EPISODE 49

[0:00:11.4] SC: Hello and welcome to another episode of TWiML Talk, the podcast where I interview interesting people doing interesting things in machine learning and artificial intelligence. I'm your host, Sam Charrington.

This past week I spent some time in San Francisco at the Artificial Intelligence Conference by O'Reilly and Intel Nervana. I had a ton of fun and got a bunch of great interviews from some amazing people doing awesome work in ML and AI. I got to talk to folks like Gunnar Carlson of Ayasdi in Stanford who's applying topological models to machine learning, like Ian Stoica of UC Berkeley who's RISE Lab is building Ray, a distributed computing platform for reinforcement learning, and like Mo Patel and Laura Frolich of Think big Analytics who shared a bunch of great use case stories with me.

I'm super excited about my interviews from the conference and I'm looking forward to sharing them with you. Make sure you check back on us on October 9th to catch the full series. In the meantime I've got a great interview for this week. Like last week's interview with Bruno Goncalves on Word2Vec & Friends, this week's interview was also recorded at the last O'Reilly AI Conference back in New York in June.

Also like last week's show, this week is focused on natural language processing and I think you'll enjoy it. I'm joined by Jonathan Mugan, co-founder and CEO of Deep Grammar, a company that's building a grammar checker using deep learning and what they call deep symbolic processing. This interview is a great compliment to my conversation with Bruno, and we cover a variety of topics from both sub-symbolic and symbolic schools of NLP, such as attention mechanisms like sequence to sequence and anthological approaches like WordNet, SynSets, FrameNet and SUMO.

I'm looking forward to your feedback on this show. Jump over to the show notes at twimlai.com/talks/49 to let me know what you'd like and learned.

Finally, before we dive in to the show, the details for the upcoming TWiML Online Meet Up have been set on October 18th at 3 p.m. Pacific Time, we'll discuss the paper visual attribute transfer

using deep image analogy by Jing Liao and others from Microsoft Research. The discussion will be led by Duncan Suthers. For anyone who's missed the last two meet ups or for those who haven't yet joined the group, please visit twimlai.com/meetup for more information.

There you'll find video recaps of the last two meet ups along with the link to the paper we'll be reviewing next month. If you'd like to present your favorite paper, we'd love to have you do it. Just shoot us an email at team@twimlai.com to get the ball rolling.

Now, on to the show.

[INTERVIEW]

[0:03:28.2] SC: Alright everyone, I am here at the O'Reilly AI Conference and I am with Jonathan Mugan who's the founder and CEO of Deep Grammer. Jonathan, welcome to the podcast.

[0:03:27.0] JM: Oh! Thanks for having me.

[0:03:28.2] SC: I'm excited to get into this conversation. You are speaking later today at the conference.

[0:03:33.4] JM: Yes.

[0:03:34.7] SC: I'm looking forward to having you walk us through your presentation, but why don't we start by having you tell us a little bit about your background and how you got into AI.

[0:03:43.6] JM: Yeah, sure. I started off in psychology. Went and got my undergraduate in psychology and I wanted to understand the human mind. As my advisor used to say, the interesting parts weren't scientific and the scientific parts weren't quite interesting. I love that. We didn't quite have a firm grasp on concrete principles that we could use to really understand what was going on. I became a little disillusioned and got my MBA and a company called PricewaterhouseCoopers trained me up in computer programming and set me off in the consulting world.

[0:04:20.8] SC: Joined the ranks.

[0:04:21.3] JM: I did. Yes. Then as I was programming, I was like, "You know, you have to tell a computer exactly what to do." This might be the kind of rigor that we need if we're going to do psychology. Of course, AI is a mix of psychology and computer science, so it seemed natural. I decided to go back and get my Ph.D., but my undergraduate was a bachelor.

[0:04:45.2] SC: When did you do that?

[0:04:46.8] JM: Well, I went back. In 2003, I went back to get my master's, because my undergraduate was in psychology so I couldn't get into a Ph.D. program started away, so I had to take calculus and all that kind of stuff and all the kids. Yeah, I got my master's at UT Dallas and then got into the Ph.D. program at UT Austin and started working with Ben Kuipers.

[0:05:08.7] SC: Okay. What was the focus of your research there?

[0:05:10.9] JM: Yeah, my focus was on developmental robotics. How can you get a robot to learn about the world in the same a child does? The idea is the robot just wakes up or is born and had some knowledge built in, but not much, and it wants to build it all up from the beginning. The robot pushes objects around and learns relationships between his hand and the object and it learns how to form actions and how to build up perceptions, like, "Oh, my hand is to the left of the block. That's actually significant, because that determines when I can hit it to the right." It build it up that way and then I finish that up and graduated in 2010.

At that time AI was not hot like it is now, and it's amazing the change. I got a post doc at Carnegie Melon and I studied under Norman Sadeh and at the intersection between kind of computer science and human computer interaction. So we studied if you have this location device that gives you your location — Or excuse me, that broadcast your location to family and friends, when exactly do you want to share your location? At the time this was somewhat novel being able to share your location. There were a lot of privacy issues around it then like, of course, there still are now. The device would learn through interaction with you when you want to share based on who's asking and where you are in time of day.

[0:06:44.2] SC: Interesting.

[0:06:46.7] JM: My wife actually is a Texas gal, and when I went to Pittsburgh she didn't come with me. I flew back home every weekend.

[0:06:54.4] SC: Oh, wow!

[0:06:55.1] JM: Yeah. It was quite a deal. I flew back one weekend and eventually she says, "You know, we're not going to Pittsburg." I got a job in Austin at a small company called 21CT where we do defense contracting work for the Department of Defense, so data mining. At that job I pushed into natural language processing, because one problem I found with the development of robotics is it was really hard to get funding unless you're at a university, because it's so far off. We have robots in factories all over the place, but we don't want them staring at their navel and wondering about life.

Most of the funding, most of the push is towards robots that can do actual concrete things right now, and I'm more interested in the fundamental concepts below that. The fundamental concepts that enable a child to just grab the world and understand it.

I saw that language might be a good kind of in between. Language is very important right now. It's very useful. Chat bots are a thing. Language interfaces are important. Computers have to read tons of documents. I thought, "Okay. Language might be a way that I can both feed my family and study the stuff I care about," and then of course along the way I founded Deep Grammar, cofounded with Christian Storm, but that ties into my talk today because my talk today is about how can we go from natural language processing down to these fundamental concepts into real understanding?

Because right now natural language processing is kind of sad, because we're just at the surface. I was amazed when I first got into the field. We treat these words, we're tokenizing the words and maybe we parse, but we just have the string of tokens and then we do stuff with the tokens. The computer has no idea what these tokens mean, we just look for patterns in the tokens. In my talk I start off with a TFIDF, which you take a document and convert it into a

vector. If your vocabulary is 50,000 words, it's a 50,000 long vector and you lose the word ordering, so man bites dog is the same as dog bites man.

[0:09:14.2] SC: TIFDF is Term frequency-inverse document frequency, right?

[0:09:14.9] JM: Term frequency-inverse document frequency. The term frequency is aardvark shows up twice, then you get a two in the aardvark slot, and then you scale that with the inverse document frequency by how often aardvark shows up in your corpus.

[0:09:29.0] SC: The less frequently it shows us, the more important it is in the context of the document. That's the idea.

[0:09:34.6] JM: That's right. In that you had vector helps discriminate that document better, because it has that scaling.

[0:09:43.6] SC: It's interesting that you start off by talking about the fact that NLP is not based on a lot of inherent structure, because previous conversations I've had with folks, the general kind of understanding I've come into is that that's where — That lack of structure, meaning taking a statistical approach as supposed to linguistic approach has been the source of all of the advancements or much of the advancements in NLP over the last few years. Do you disagree with that generally?

[0:10:17.1] JM: It's definitely true that we've been able to do a lot of cool stuff. In my talk I talk about two paths, the symbolic path and the sub-symbolic path, which is the deep learning stuff that everybody's doing now. Yeah, with deep learning we're able to generalize across tokens.

One problem we had before if you said I got into my car and went to the store versus I got into my truck and went to the supermarket, those look like very different sentences in TFIDF and you had to manually go in and say truck and car are pretty similar and store and supermarket are pretty similar. You can do that for a few things, but you just can't think of all of these possibilities. Deep learning is really great for that. The Word2Vec. So everything is a vector and it turns out that, of course, car and vehicle are going to be very similar and car and truck and supermarket and store.

If you, instead of do the TFIDF, you could just even average the word vectors or you can do an RNN where the last state is meaning of the sentence you're able to really capture similarity across sentences in a way you can't do as well with symbolic methods. You still don't have any understanding there. When you do word to vec, what you're doing is you're learning a vector for a word based on the words that typically go around it. The algorithm is you go through your whole corpus and for every word in the corpus you go through one by one, you take the vector for that word and you push the vectors for the others words closer to it and you push all the vectors for the other words that aren't close to it away and then you move to the next one and you keep doing that over and over again until you've converged.

That's great, but it only captures what people say. Most of the knowledge that's needed to understand language is so obvious that we never mentioned it. That kind of stuff just doesn't show up in word vectors. Even when you get this vector at the end, it's still not clear what to do with it. You think about some of the biggest advances have been or most exciting ones have been in machine translation. The machine still has no idea. It's just spitting out tokens. It encodes it with the encoding RNN and then the decoder, it spits up the next word based on the previous state, the previous words, and then if it has attention to all of the previous encodings in the encoder, but it's just a softmax spitting out tokens. It doesn't have any understanding of what it's doing, which is in some degree why it's so applicable in so many different domains. You can create a parse tree with it. You can even encode a picture into a vector using the CNN and then run the decoder and that's how you get this capturing work that's really exciting. There's still no understanding there.

You end up this vector, so now we're on the sub-symbolic path, but what can you do? The next thing that people started doing was well — So attention, what you're doing when they added attention to the encoder-decoder method, when you're about to generate a word in your translation, you're looking at all of the previous words in the sentence or their encoded representation, the hidden state representation there'll be of the sequence.

What it's doing is it's looking at facts about the world to figure out which ones are relevant for generating the next word. What people started doing was they said, "Well, what if I just feed it in a story? I could feed it in a story where like Tom went to the store, Tom came home, Tom picked

up a jar, Tom went to the airport, and now the question is; where is the jar?" If you feed it enough of these stories, it's pretty amazing that the computer can answer it, it's at the airport, presuming that you never put down this jar, that you just carried it with you for the rest of your life.

That's really cool, but you have to generate these stories automatically and the reason you have to generate it automatically is because you need so many stories that it needs to be able to find these statistical patterns underneath.

[0:14:26.1] SC: This mechanism that's enabling this is attention. Can we maybe double click on that to talk about how that's implemented to maybe get a — I've heard attention come up a bunch of times, but I haven't dug into it in any level of detail and I'm wondering how that manifests itself in some of these deep networks and stuff like that.

[0:14:47.4] JM: Yeah. I'm thinking that Google just came out with this Tensor to Tensor thing which is — I'm thinking of how they do attention. You have a set of keys and a query and keys and values and what you're doing is for some query you're looking at all of the keys define the most similar key and then you take that value and the similarity between the query and the key is the weight that you use for the value. You end up doing a weighted average of the values.

[0:15:19.3] SC: Is that implemented in a neural network or are we talking about there's an external structure, like a database or key value store or something that your —

[0:15:26.4] JM: It's all neural network. When I say key and query and value, these are all vectors.

[0:15:31.9] SC: Okay, got it.

[0:15:33.7] JM: In the sequence to sequence model, what you're looking at is these maybe that the keys and values are one in the same, but you're looking at — Your query is my current state when I'm trying to generate. You could have; I went to the store, and then you're translating to Spanish, fui a la supermercado, and then when you're trying to — My great accent. When you're trying to generate supermercado, you look back at I went to the and you look at those

encoded representations along the way and you take your state at supermercado or at the state before your generate supermercado and you compare how similar those are. Then you take the weighted average of those values and then that value comes in to where you would normally generate supermercado and that value is taken into account. It's just another vector along with the vector for the previous state in your decoder and the vector for the previous word you generated. Then for each vector you have another matrix which you multiply it by and then you add those things up and you throw that in the softmax and then that's your output.

A neural network at each point is generally a — Or very often, a multiplication of a matrix by a vector, and then you put some nonlinearity on the result.

[0:16:56.3] SC: Okay. Attention basically is you're storing — You're kind of storing up these vectors and referencing them from the PASS essentially and including those in your end calculation.

[0:17:11.6] JM: That's right. What they're doing in the story generation, or excuse me, the story question and answering is they're encoding the parts of the story as vectors and then when they want to answer a question, they go back and look at the parts of the story and figure out which parts of the story are most relevant to answering that question and they do a little computation on top of that and that's where your answer comes from.

[0:17:34.1] SC: Interesting. Are there limitations to the amount of memory that you're able to refer back to?

[0:17:41.2] JM: Generally, there's not limitations in the amount of memory but you're generally taking a weighted average and you do that because if you just take kind of a hard attention, then you can't do the back propagation as well. You take a weighted average, so things kind of get watered down a little bit. I don't think that's a huge problem. More of the problem is it's just a very simple mechanism and it can only do so much.

[0:18:07.7] SC: I think that's where you were going before I kind of interrupted you to push into this. Attention is starting to approximate things that look and feel like meaning but it's still not quite there.

[0:18:18.8] JM: Yeah, you're going back over your previous experiences and saying, "Oh, this one is relevant," and then pulling it in. Yeah, which is cool, but the robot doesn't have any previous experiences. This story generating, or this story question answering systems are really cool, but there's no built in knowledge. When we answer questions about stories, we bring a whole lifetime of knowledge. These all start from scratch.

What we need to do, I think the next step on a sub-symbolic path, is we need to have systems that interact in our world with the objects and relationships in our world. You can imagine like a little robot that can pick up objects and move them around and then it knows what a bottle is because a bottle is partially the hand fixture that it needs in order to pick it up. A bottle is partially that if it knocks it off the table, it breaks. A bottle is partially that turns it to the side, water comes out. All these things are part of the definition of bottle, and so when it's pulling out memory, it's not just pulling up parts of the story, it's pulling up huge banks of things that it just experienced before. Then you can make inference from that that you wouldn't be able to otherwise.

We don't quite know how to do these advanced inferences based on experience other than the kind of basic models we have now, which is like sequence to sequence and CNN and some other ones, but it's going to be exciting to see. One of the things I really enjoy about the deep learning is every time a new configuration comes out or a new one that goes in the zoo, I'm like, "Oh, cool! We're getting a little closer."

Envisioning the brain, there's just thousands upon thousands of different kinds of configurations of neurons and at least to some approximation, one of them might be a sequence to sequence, and another one might be a CNN, but there's hundreds more that we haven't discovered and then it'd be cool as we get better and better with each new one.

[0:20:17.7] SC: I think most of what we covered now, I mean it sounds like a lead up to a specific area of research or interest that you have that kind of promises to help address this issue. Where do we go from the sub symbolic, maybe another way to ask this is, is it your observation that a more symbolic approach is kind of the answer to the ills of the sub-symbolic approach, or do you think the path forward is still sub-symbolic, but extending it to incorporate more understanding?

[0:20:52.9] JM: I don't know. Yeah. In my talk I cover both approaches as if they're separate approaches, and there's been surprisingly little overlap in the approaches.

[0:21:07.1] SC: We've talked about the sub-symbolic mostly. Should we take a few minutes talking about the symbolic stuff and what's happening there?

[0:21:12.4] JM: Yeah. Sure. We can do that. Yeah, we were talking about a TFIDF vector and that it froze out word order, but you can do a lot of stuff with it. You can say, "This document is [inaudible 0:21:23.5] this other document." You can even throw it in the machine learning classifier and do sentiment analysis or document classification. That's pretty neat. I mentioned sentiment analysis, so the next step in sentiment analysis, getting a little closer to actual meaning is a sentiment dictionary.

Often, you'll have a dictionary that says, "Okay, the word terrible has a negative sentiment, and the word good has a positive sentiment," and you have some simple mechanism that says, "Well, not terrible. You have to inverse, un-verse the word." That can get you pretty far, but for each symbol now you're going into your dictionary and you're assigning some very simple meaning. That's kind of the first step to assigning meaning, or you could consider it one first step.

There's also a whole set of representations that people have built. When you build a representation, what you're doing is you're taking symbols and you're creating relationships between symbols and then presumably if you set up this symbol system, you can map what people say to the symbol and then you could map what the computer should do based on whatever symbol got lit up.

Let's say you're a tire company and you want to watch Twitter to see who you should try to sell tires to, you could mention anybody. You could either set up symbol system where it says, "Okay, a car has tires. A truck has tires. Toyota is a kind of car, and therefore anybody mentions a Toyota, it links into tires," and that helps you when you take a sentence, you say, "Given this sentence, can I find tires linked anywhere in there, even if I don't mention tire explicitly."

Of course, this is kind of brittle and there's been a lot of work in setting up these symbol systems. The most famous is probably WordNet, where you have these set of synsets which is a synset is a group of words that all mean the same thing. It's like a meaning. Vehicle might be synset. In that vehicle you'd have — Maybe car is one synset. Then car you'd have motorcar, car, and it could have car in all different languages, but it means car, then you'd set up relationships between these things, like car is a kind of vehicle and then you would have sports car as a subset of that.

WordNet is really popular and it's really good. It kind of gives a sense of a definition for most words and you can also have a word be in two different synsets. Bank would be in a synset for riverbank, but also a bank where you deposit money. That's one.

Another one is FrameNet. FrameNet builds up a little bigger situation. WordNet is about individual words. FrameNet is about situations. One example is the frame comers by, which means somebody buys something from someone else. That frame is triggered by some set of keywords, like bought, purchase, sold, and when that frame is triggered, what FrameNet does or at least an implementation that uses FrameNet goes in and tries to find the roles. Who is the buyer? It could be Bob bought a car from Tom. The buyer is Bob, the seller is Tom and the thing purchased is car.

You've converted this sentence into a frame with roles, and now that's machine understandable. It's kind of nice because you move up from individual words into kind of meaning of situations. FrameNet doesn't go very deep. FrameNet doesn't say, "You don't have the fundamental things going on," like forces and — Well, there's a little bit of that, but you don't have the things that a child knows, a very young child, and it turns out in AI, that's the hardest part. We started off thinking that chess was the pinnacle of intelligence, and now it turns out that picking up a bottle of water is really hard. Who would have thought?

All the things, they say all of the things a kid knows by 3 or 4, if we could get those in a computer that would just be amazing. That's what I would really love to try to do. If we're going to build that with a symbol system, we have to go deeper. One symbol system that does go deeper is SUMO. What SUMO is it's a full anthology, meaning it goes all the way down. You look at cooking. What is the word cooking mean? Cooking is — I don't remember the exact

thing, but cooking is a process which isn't a thing, which isn't entity. It goes all the way. It takes it all the way down.

That's really useful, but what we need to do now is figure out how we can get things like SUMO tied into WordNet, and there has been some linkages. SUMO already does tie in with WordNet, but how we can get all these different representations together, because what we want to do next is build — If we're staying in symbolic land, build a causal model of how the world works. Here at this conference, Josh Tenenbaum yesterday was talking about that.

An entity needs to understand that when it pushes a table all the things on top of the table are going to move. If you try to put that in logic, it's hard, and you need like a model where you can just read it of the model. In some sense, FrameNet is kind of like that. If there's a frame where Bob sold a car to Tom and then you ask, "Who has the car afterwards?" It's Tom. You could just read it of the frame or you can have that associate direct with the frame. You could put that in with the frame.

What we need to do is build deep causal models that go all the way down to these things called image schemas. Image schemas are the language independent concepts that we use to understand everything in our world. Like often Johnson and these kind of kinds, Manler. She's a psychologist. You put a development psychologist. One is containment. When you have a bottle of water, the water is contained in the bottle, which means that if you move the bottle, the water goes along with it. Another is support. The bottle is on a table. You need these concepts before you can understand language, because language understanding is built on all these stuff. When we talk to each other, we never say these things.

One of my — one joke I like to say is you can imagine a romance novel where there's a table in between two lovers and the man pushes the table aside, and in a novel they would never say, "As he pushed the table aside, all the objects on the table moved because there was force pushing down." That's just not in there.

[0:28:09.4] SC: There was a sound of the legs scrapping across the floor and that produced gashes in the floor.

[0:28:15.4] JM: Suddenly the objects were at a new location.

[0:28:19.8] SC: We make a lot of assumptions when we talk.

[0:28:21.2] JM: We do. If we didn't we'd never get anything done. What we need to do is build up causal models of the world on to which we can put these symbols that we define.

[0:28:32.2] SC: I can't decide if it would be fun or extremely boring and tedious to run these models in reverse and generate that romance novel.

[0:28:42.8] JM: It'd be the worst novel ever. It'd be awesome. It's like you have different additions of books, the big print, and then this is the computer addiction. Those are basically the two paths. To get from where we are now in natural language processing, which is just working at the symbols either with the sub-symbolic where we're able to learn vectors for these individual things, or the symbolic where we just write down what they are in the computer that can really understand, because it understands the fundamentals.

It's hard to say where to go from here, because one problem is you need to find a commercially viable need for the simplest possible common sense knowledge. We all have chat bots. They're everywhere now, but they require already way too much knowledge to be good. If you go out of — If you stay within a particular domain, you basically just hardcode everything and you can have chat bots where it's learned using sequence to sequence models, but that's just gibberish back and forth. It's no different from Alyssa really.

If we have a chat bot that actually is general, if you ask things off script and can answer your questions, we're going to need these fundamental concepts. In fact, one of my dreams is to build a chat bot for children that you get at age 3 or 4 and it lives in your mom's phone and it teaches you concepts about the world, and it's also your friend and it learns about you. The cool thing about being a teacher is that if it teaches you, then it knows what you know. Then it explains other things to you. It can explain too many things in terms you already understand. It can also make things interesting, because let's say it knows that your favorite animal is a giraffe and then it can say — When it's teaching you math, it can say, "If you have six giraffes and you buy two more, how many do you have?" That's the kind of thing that engage parents too. It

would be cool to do that in an app, and I have this dream that you put that in for children and you have your developers frivolously working behind the scenes making better and better and better technologies so that when a child gets older, the app just turns into the operating system for the child. The child now uses this app to interface with its whole world.

Since the app has been with the child since the beginning, the app really knows the child and so it can be the ultimate and customized. Then as an adult, when it's guiding me through how to fix my dishwasher, it knows that I know nothing and it literally has to tell me, "Lefty loosey, right tighty, remember." As supposed when you on Wikipedia or on the web, you just have no idea.

Yeah, and then even when you're old, if you become — Your faculty start to go and if you're standing in the kitchen and you can't remember how to make coffee, the app can then be in the cameras in the room and say, "Hey, you make this coffee this time of day. The filters are covered over there. That's the first step," and then guide you through and maybe we could stay independent longer.

[0:31:50.0] SC: What a great vision for an app.

[0:31:51.9] JM: That would be awesome.

[0:31:53.0] SC: Now, how much of that is Deep Grammar trying to take on?

[0:31:56.3] JM: None. Yes. Deep Grammar is trying to take on — When I write, I make a lot of really dumb mistakes. It's just the human in me. I think one thing in my hand is just to output something different. I've always been amazed that grammar checkers couldn't capture that. Spellcheckers came along and they were amazing. They really — I don't know how many people around remember days before spellcheckers. It was a huge advance. I always told my teachers, I'm not going to have to know how to spell. It turns out I was right about one thing. Very few times was I right, but that I was right.

The grammar checkers, I was always — They were in Word for a long time and they just would miss obvious wrong stuff and it really bugged me and I always thought machine learning would be the way to go. I started with n-grams, which is sequences of tokens. That's a few years ago

and it turns out, if you think about it for five minutes, it turns out that for n-grams, they're like sequences of three or four. You take the "I went to the", that's like four words and then you have a distribution over to the next word.

If you write a word that is not in that distribution or it doesn't have a lot of weight in that distribution, like donkey, but it's similar to a word that should have high weight, like store, although donkey and store aren't similar, then you probably made a mistake. If you say, "I went to the stored." That's a mistake. It should be obvious. Stored is very similar to store, and stored is going to have a very low probability.

The problem with n-grams is you can't — It's that similarity problem we talked about four, because I went to the store, I went, I drove, I meandered, I walked, all those are very different to an n-gram probability thing, and so in order to train such a thing, you would have to have seen all these things and you'd have to store the vocabulary size to the fourth to store all these probability.

When I started working in deep learning and I said, "Oh, sequence to sequence is the way to go for this. You encode this thing and then you decode it," and you get the power of deep learning, that power we talked about before, that similar words are going to have similar vectors, and similar sentences are going to have similar vectors to other similar sentences. First thing is, "Oh, okay. I got to write a patent on this. This is going to be how we're going to do grammar checking," and that's how we got started. Yeah, what we do is we encode it and then we decode and if the thing we decode is different from what you wrote, then there's a problem. Especially if what you wrote is similar to something that would have high probability.

[0:34:43.1] SC: How do you capture that similarity?

[0:34:45.9] JM: We use just typical — The easiest thing you can do is like what [inaudible 0:34:51.0] distance, which is add a distance on the letters. You can do that with similarity, but there's also a bunch of other little similarity things we do that we take advantage of a lot of the acquired knowledge over the years in grammar.

Yeah, we kind of have a sophisticated similarity measure. In addition to sequence to sequence, we throw the kitchen sink of deep learning at it, a bunch of different CNNs and stuff. We've got pretty good now. Sometimes it still fails in a way that's disturbing, but if you make a mistake like the wrong version of there or to, too, two, it's really good at that. If you catch it better than anything.

There's a lot of different markets. The biggest market as you might imagine is English as a second language, but — And we get people all the time emailing me, "Please get this thing going. Please. Please. Please." The English as a second language is particularly challenging, because sometimes when you're not familiar with a language, what you write is so far from correct that the machine just can't — It just doesn't know where to start. That's a particular challenge, and then sometimes there's even a bigger fundamental challenge that the whole sentence has to be rewritten. The only way to do that is to understand what the person said. We've been talking this whole interview about how computers just can't do that. That's going to be a problem for a lot of time to come, but we can finally — These dumbs mistakes, I look at them and I can't believe that the grammar checker didn't catch that. Now, it can catch those. That's really exciting.

[0:36:22.8] SC: Do you offer this as a service for folks? I use a service for writing called Grammarly, which you may be familiar with, that does a decent job for some things, some aspects of their implementation are kind of bad. Just the UI, user experience is kind of wonky, but I can imagine that as a go to market model. I can imagine more of a platformish approach where you're offering APIs to developers to build things around. How are you guys going at it?

[0:36:56.2] JM: Yeah, we are in the process of trying to decide where exactly we're going to focus, because Grammarly is now really big. They got a lot of smart people working and it's going to be hard to go head to head with them. I think we've got some ideas that are really good, and I think we do some things better, but they're just hiring like crazy.

[0:37:16.8] SC: You can start by plugging in to any of the editing apps on the Mac, which they don't support, or I don't think that they plug in the Google Docs or anything like that. There are some holes there. What kind of mote that gives that's another issue about it?

[0:37:31.4] JM: Also, Grammarly is pretty expensive. I think it's like $10 a month and there's a lot of people around the world who really need this, but $10 a month is a lot of money. If we have a service that's really good, better than things that used to be around before but maybe we don't have all the bells and whistles of Grammarly, especially if we can fix those things that are really hard for a computer to catch.

Grammarly, looking at what they've done, it looks like they spent a lot of time implementing a lot of rules, like comma rules. We can catch the subtle things that just pure learning can find, and that's what a lot of people need, because when you're English as a second language, you don't have the ear that we do, ear for the language.

[0:38:11.3] SC: The ESL market is really interesting. Language is a hobby of mine and one of the apps that I've been using recently that I really enjoy is this app called Tandem. It basically allows you — It's kind of a global language learning community, and there have been a bunch of these, but it's the best implemented by far. You basically go in this app, you tell it what language is you're learning and it will match you with people that speak those languages natively that are trying to learn your languages that you know.

You'll get in these conversation with folks and depending on their level, you've got the — I think the interesting conversations are when folks are beyond the, "Hey, I'm going to Google translate everything I want to say," because you know the failure, like you can spot those really quickly. But then there are folks that know enough English they're just typing what they think is right and sometimes it's a little hard to decipher, but most of the time you can kind of get what they're trying to say. They're just not saying it wrong. If your stuff was plugged into this process as like kind of a side channel, a trainer or a coach or something like that, I think it would — The big challenge for language learners is like decreasing the cycle time of learning and iteration and accelerating the process. Something like that could be really interesting.

[0:39:37.6] JM: Yeah. I hadn't thought of that as like a coach. You said this and maybe change it to this other thing. No, that's a good idea. Another place that's like video transcription is big now and a lot of times you'd have to pay a human to do it. It's done automatically, but then you need to pay a human to make sure it's done right, because you get this text out and sometimes it doesn't quite hear. That's very much like a grammar correction problem. We could do that.

Yeah, we're trying to decide what exactly, what niche we should go, make it cheap or make it an API, do transcription. Maybe there's some publisher that have reached out to us, they say, "Look. We send out all these books and we have to pay people to go in and read each one." If we use you guys, then we have to pay them — We could have them do more books per person, because they would have less tedious stuff do to. You'd catch the obvious stuff. That's another option. We're kind of standing at the crossroads right now trying to figure out what we're going to do with it.

[0:40:38.9] SC: Does the technology get into or give you the ability to address stylistic issues as supposed to correctness?

[0:40:47.9] JM: It kind of does both at the same time, but it doesn't help you rewrite things. It's basically going to help you write the way it was trained. We started out training on Wikipedia, but then it wanted to fix everything to be very Wikipedia.

[0:41:09.5] SC: The thought that came to me was the artistic style transfer stuff where you take this picture and make it Picasso-esk. I love to take my writing and make it in the form of some other author.

[0:41:25.5] JM: That would be cool. It doesn't work as well in language as it does in vision, because in language you're making a set of discrete decisions. In vision you have pixels which are much more amenable to small gradients, and that's why they've had such huge success with vision. Language is harder. They're starting to get some work in that area. Some of that new stuff is applying GANs to sequence to sequence models. What you do is instead of using the cost of generating each token on what you're training in the decoder, you use some other measure of the sentence. Again, it would be the probability based on some discriminator function, the probability that is generated by the computer or by a human. Then you have to get that answer back into the system so it can learn and that's usually done like with reinforcement learning. That's not very efficient right now for language and it kind of works and there's a lot of advancements, but still got a ways to go.

[0:42:26.4] SC: Okay. I just finished a report on industrial applications of AI. It ended up being like 30 pages, and I love to put that through like the Hemmingway translator.

[0:42:41.3] JM: Or Cormac McCarthy or —

[0:42:45.0] SC: That will be awesome.

[0:42:45.8] JM: Yeah. I think we'll get there.

[0:42:47.5] SC: I can assure you the sentences. I think for Hemmingway will be a lot shorter. It's going to be a run-on sentence kind of guy.

[0:42:55.3] JM: Yeah. That would be great. There is some of that. If you train the system on Hemmingway, it's going to want to generate tokens that are Hemmingway-ish. You feed your sentence in and it's going to translate it and it'd be shorter and something about a fish probably.

[0:43:14.6] SC: Nice. Awesome! What's the best way for our folks to kind of keep tabs on what you're up to and follow along as you guys iterate on this model and figure stuff out?

[0:43:25.4] JM: Yeah. We have a website, deepgrammar.com. On that website, you can try it out. Type in a sentence, it only does one sentence at a time right now just because we have a cheap serer up on Amazon. Then you can join our mailing list, and I tweet my life out at Twitter @jmujim.

[0:43:42.7] SC: Okay. Awesome. Thanks so much. It was great chatting with you.

[0:43:45.7] JM: Oh, great. It's been fun.

[0:43:46.9] SC: Awesome.

[END OF INTERVIEW]

[0:43:49.9] SC: All right, everyone. That's our show for today. Thank you so much for listening and, of course, for your ongoing feedback and support. For more information on Jonathan and the topics we covered in this episode, head on over to twimlai.com/talk/49.

If you like this episode or you've been a listeners for a while and haven't yet done so, please take a moment to jump on over to Apple Podcast or your favorite podcast app and leave us that five-star review. We love to read this and it lets others know that the podcast is worth tuning into. If you've already done this, then thank you so much. We greatly appreciate it.

One last note, you've probably heard me mention Strange Loop, a great technical conference held each year right here in St. Louis. I'll be attending later this week and I encourage you to check it out. Also, the following week, on October 3rd and 4th, I'll be at the Gartner Symposium IT Expo in Orlando where I'll be on a panel on how to get started with AI. If you plan on being there, send me a shout.

Thanks once again for listening, and catch you next time.

[END]

