BORIS KATZ: When a scientist though approaches a complex phenomena or when an engineer looks at a difficult problem, they usually break it up into pieces and try to understand this phenomena and solve them separately. Understanding intelligence is one such very, very complex, in fact, extraordinary complex problem, and over the years, this divide and conquer approach produced a number of very successful fields like computer vision, natural language processing, cognitive science, neuroscience, machine vision, and so on.

But we need to remember the most cognitive tasks that humans perform actually go across modalities, which is they span these established fields. And the goal of our thrust is to bring together techniques from all these fields, create new models, and for solving intelligence task, we also would like to understand how these tasks operate in the brain. So I will start with one task, which is scene recognition.

What does scene recognition involve? Well, machines, in order to recognize a scene, a machine needs to do some type of verification. Is that a street lamp? It uses the detection of other people in the scene. It needs to do identification. Is this particular building, Potala Palace, somewhere in Tibet? It needs to do object categorization. Look at this image and tell me where are mountains, trees, buildings, street lamps, vendors, people, and so forth.

We should also be able to recognize activity. What is this person doing? Or what are these two guys doing here?

Well, currently, our machines are pretty bad at all of these tasks. I understand that there has been quite a lot of progress recently made in machine learning. And I've also seen some claims that machines perform better than humans in some visual tasks. However, I think we should take these claims with a grain of salt.

First, there is nothing amazing about machines doing certain things better than humans. People did it over millennia. Humans needed tools to build pyramids. They build tools to carry heavy things, to lift them, to go faster, and so on.

Well, you'll tell me, no, no, no, we need, we are talking about intelligent tasks. Well, for $5 you can buy a calculator that multiplies numbers much better than you do. For $100, you can build a gadget that has a huge lookup table and plays chess much better than any of you.

So when you hear that a computer can distinguish between 20 breeds of dogs, or something like that, better than you do, I don't think you should assume that the vision problem is solved. Well, understand, I'm not saying that because we have a dramatically better solution. Not at all. My point is that the problems of real visual understanding and real language understanding are extraordinary hard, and we need to be patient and try to understand why and eventually find better solutions.

So back to visual understanding problem. So as I said, machines are better at these things. But humans are absolutely awesome. You have absolutely no trouble to do verification detection, identification, categorization. You could do much more than that. You could recognize spatial and temporal relationships between objects. You can do event recognition.

You can explain things. You can look at that image that I showed you and tell me what past events caused the scene to look like it is. You can look at that scene and say what future events might occur in that scene. You can fill gaps. You can hallucinate things. You could tell me what objects which were included in that scene which I've barely seen, actually, there might be present there and what events not visible in the scene could have occurred.

So why are machines are falling short? Well, in part, our visual system is tuned to process structures typically found in the world, but our machines have no idea. They don't know enough about the world and they don't know what structures and events make sense and typically happen in the world.

So I will show you a blurry video. And I wonder if, well, some of you who didn't see it, whether you could figure out what's going on.

Who have not seen this video? Could you tell me what you saw?

AUDIENCE: A person was talking on the phone then switched to working on his computer

BORIS KATZ: Right. Well, this is amazing. Even with almost no pixels there and you still recognize what's going on, well, because you know what typically happens in the world. Well, of course it was sort of a joke and people sometimes make mistakes, too. So here's the unblurred video.

[AUDIENCE LAUGHTER]

Well, but, all jokes aside.

AUDIENCE: [INAUDIBLE]

[AUDIENCE LAUGHTER]

BORIS KATZ: Jokes aside, though, it would have been extraordinary if our machines could make mistakes like this. Not even close.

Well, so we may want to ask ourselves questions. How is this knowledge that you seem to have is obtained? And how can we pass this knowledge to our computers? How can we determine whether this computer knowledge is correct?

Our partial answer is using language. And I bolded the word partial here, because clearly there are many other ways humans obtain knowledge about the world, but today I will be talking about language and show you what is needed to give knowledge to the machine.

So we have a proposal. We would like to create a knowledge base that contains descriptions of objects, their properties, the relations between them as they're typically found in the world, and we want to make this knowledge base available to a scene-recognition system. And to test the performance of the system, we will ask natural language questions.

One of us here has decided to see what questions people actually ask, and he set up an Amazon Turk experiment, where he showed people hundreds of images and asked them to write down, generate questions about these images. And I will show you a couple.

Here's a scene. In here, the questions that people ask, you know, how many men in the picture? What's in the cart? What does the number on the sign say? Is there any luggage? What is the color of shirt of some lady?

Another example. Who is winning, yellow or red? Well, the answer is, of course, red, but how did you do that? Well, you need to know that this is a sporting event, which all of you do. That it involves, in this particular sporting event, winners and losers, that you do. It needs to know that this sort of short-hand yellow and the red means people wearing these colors, rather than color themselves. You need to know to pay no attention to people wearing red or maybe blue

in the audience. And you also need to know that a participant on the floor is likely a loser, not a winner. That's a lot of knowledge.

So, back to our proposal. We want to try to give at least some of that knowledge using language, and for that, of course, we need tools. And over the years, we've built a system called START, which, in fact, contains some tools that could be helpful for this task. And I will be happy to share the API with you so that you could use the system and maybe try to see what to do with the knowledge that you give it.

So there are only three tools on this slide. So one is going from language to structure. So we know that to provide machines with knowledge, we give the machine a bunch of sentences, texts, paragraphs, and that will be converted into some kind of semantic representation. And I will show some details of that.

We want to go the other direction. We want a machine to explain what it does or describe its knowledge using language. So we have a generator that does that, that goes from semantic representation to language.

And we want to test the machine, because it's very important to know whether what you thought the machine, if actually it understood you correctly. We want to ask questions, give queries. Those will be converted in semantic representations that will be matched against what the machine knows. And the computer will either give you a language response or perform some actions, which will indicate that it understood what you asked.

So we will go through these tools and I will describe you the START system in detail, but I just want you to remember that this is a discipline and engineering enterprise and these are the tools that I want to give you and other people so that you could start thinking deeper about human abilities and modalities, like vision and language and others.

Some of the building blocks of the START system. We need to parse language, we need to come up with semantic representation, generate match, reply, and so forth. So let's very quickly go through that.

Most of you, somewhere in middle school, learned about parse trees. Linguists love them. They're beautiful. This is an example of a sentence from Tom Sawyer. Tom greeted his aunt who was sitting by an open window in a pleasant rearward apartment. And linguists like to argue about exactly the representations, but pretty much most of them will agree that

something like that represents the structure, the syntactic structure of the sentence.

Well, they're beautiful and nice, but they're really horrible if you want to store them, if you want to match them, if you want to retrieve from them. And so we use the information found in parse trees, but we developed a different representation which we call Ternary expression representation, which is a more semantic representation of language.

It is syntax-driven, but it highlights semantic relations which humans find important, and it also, because we give this knowledge to computers, we made it very efficient for indexing, for matching, and for retrieval. It's also reversible, and I'll explain in a second what I mean by that. We implemented it as a nested subject, relation, object set of tuples.

So here is a different sentence. Say you have an image. You may recognize one of the characters there in the back. It's Andrei Barbu. And say you want to describe, in language, what you see here. And you say something like, the person who picked up the yellow lemon placed it in the bowl on the table.

Using this structure, subject, relation, object structure, you could, and first parsing the sentence, you could create this Ternary expression. And you could see, person picked up lemon. That same person placed that lemon in the bowl on the table, and that lemon happens to be yellow.

To make it a little bit more easy for you to see what's going on and convenient for humans and machines, we created a sort of topologically equivalent linearization of that graph, that knowledge graph, and as a set of triples.

They're a little bit misleading here just due to its simplicity, but all words here, of course, need to have an index, because if you have, say, a tall person and a short person or, then, you will have to distinguish between them. So you need indices, but for simplicity, I didn't show them here. And then the verbs also have indices so that when you use the same word place here, that is the relation, the triple, that happened to be in the bowl, that the person placed a lemon in the bowl.

So this is all representation and when we distinguished at least three types of Ternary expressions. So the first type you see here, syntactic structure of a sentence.

We also have syntactic features. The fact that the sitting was, it was past tense. She was sitting, it said here. But it's a progressive tense, was sitting, and also what kind of article things

have. The window had an article and an indefinite article. And also the lexical features that don't change from sentence to sentence, the fact that Tom is a proper noun and so forth.

Well, I told you that our representation is reversible. Well, we need to be able to teach machines to talk to us. And there are many reasons to do that. Some of them are shown on this slide.

We want your robot or your computer to explain what it does somewhat remotely. You want your machine or your robot to answer questions which are complex and the robot may want to ask you about it for clarification. You want to keep track of conversation history and state. Engage in mixed-initiative dialogue. Offer related information.

So all these things need to happen in the dialogue and, therefore, your computer must be able to speak to you in a language that you understand. In fact, I find that the biggest problem with learning systems that we have today is that some of the work can work quite robustly, and sometimes give you good results, but you have no idea why.

You press a button, you say, aha, here's the number, and put it in the paper, but more recently, people started looking at why it does what it does. But, again, it's done by numbers. It would be really wonderful if our learning system could tell us why they came up with their conclusions.

So we need language and so we created, sort of built a START generator that goes from those same expressions and create natural language. This is why we call this representation reversable.

So given a set of Ternary expressions, the machine will create a sentence, the person who picked out the yellow lemon placed it in the bowl on the table. But, of course, this is a little bit silly, just parroting the same sentence back. You want the machine, for example, as I said, to ask you a question or indicate a negative statement or rephrase it from different pieces that it knows about.

So, in fact, our generator is very flexible. So here is an example where by, say, observing the world, robot adds more information to this representation, and they're indicated here in blue. And now from the original sentence about, which was the person who picked out the yellow lemon, placed it in the bowl, the machine, by just adding a couple of new relations, the generator will be able to ask a question, or the human. Will the person who placed the yellow

lemon in the bowl on the table pick it up soon? And so forth.

All right, so we talked about parse trees, about semantic representation, about generation. So what do we do with that?

Let's-- Supposing that you gave some knowledge to your machine, hear that Tom Sawyer assertion, and somebody asked you, was anyone sitting by an open window? Well, what needs to happen is this question needs to be converted into Ternary representation as I indicated, and this knowledge base, we assume, had the knowledge from the original assertion plus million other assertions, of course. So we would need to match the representation of the query with the knowledge base, and the machine will say, aha, here is the match.

Well, this is very simple here, if you think window needs to open to window, open to match window open, open, and sit and sit, and then the word anyone, which sort of matching the word, needs to match aunt. But, in reality, of course, people ask questions which do not that closely follow what the machine knows, and so for our matcher, it needs to be much more sophisticated. This is just the graphical effect, sort of knowledge, graph representation of that match.

So START distinguishes things like term matching, which as I showed, could be a lexical match. Of course, it knows synonym. It knows hyponymy when it goes one way. It's like a car is a vehicle. And as you can imagine, the match also needs to go one way. If I say I bought a car, it also means that I bought a vehicle. But if I say I bought a vehicle, it's not true that I bought a car because I may have bought a truck, but this aside.

So it's pretty easy to do matching on the level of terms of words, but a much more complex problem is to match on the level of the structure. And I will show you some examples of this problem. But by now, you must have figured out I love to stare at English sentences. I hope you do a little bit. If not, please try.

So here, let's consider a couple of verbs. So here is the verb surprise. Let's consider these two sentences. The patient surprised the doctor with his fast recovery. The patient's fast recovery surprised the doctor.

Now for you who are used to understanding language so quickly, it's even hard to hear what's different about the sentences. But if you actually do the parsing, you will see that the parse

trees are dramatically different, and therefore, our Ternary representations will be very different. So we need to find a way to tell the machine, yeah, that's the same thing. And the linguists call these things syntactic alternations as well.

In a different verb like load, here's a different alternation. The crane loaded the ship with containers or the crane loaded containers onto the ship. Again, means the same thing pretty much, but the surface representation is different.

The next one is in terms of a question. Did Iran provide Syria with weapons or did Iran provide weapons to Syria? Let's see if it's true for every verb in the universe. So let's try to look at the, say, surprise alternation and use it with a load verb or the other way around. So I hope you're bearing with me.

So linguists put stars in front of bad sentences. And so here I tried to use the word surprise without alternation, which load allows you to do. Here, you do the same onto, and it says the patient surprised fast recovery onto the doctor. It makes absolutely no sense.

Here I tried to do surprise alternation for the word load. And it says the crane's containers loaded the ship. Again, complete gibberish. And the same below. Did Iran's weapons provide Syria?

So it looks like a really horrible story. Every English verb, it looks like, has a different way of saying, expressing these alternations, but fortunately, this is not the case. Let's go back to the verb surprise.

Well, let's look at verbs similar to the word surprise. So you could use the same alternation with the word confused. You can say the patient confused the doctor with slow recovery, which will convert into the patient's slow recovery confused the doctor. You can say the same thing with anger, disappoint, embarrass, fight, and impress, please, threaten.

And what is really amazing about it and very interesting that this syntactic alternation works the same way for verbs of the same meaning, of the same semantic class. And this particular class is called the emotional reaction verbs. And it's a large semantic class of about 300 verbs, and they all behave identically from the point of view of that alternations. And it's true for all other alternations that I showed you.

So that's, of course, is good news because it makes an interesting connection between syntax and semantics, but it also allows to build lexicons that are more compact and easier to deal

with. And one can imagine creating this verb membership automatically by looking at large corpus. And this is how, presumably, children learn these verb classes and these alternations.

All right, so now that you know how to match and it's not just trivial match only, like I showed here, but a more sophisticated match on the level of structure as well, let's see what we can do after the match happens. So here's the same sentence and the same question. Was anybody sitting by an open window? We retrieved the structure and then we could tell our generator, go and generate the sentence, and it will do that. Tom's aunt was sitting by an open window in a pleasant rearward apartment.

Well, it's not as interesting. It's sort of parroting the same thing I tell it, ABC, and it told me back, ABC, if I ask who BC or something like that.

If you want to build a question-answering system, we want it to be able to, in response to a question, understand it, go somewhere, find the right answer, and give it back to you. And we build that and we do it by a general, where, if looking at it, our system can execute a procedure in response to a match to obtain the answer from the data source.

So an example is here and I can show you some screenshots or, in fact, if you like it, we can play with the system live and you'll see what it does. So it executes a procedure to obtain an answer from the data source. If you say who directed *Gone With the Wind,* from match, it will happen between what you know, what you ask, and what the system knows, some script will get executed and the machine will go to some data source, find the answer, and give it back to you.

So how this is done? Well, in order to explain that, I need two more ideas. One is a natural language annotation idea. So annotations, sentences, and phrases that describe the content of retrievable information segments, this is a graphic sort of, in a cute way, show this sentence level, or phrase level labels on some data, and they describe the retrievable information segments. Annotations are then matched against submitted queries and the successful match results in, either retrieval of that information or some procedure, to retrieve that information.

And the special case of this procedure is done using our object-property-value data model. This technique can connect language to arbitrary procedures, but as I said, let's consider a lot of semi-structured information sources available on the web that can be modeled using this object-property-value model.

Well, what are these, what kind of repositories have this property? If you think about it, almost anything that humans create on the web, which is semi-structured, is like that. If you have a site that has a bunch of countries with properties like populations, areas capitals, birthrates, and so forth, the country is an object, the word population is a property, and value is the actual value of that property. You can have people with their birth-dates, you can have cities with maps and elevations and so forth.

So in a sense, this object-property-value model makes it possible to view and use large segments of the web as a database. And schematically, here's how START uses this model. A user asks the language part of the system a question, the system needs to understand the question, understands where the question might be found, and what is the object and property implicit in the question.

After START does that, it says it has a friend called Omnibase, and it says, go get it. Go to this site. Go for that symbol called France and go get the population. And it will go to some world fact book and get the population.

So this is how the system works and here's an example of such a question. Here, the question is, it's a screenshot, does Russia border on Moldova? The system says, ah ha, you want to find out what countries border Moldova, and find out whether Russia is among them. And then it actually checks that and it tells you, no, Russia does not border Moldova, because it doesn't find Russia in this response.

And just for comparison, if you ask the same question of a search engine, it will give you 24 million results, today maybe 240 million results, and none of them really answer the question.

Well, I just want to tell you that the ability to understand something really helps. In this case, the ability to understand language gives you a lot of power. You can do a lot by trying through the keywords and you can retrieve a lot of documents, and this is how all of, pretty much, modern systems work. But if you want to do something a little bit more complex, it would be nice to understand some.

So here's an example of a complex question. Who is the president of the fourth largest country married to? Well, if you can analyze this question into pieces, then you can very quickly figure out that, right away, just throwing the pieces on your knowledge base, you cannot resolve it. But we've built a very nice syntatical-based algorithm that allows us to resolve the complex questions into subsets of simpler questions and understand in which order to ask them.

So the machine will say, oh, first I need to find out what's the fourth largest country, then who its president is, and then with that, who he is married to. And, very quickly, this is how, schematically, it's done. This is sort of an under the hood Ternary expression representation of the question.

The machine says, oh, too hard, let's first find out what the fourth largest country is. It's China. Then let's find out, it's still hard, so let's find out who the president of China is, found the name, and then the next is just a lookup table. Who is he married to? And it gives you an answer.

Some other examples. In what city was the fifth president of the US born? And finds like a James Monroe and gives you the city.

What books did the author of *War and Peace* write? Finds Leo Tolstoy and finds his books from different sources.

So the technology that I described, object-property-value data model, our Ternary expression representation, complex question answering, the annotation, natural language annotation, representation. They, over the years, inspired a bunch of companies and a bunch of technologies, starting with Ask Jeeves, who, I guess, existed before you guys were even born, to Wolfram Alpha, who pretty much took [INAUDIBLE] wholesale, to more recently, Google QA started doing really wonderful things using this idea that, everybody had the idea that you should go from surface to a question.

If you have a question, you throw your question onto the web and you get some answer, but it doesn't work with high precision. So the idea that you need to curate knowledge and build some huge depository of knowledge was picked up by these companies to, certainly now, all these companies do quite decent question answering. And the same is true for Watson and Siri, and I was involved in some of these things, so I will show you.

Let's see. Right, so let's start with this. About 10 years ago, we, on top of START, we built a system that was connected to a cell phone. I don't know how many of you remember the world without smartphones, but that was when smartphones wasn't there. There was no such thing as iPhone. So there's a vanilla phone that, all it did, it made phone calls. Of course, it also had a camera and unlimited text. But it really didn't do much more and we decided it's time to connect it to language.

So we convinced the company to fund us to do that, and we build some language called StartMobile. And this is an intelligent phone assistant, which could, at the time, retrieve general purpose information, provide access to computational services, perform action on another phone, trigger apparatus, like a camera on a phone, receive instructions.

And we have a, talking about YouTube, we have a video that shows the system in action. That video is quite old. It's from beginning of 2006, and at the time, we did not connect to speech, but you could see what it does by, the user was typing in questions for the system in that particular video.

So there's no narration so if you read the captions, you'll figure out what is going on.

So here's my former student, who actually went to Google to transition our technology eventually. And he's not sure whether he needs to take his coat or not.

[JAZZ MUSIC]

Again, this is very dated, of course, because now the temperature is almost uniform, but, again, that was 10 years ago.

Is there any sound?

AUDIENCE: Yes.

BORIS KATZ: All right, those of you that know Cambridge know that station.

Where am I? The GPS just came about and we were lucky to connect it and so now the guy gets the map and knows where to find where he needs to go.

So this is our data center for those who haven't seen it. This is where CSAIL is. Again, it's dated, because right now this lawn is a huge building, but it wasn't there at the time.

Oh it says here, trying to reach my mother. I don't know why it shows you this stuff, but.

AUDIENCE: [INAUDIBLE]

BORIS KATZ: He's worried about his mother and so he decides to tell her, remind my mother to take her medicine at 3:00 p.m. And we'll see what happens with that later.

Take a higher resolution picture using flash in 10 seconds. I don't think any phones can do it

even today for some reason. I don't know why it's so hard.

[AUDIENCE LAUGHS]

AUDIENCE: For a selfie.

BORIS KATZ: Right, for a selfie. Very good, yeah.

All right, so his friend is busy and he wants to entertain himself, I guess, but he doesn't quite know how to do it.

How do I use the radio on my phone?

Well.

All right, now he knows.

[MUSIC PLAYING]

All right, mother's health.

[AUDIENCE LAUGHS]

Right, so, exactly. So a delayed accident happened on her phone so we inserted the thing on her phone and then she got this warning, and that's my stuff. They all turned out to be very good actors.

All right, so this is the last thing. Traveling, she is going back. She now has a car. And this is the last thing that I'll show you.

[JAZZ MUSIC]

How do I get from here to Frederica's house? Well, if you think about it, this is a very hard question. You need to know that here is here and go to GPS and find the location. You need to know Frederica's house from your list of contacts that need to go there, and you need to then send it to, in that case, we send it to, I believe it was MapQuest, I don't even know if it exists now, to actually give the directions.

Well, anyway, so.

Well, it was a little bit of a sad story actually. So we built the system. The company that I mentioned was Nokia. We showed them the demo. They were very excited about it. They said, well, can we put START on the phone? Because in that application, the signals were sent to MIT from the phone and the answers were sending back to the phone.

I said, well, it doesn't seem right. START is large and there was no internet that could take care of that.

They said, no, no, no, how big is your system? Can you talk through the company to put it to a LISP compiler, to put it on our chip, and so forth. We need another phone.

Unfortunately, the word cloud hadn't been invented yet. Maybe I would have been more eloquent in explaining to them why they don't need to have the system on the phone. And so they didn't want to use that the way it is.

We wrote a paper, showed it to them, say, do something about this, it will be too late. And right at that time, Apple released its first iPhone. So I go to like senior vice president and say, look, these guys are ahead of you. You should decide about it because they will do what I gave you.

He asking, how many iPhones did Apple sell last month? I said I read somewhere like a couple of thousand. And he starts laughing hysterically. He said, we, my company, ships one million phones every day. Why do we care about Apple, he said.

Well, so, we gave this talk. That was September 2007 by then. In December, somebody started a company called Siri, and then two years later, Siri was bought by Apple and the rest is history. And Nokia was sold pretty much on a yard sale and doesn't exist anymore.

So be visionary. Don't think that you know what you are doing all the time.

Yeah, people often ask me to say a few words about *Jeopardy!.* The question that the IBM team was hoping to answer was actually a very important question. Can we create a computer system to compete against the best human in a task, which is normally thought to require high level of human intelligence?

I was involved with them from the very beginning for various reasons which I will not go into. They put together a wonderful team, some really good people, very devoted people. They spent four or five years of their life, pretty much, totally devoted to that. And they built a system, and these are the kind of, I guess--

I don't know if any of you know what *Jeopardy!* is but pretty much, people ask the question, which for various reasons, is formulated not as a question but as an assertion, with the most to reflect these that are pronouns and you need to give an answer, which they call the question, for some reason. There's a gimmick. You have to say what is envelope instead of envelope, but let's not pay attention to that. It doesn't matter.

And this is very hard. To push one of these paper products is to stretch the established limits, and you need to figure out that to push the envelopes means stretch established limits. This is a idiom for those of you who are not native. And the answer is envelope.

A simple question is the chapels of these colleges were designed by this architect? And you need to figure out that Christopher Wren is the answer.

Of course, many questions involve a question decomposition. So here's an example of a real question. Of the four countries in the world that the US does not have diplomatic relations with, the one that's farthest north? So it's pretty much asking several questions.

One is the sort of inner sub-question. The four countries in the world that the US doesn't have relations with, and the outer sub-question is, now that you know the answer of these four countries, which is the farthest north? You do a little bit of arithmetics and you find the answer is North Korea. And of course, this is very similar to what started years before, is pretty much you decompose the question and you solve them separately, as I showed you a few minutes ago.

So Watson actually took a bunch of ideas from START, the Ternary expression representation, the natural language annotations idea, the object-property-value data model, and the question decomposition model, and applied them when they could really analyze syntactically the question where the question was not too convoluted, and when there was a semi-structured resource, or several resources, to find an answer.

But many questions, of course, were not like that, and stretching the envelope is one example. And so Watson used some statistical machine learning approaches and they did quite a good job of looking at a lot of data to resolve and answer these questions.

Their pipeline is, really, miles long, because each of these bullets has three, which are a bunch of bullets with a bunch of bullets that the task that they were doing, but on a very high level, they needed to content acquisition. Pretty much, all right, there was a problem. The

company, *Jeopardy!,* told them the web cannot be part of that because everything that Google knows everything. So what is it that you guys are doing?

So they had, so what would you do if somebody tells you you cannot use the word?

AUDIENCE: [INAUDIBLE]

BORIS KATZ: What's that?

AUDIENCE: [INAUDIBLE]

BORIS KATZ: Well, you pretty much bring the web and put it in a box. And this is what IBM did. They took every repository, interesting repository, every database, every encyclopedia, every newspaper collection, I forgot where the blogs existed at the time, and just had a lot of clusters and a lot of memory and everything was there. So now they could tell the company no web. We are smart without the web. So that was the first thing.

Then, they actually, there were some wonderful natural language processing people do, so they did question answering, they searched the documents. So what they really did, they took the clue, they call it, the question, throw it on not the web but on their web, find tens of thousands of documents that even closely match these keywords, and then the real work just started.

They had this kind of filtering, that kind of filtering, this kind of answer generation, that kind of. They would score it, they will pay new evidence, they will do it again, they will do ranking, and they will decide how confident they are about that and then they will decide whether it's worth how much they spent, incredible amount of time figuring out how much money. I don't actually know much about *Jeopardy!,* but apparently you have to tell them how good you answered. Well, you need to come up with a number of how expensive you are, how much you will make if you win, and how much you lose if you lose. And so they did it all and they built a wonderful system.

In the beginning, when I started going there, that was very, very slow. It ran on a single processor and really took two hours to do this pipeline. But that, if you think about it, it's a very plausible problem. You could send it all to, in that case, I think by the end it was like several thousand cores, and they easily reduced the time to three seconds, which was passable and doable for the competition.

And so they won, as you all know. It's a great system that nails the state of the art in natural language, in QA, in information retrieval, machine learning. It's a great piece of engineering. It reignited, no doubts about it, public interest in AI. It brought new talented people in our field.

So this is all great news. But let's look at some of the sort of blunders that occurred, well, both before the competition and after. I had a whole bunch, a collection of those. I'll just show you a couple. This actually happened before the competition so they figured the problem.

So the question was, again, it's called a clue, in a category of letters, in the late 40s, a mother wrote to this artist that his picture, number nine, looked like her son's finger paintings. Well, for those who are quick at that, I'm sure you know that it's Jackson Pollock, but Watson sent Rembrandt for some stupid reasons. It failed to recognize that late 40s referred to 1940s, or rather they thought it's made in previous centuries, and apparently, it has something to do with number nine. That was in a bunch of documents related to Rembrandt and so it said Rembrandt.

Another more famous blunder, because that happened at the competition, the category was US city, and the question was its, which is a series, largest airport is named for a World War II hero, and its second largest for a World War II battle.

And again, those of you quick at that will know that in Chicago there is this O'Hare Airport and he happened to be famous here from the world. And the second airport called Midway, which is a famous battle in the Second World War. And Watson presses the button and says Toronto. And there's a sort of gasp in the audience, and also like tens of millions or hundreds of millions television sets around the world.

And again, there are some stupid reasons. Watson did machine learning, as I said, and it statistically figured out the category part of the clue, which in this case, since this is a US city, it might not be that important, so we should pay less attention to it. It also knew that Toronto team playing in baseball-- is that true? Yes. In the US baseball league, and that, in fact, one of Toronto's airport is named for a hero. Although, it's a World War I hero.

So it put it all together and said Toronto. In any case, it won anyway because it did an amazing job on answering many more questions, but the question to us is whether this is what we should all be striving for.

I'm certainly all in favor of building awesome systems, and they did, and I explained to you why

I think it's good, but IBM has not created a machine that thinks like us. And what's in success didn't bring us even an inch closer to understanding human intelligence. And the positive news, of course, is that there was those blunders, should remind us that the problem is waiting to be solved and you guys are in good positions to try to do that. And that should be our next big challenge.