that as well. For the most part, we try to simplify it wherever possible.

[0:13:56.2] JM: All right, well we'll come back to the architecture of building a search cloud service a little bit later. I want to talk more about the basics of a search engine and how a search query gets processed, because I think there's probably some people listening who maybe they never took a course on information retrieval, maybe they never really read about search engine, so even though they're a software engineer, they've worked with a SQL database, they know what a SQL database does, they know how to query a SQL database, maybe a search engine is still somewhat opaque to them. If I enter a query into some document search engine, describe how that query gets processed and how that user gets search results back at a high level.

[0:14:47.1] LC: Yeah. The first thing to point out is that in many ways, the interaction that happens against Azure Search is not that dissimilar to how you would interact with say that SQL database. There is some client code that connects the website, or whatever is doing the search to the search engine and just like in SQL where maybe using .NET, or some other APIs to access your SQL store. Azure Search also has APIs, whether it's REST APIs, or .NET SDK. There is a developer that would be building some web application, some other client application that would leverage those APIs.

Now what happens as you mentioned when you have that search box is user enters in those terms that they're looking for and that gets formed into a query. Unlike a SQL query, this is an Azure Search query that says, “Okay, this is the location of my search service. I want to query this index as we just talked about, that's a place that we store the data. Here's the full-text query that I want to query and maybe some other parameters that would be used.” Say, I only want the top 10 results, or maybe I want to sort it, or maybe I want to tune the results in a different way. The types of things you can do as a part of the query are maybe different from SQL, but the actual interaction pattern is actually very similar.

[0:16:19.3] JM: Okay. Now if I enter in a query and I'm going to get back a set of documents, is that retrieval process using something like – we've done shows about text retrieval in the past and we talked a lot about the tf-idf search, for example. Is that query to document tf-idf process, is that still a large portion of how you can return relevant documents in response to a search query?

[0:16:46.0] LC: Yeah. That is still very much a part of most search engines. There's things like tf-idf, there's something called BM25, there's all kinds of different algorithms that enable a search engine to effectively take words and match it to the documents, or the content in your index. That's just one part of it, because it's one thing to be able to find what is textually relevant, but it's another thing to understand what is relevant to the user that's asking the question.

That's where this idea of scoring and tuning comes into place. I'll give you an example. Let's say that you were building a used-car website and you had all kinds of different used cars that you wanted to sell. If I look for a Ford Mustang, I put that into the search term, it's one thing for you to show me the documents that match the words Ford Mustang, but it's another thing if you can actually show me results where there's a car for sale, a Ford Mustang for sale that's in close proximity to me.

Or maybe if you can understand things about me, maybe I have a preference for older models of Ford Mustangs compared to newer ones. You can actually leverage all these different factors to score, or weight results differently. That's a key thing of search is it's one thing to find something that's relevant using tf-idf, or BM25, but it's another thing to be able to then leverage factors to then personalize those results.

Then it could also be for business value. Sometimes companies want to say okay, if this is relevant document, show this one higher up in the results because maybe there's higher business value to it. There's all kinds of factors where scoring plays an important part, and we're trying really hard as part of Azure Search to simplify that as well, because it's such an important thing.

[SPONSOR MESSAGE]

[0:18:50.5] 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]

[0:20:58.0] JM: When I was in college, I learned about tf-idf and it was really one of the most mind-blowing things that I learned in just all of my computer science education. Just the idea that you can enter in a query and then you could turn that query into a vector and then you have all of your documents also modeled as vectors and then you have a mathematical relationship that you can define between your search query and each of those documents. You can model them in space and then find how the query differs from all of those documents in space and then you have a ranking. It turns out to be really, really good.

Of course, if we think about the history of search engines on the internet, what we found that is that there's also a lot of latent information beyond just text distancing. This is all in PageRank. PageRank was all about how the pages were referencing each other, which in contrast to a static text document search engine, the pages are getting updated all the time, there's new pages getting added and then you have this this real-time management problem of all these documents, and then you can have additional intelligence that you could add to the search engine over time based on okay, so you have this linking relationship and then people are clicking on stuff and like oh, maybe something had the most references, but people still don't click on it in response to this query, so maybe there's something wrong with that query relative to the document, and so we can maybe add in this other layer of scoring based on how people are actually clicking on stuff. You just get into layers and layers and layers of intelligence that you can layer on to a search engine. It's really a bottomless problem.

[0:22:49.3] LC: Yeah, you're right. You're exactly getting into some interesting areas, because tf-idf is absolutely amazing, like you said. There's a reason why it's been around for so long is because it's so effective of that. What people quickly realize and what our customers found as well is that it's one thing to use tf-idf, but some of the other examples are what happens when you have synonymous words.

People search for this one word, but it's not referred to at that way in the content. How do you map those things? How do you do these things such as, I think you're talking about feedback loops, whereas actually learning from what users are doing and trying to optimize the results based on interaction patterns. Or maybe things just change, there's news articles, maybe what was relevant a month ago is not relevant now.

Taking all those things is actually hard to bring into that. One of the things that we're trying to do is try to see if there's things that we can do from Microsoft to bring in these capabilities, so you don't have to do quite as much work. One of the things that we did do is through Bing, we have this ability to handle more of a natural language processing, where it's not NLP. It's more understanding different variations of words.

Sometimes there is this concept of stems, which is to look at a word like run, runner, running. Search engines can reduce it down to its stem. Through Bing and it's work that we've done, we have not only the ability to do that, but we can get different forms of words. Mouse means the same thing as mice. We can bring in those capabilities, which are very powerful, and some of the other areas we're investigating is just through Bing and the knowledge, I don't know if you know, but in Bing, we have this huge knowledge graph that has billions, like billions and billions of entities that know relationships that if I look at that Ford Mustang, I can understand relationships to Camaros.

We're looking to leverage this data and this knowledge of Bing and integrate it so that all those hard things that you just mentioned are not things that are so hard now. If we can tie that in, we can simplify things hopefully a lot more to get those abilities you just talked about.

[0:25:10.6] JM: This is one aspect of machine learning that really everybody can be excited about. I think, when people first started talking about machine learning a lot, like maybe three or four years ago, when it really started entering in the hype cycle, I got a little intimidated because I was thinking, “Okay, now I'm going to have to learn all these aspects of deep learning and whatnot,” and that's not really the case. What the case is that you now – as long as you trust that these machine learning algorithms do work, and they – I've looked at them a little bit close – more closely, and it's just statistics and it does make sense how these different things work.

You eventually trust that these APIs are actually building meaningful relationships, whether it's I give you an image that I just took five minutes ago and you can send me an API response that detects that there's a cat in it, that's an amazing API, and that's now a primitive that I can put into my – whatever my application is for a very low cost. The same is true for building really rich relationships among entities in a knowledge graph.

Word2vec is something that's relevant here I think, because word2vec allows you to model – as long as you have a large corpus of text and everybody has a large corpus of text that they can use to train a word2vec model, you can find synonym, you can build relationships like king to queen. What is the relationship between king and queen? Well, it's that's the same position, but swapped between genders. How would you describe that relationship to a computer? Well, we now have ways of doing that and now it's just hidden behind an API, so that you can – it's abstract API.

What's amazing about that, the fact that it just gets hidden behind an API is now somebody who's working on search as a service, who doesn't have time to really think about how do you build a word2vec model can just say, I now have an API for detecting synonyms and how am I going to put that into my search as a service product.

[0:27:08.7] LC: Yeah. I think you're right. Just to continue on with what you were saying of things like word2vec, I also think that things like that are just so effective and it's actually not that hard to really start getting into that. That actually links back to what you're saying of tf-idf.

I've actually personally found that word2vec. There's also ways that you can leverage that tf-idf model, even some BM25 model that I was talking about earlier to find interesting correlations of words and terms, just like you have in word2vec that are very effective. I actually have a Github site, I'd be happy to share with your listeners as well if they're interested, that goes through some of those techniques in more detail.

I find that's correct. It's not only things like word2vec. I've also worked with customers that have created these really interesting models where it takes say loyalty card information and starts to understand some preferences of those users, and they start using that as part of the search queries as well. There's so many interesting machine learning areas that are actually not that hard to get into, but are incredibly effective. I completely agree with what you just said.

[0:28:20.7] JM: You can take a loyalty – I like that loyalty card example, or maybe if you want to discuss another example in more detail, but I find that one interesting. I can imagine a loyalty card software development company, they're probably not super well versed in machine learning, but the machine learning APIs are accessible to them to some degree. What do they need to know about machine learning and what do they need to know about search in order to be able to leverage their domain-specific data to make more informed search results?

[0:28:54.1] LC: Yeah. It's like everything, especially even machine learning is it can get very complicated very quickly, but there are a lot of things that you can start out with that are actually not that hard. For example, there are some very simple machine learning mechanisms such as item similarity that you can use, and there's actually some really good open-source technologies that you can just pick up and leverage from that, where if you start learning okay, this is the user ID, this is the item that they clicked on, if you start taking that data and you pass it into one of these items similarity functions, it'll spit out these correlations and say that, "Oh, this user, he's similar to this other user. Or this user that looked at this product, that's actually very similar to this other product," so that you can actually leverage that information as part of the presentation to the users.

Those things such as item similarity, you can just start leveraging pretty quickly and they make a big impact on the overall experience when you're building out these search experiences. Those things such as loyalty card data is typically very easy to extract and build out from.

[0:30:05.6] JM: Okay, so you're describing a collaborative filtering model. Let's say I've got a bunch of users that are searching across my – let's say I connect to a search as a service product and I'm running this loyalty card website. This is like people who are searching for loyalty cards on this website, like they buy loyalty cards on the website.

[0:30:26.8] LC: Oh, the example I was thinking is let's say that you have a grocery store and all of your customers have a loyalty card, so that whenever they purchase things, they put in their loyalty card. What you can do is then you would have information on what that user was interested in. Then, you can use that to help present users with more relevant data. Actually say, “Oh, I see that you have a preference for these types of brands, or these items,” so that you can actually really personalize that experience for the user.

The example I was thinking was more in that area. Now obviously, you have to think very carefully of what type of data you want to leverage there, but those things can really improve the experience for the user, so that if I'm searching for something, show me something that is more relevant to myself.

[0:31:15.8] JM: Right. Okay. That loyalty card data would get repurposed for, like what is the search use case? Is it like the grocery store also has an online website, for example and that they have the loyalty cards connected to the person when they search for groceries online?

[0:31:36.0] LC: Yeah. What typically happens in this case is that let's take that grocery store example. They have a website and someone comes in and maybe they log in, they have this information that knows that this is the user doing the query. If that retailer, that grocery store has that personalization information, they can then do the query, but they can say, you know what search engine? I know that this user has a very specific interest in these brands. If there's a matching item that meets that criteria, give it more relevance. You can start leveraging your own data to optimize the query that you're executing against the search engine.

[0:32:15.8] JM: What's the relationship between the collaborative filtering model and the search index and how often does that data get re-batch, or is it streaming? Can you tell me a little bit more about the relationship between those two modules?

[0:32:31.7] LC: Yeah. A lot of it depends on the actual customers themselves. A lot of times what customers will do is they will build out these – they'll have say a nightly job that goes through and says okay, here's the data that I have. Based on the current usage patterns, or current purchases by my users, here are the similarities that I am seeing. They would actually store that as part of the search index, so that when somebody searches for an item we can automatically bring back the idea that, oh, people who have purchased this I've also been interested in these. You can actually use that a bit of an up-sell as well. Just having that part of the index can be very, very effective.

You can also do it on demand. There is a lot of ways where you can say, “I just found this user that did a query. Are there any other types of users that are similar to them?” Then you can find that, get that information back to help once again optimize the queries that go through to say, “Oh, people have searched for this have also searched for that. Are you interested in these things?” It's really all about helping lead the user into areas that are interesting to them.

[0:33:43.6] JM: Do you start to pay a latency penalty when you add in more intelligence and different elements to your search when it becomes less naïve?

[0:33:53.9] LC: Yeah, sometimes it does. In those cases where I was talking about using the batching where you're actually processing the content and nightly job and you're loading it in to the index, there's really not an overhead there, other than the fact that you processed it at all and as an angling job. If you do this as part of a dynamic on-demand query, you are right that this can impact some of the latency, because first, you have to do a call to get those recommendations, bring that back, change the query and then re-execute it. The advantage of the dynamic is it's more real-time. The disadvantage is that like you say, there can be an addition of latency that comes into play.

[SPONSOR MESSAGE]

[0:34:45.8] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes, so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM. Now, Datadog has application performance monitoring for Java.

Start monitoring your microservices today with a free trial. As a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/datadog. Thank you Datadog.

[INTERVIEW CONTINUED]

[0:35:40.3] JM: There are a lot of people that listen to this show because they're curious about how some of these cloud products are built. Can you give me to some degree an overview of the architecture of Azure Search and what happens when somebody spins up a new search index?

[0:36:00.1] LC: Sure. A couple of things that I'll just point out at the start is Azure Search is actually leveraging an open-source technology called Elasticsearch. Now we picked that technology because it's a really effective and powerful search engine. We actually run that within our Azure environment on Azure VMs. Now we have our own API layer that sits on top of Elasticsearch, so it allows us to somewhat simplify the API, that queries that you do against the search engine. It also has some protection capabilities into it as well.

Elasticsearch is very much the core of the search engine. Now what we also do within Azure is we do things such as persisting your index, that data store to Azure blob storage. Azure blob storage is one of the places where we can store files as well. We use that as a persistence that they tie in very closely.

Now when a customer comes in to actually create a search service, they don't see that there is an Elasticsearch service. They don't see that there are VMs that's created on their behalf. All that they do is they create the search service. They say, "I want it in this region. I want it of this size, of this capacity. Go in and create it." We come back and we give them all the API details that they can then start using and leveraging.

For the most part, we are purely using Azure technologies, whether it's Azure VMs, we're using the blob storage, or using a number of other things that we just packaged together so that we can make managing and running these things really effective. Then on top of it, what we're doing is we're starting to bring in, or we have been bringing in a bunch of different Microsoft capabilities that you can't get just by picking an open-source technology.

I talked about some of those things from Bing that we're bringing in. There's a lot of things from our cognitive services, machine learning capabilities that were integrating directly as part of that system as well.

[0:38:09.3] JM: Where are the integration points that you work those machine learning APIs into Elasticsearch?

[0:38:18.1] LC: For the machine learning, the vast majority of that is what we're talking about as we were ingesting the content. It's going through that content, learning interesting things about it and then adding useful information to it. That's the vast majority of it. Now there's other areas that are starting to become more popular. If you think of traditional search, there's some search box in there and somebody types in some words and you get results.

What's happening now is that people are moving beyond that. They use Google and they use Bing every day and they're starting to understand that they don't necessarily need to just enter in words. They can ask natural questions, like find me a home that's in a good neighborhood in this price range. They can ask more natural language questions. The interaction pattern is different.

That is very critical to artificial intelligence, because when you get those queries, you need to understand the intent. What are they actually asking? Then mapping that into a search query so that you can get the results. People are moving towards using voice, not entering terms, or actually talking and expecting search results to come back. Artificial intelligence is becoming very important, not only in the processing of the content, but also in how you actually interact with the search engine itself.

[0:39:45.2] JM: Let's go back to talking about these high-level use cases. Do you have any more unexpected customer use cases that you've seen?

[0:39:53.4] LC: Yeah. We've definitely had a number of those that have come up. One of the interesting things when we first started the project was that there was a lot of assumptions that pretty much everybody's going to have some search box, and we talked about how that's changing in one ways. We're also finding that there's a lot of customers that use search engines and there's no search box at all.

One of the things that is surprisingly powerful of search engines is this ability to group content quickly to allow you then to filter it in. We call those facets. That's the idea that whenever you go to a retail site and you search for shoes, maybe it shows you a facet of colors, maybe a facet of manufacturers, facet of size range and price, whatever, and then you can drill in and filter on the results.

One of the things that I found really interesting is that a large number of customers actually use our search engine without a search box at all. They just show the user these categories that allow them to quickly drill down into them, which was somewhat surprising to me at the start. Secondly, I'm actually finding interestingly that people are starting to use search engines as an alternative to machine learning.

What they're using is let's say that you bring in – you have a business to business where you get this set of data that might contain business names and maybe that business name that from your customer refers to as International Business Machines, but you refer to it as IBM. Using search engines to actually find what is the best matches to content. They're actually using things like that, tf-idf and other factors to help find relevant information that is very, very surprising to me. The viability of using search engines beyond just a traditional search box is I find fascinating.

[0:41:56.2] JM: We had this show recently where we were talking with Airbnb about their search engine. Airbnb's search engine is very domain-specific, because if you search – I mean, talk about looking at search from a different angle, they look at search as more of a marketplace matching algorithm, because they want the customer to convert. Their search engine is like, does this search query from this person have a high likelihood of matching up with a property that will convert them quickly?

I don't know why that would specifically not be something that they could they could offload to a cloud service, or Elasticsearch. I think they built it actually from solar. I think their search engine is built from Solar, which I know is the roots of Elasticsearch, so to some degree it's similar and it's built off an open-source project. Maybe you know the finesse between them a little bit better.

In any case, are there any domain specific use cases where you would like to accommodate them since you're building a search as a service, but it would come at the cost of keeping the service flexible. If you wanted to do some home rental search engine, if you wanted to make that easy to build on Azure Search, that would come at the cost of making the product more complicated. I'm just wondering about the design decisions of somebody who is architecting a cloud search product and if you have to make trade-offs into in certain type of customer use cases.

[0:43:25.0] LC: Yeah, it's true. When you build the platform as a service, you have to make a certain number of decisions on how to – what you expose, what you don't expose. Because if you expose everything, then it's no longer a platform and you can't really manage it easily for the customer on their behalf. It's true with Azure Search. We had to make some decisions as well.

If you take that Airbnb case, I don't know, but I suspect they have a really large number of people that are focused purely on search and building up machine learning models for that. Now not every company can allocate a lot of people to building an effective search. We try to see if there's ways that we can offer a similar set of capabilities to what you would see in Airbnb, but maybe in more of a platforming way.

For example, when we first built the search engine Azure Search, we talked to a lot of people and we understood the types of scoring and the tuning that they do, such as what you just said, what Airbnb. We started to learn that the vast majority of customers have a very specific type of tuning than they need to do. Rather than offering every single piece of API, or customization, or machine learning hosting that is there, we could say, "You know what? Let's just provide these capabilities out-of-the-box, so that you can just say yes, I want to use that."

Now, is it ever going to be as effective as sending an army of people to build out your own machine learning models? Probably not, but it's probably pretty close. If we can offer a good set of capabilities to do a similar type set of things to what you see in Airbnb, we find that a good portion of the customers really appreciate that, and there's value to that. I think your point is very valid that when you look at platforms, you're never going to have the eact same set of capabilities that you would find in core Elasticsearch or Solar. Then, that comes at a cost of management and cost as well.

[0:45:33.3] JM: Tell me about some of the other challenges of building a managed service.

[0:45:38.1] LC: Yeah. Well, these data stores are never perfect and there's things that happen with them, whether it's corruption, other things that occur. One of the one of the interesting things you find is that when people first start with these technologies is this very easy often get started, but as you start adding content, as you start really pushing the limits of it, you start needing to think about replication and availability and all these other factors, it often gets really, really tough.

Now we have, I can't say exactly how many, but we have a large number of these Elasticsearch instances around the world. We've learned a lot through all the telemetry we've built and the automatic recovery of those systems, to make sure that we can keep these up and running. It's not always easy. We had a lot of pains when we first started to try to learn all the different ways that these stores can potentially fail. It's really important as you can imagine that we don't lose customer's data, and we keep it available as much as possible.

There was a lot of learning that we took in making sure that we can make these services available. I think that it's hard for people just getting into this to realize how challenging sometimes that can be. I mean, it's hard to express that value in a platform until users have really gone through and actually built something large.

[0:47:05.3] JM: As we start to wrap-up, I'd love to know what kinds of research you're seeing that is interesting to you, either in the area of search or machine learning that you think might be applicable to a managed service like this.

[0:47:18.9] LC: Yeah. There's a couple of interesting areas that we've been focusing on. One is in the area of what we call a custom-named entity recognition. Named entity recognition is the ability to say okay, in this content, here are the interesting like I said, organizations, people, other key phrases that were found. That's entity recognition.

The ability to do custom named entity recognition is I think one of the most interesting things from a search perspective, because every single customer is going to have their own set of content, their own entity types that they're interested in. I talk about medical maybe as diseases and disorders. If you look at legal, maybe it's people like judges, or cases, or other topics, or entities that you're interested in.

Being able to leverage machine learning capabilities to go through a customer's unique content and understand very specific entity types, I think is really powerful, because then it allows people to really plug in whatever they need to do. I think that's one factor that's very interesting. The other things that we've been making a big effort in is in the area of OCR, as well as form understanding.

OCR, which is the idea of looking at an image and extracting text from it is something that's been around for a long time, but it's never been really effective. We've actually started using machine learning, specifically neural nets, to build out our models for doing OCR and is having a huge impact. We're getting very, very accurate results that come back from the extraction of content from images.

Now, it's not just files with text that we can index and make searchable, it's actually images, whether it's a scan, or a fax, or a handwritten note that starts plugging in. Machine learning is really making all of this data much more accessible for us to both ingest and enrich.

[0:49:25.0] JM: Yeah, that's certainly exciting. I'm really looking forward to seeing this make its way into more and more consumer applications. It's taking a while, but I think it's – I don't know, we've got a bright future ahead of us.

[0:49:38.5] LC: Yeah, I agree.

[0:49:40.0] JM: Liam, thanks for coming on Software Engineering Daily. It’s been great talking.

[0:49:42.7] LC: Thank you very much, Jeff. It's been great talking with you.

[END OF INTERVIEW]

[0:49:47.9] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[END]