Almond: An Open, Programmable Virtual Assistant
Giovanni Campagna

>> Hello? Is the session shared here?

>> Hello. I can hear you very well.

>> Oh, cool, cool. Should I go ahead put the slides up or should we wait a few more minutes?

>> Yes. You can go ahead and do that while we wait for a few more minutes.

>> We'll start at 9:45 or whatever it is.

>> Yes.

>> Everybody can see the presentation?

>> Yes, Giovanni, I can see it. We can see it.

>> Okay. Perfect.

>> Okay. Let's wait five more minutes. You can also share your video if you have a webcam.

>> Yeah, I should probably do that. Yes. Let's wait for a few more minutes.

>> Quick reminder to everyone in the room that after this talk, we'll have a break and then the keynote will happen on track one. Not this one, the other one.

>> All right. Need talk is Almond: Crowdsourcing an open, privacy-preserving virtual assistant by Giovanni Campagna.

>> All right. Good afternoon, everyone. My name is Giovanni Campagna and today I'm going to talk to you some of the work we're doing. We're building an open, privacy-preserving virtual assistant. This is my first time giving a full talk, so I thank you for giving me the chance to present.

We're interested in this virtual assistant because we're seeing a great opportunity called the linguistic web. The idea for this uniform language-based interface for the post-PCera. We see the personal info and intermediate all digital services and help us choose among competing offers. Virtual assistants also presents risk. In the United States there's a proprietary linguistic web in the making. Alexa has 70 percent of the 76M installed base of owners in the U.S. There's 60,000 compatible IoT devices and it presents a real privacy risk.

So why is this happening? Well, the thing is that the virtual assistants unlike other applications is a very high barrier to entry. You can make it unusably brittle. However, if you use the high ones, they can provide annotated dialogues, but it can be very very expensive, and it's also

used for privacy. So what is serving is the largest companies that can afford to build virtual assistants. Unless we change the technology. So for the rest of my talk I'm going to talk about a new, cost-effective way to build virtual assistants. This is based on formal languages that defines the capabilities precisely. We're going to present a way to use data engineering to synthesize large data sets instead of annotating by hand.

I'm also going to present autonomy, privacy and convenience. This supports a new class of commands called compound commands. I will talk about how the virtual assistants can exist on the desktop and finally I will talk about where we are and where we're going with our project.

So the first part is how do we lower the barrier of building this in AI technology. Well, we are engineers, and as engineers we see the value to entry by having five point of entries. First of all, it's cost. Many dialogue that are used. There's a life cycle, cold start, iterative refinement. There's usability, coverage, accuracy. And there is effectiveness in terms of how precise and how controllable this system is. If it is entirely statistical and a certain famous Twitter bot that didn't go very well. There's scalability, there's 1.8 billion web pages out there. So we need a better engineering methodology that is focused on this metrics. What we call the CLUES methodology is to build tools that let the world build the assistant as well as the environment for data. Our approach is granted on a semantic parser. This takes natural language input and it translates it’s to a high-level virtual language. And we propose to have a common virtual assistant programming called ThingTalk. We hope that this is built around this language so we can share the tool.

So how do we use semantic parsing for virtual assistant? Let's say the user is trying to look for an Italian restaurant. The semantic parsing will look up in particular for those where the food is Italian. So this ThingTalk runtime will look for this in the general dialogue in Palo Alto and it will support all the skills and all the services. This will take the results and will produce the response that we can show to the user. Both will have a high level, and this is taking the action to propose one result. Then it will go through. Note in particular that the high level is passed back, and this is where a semantic difference from this system because we don't need to model this one that isn't coded.

And so how do we use a neural metric? So a neural metric is essentially a magic black box. In principle, it can do anything. In fact, it can solve many problems. If you have pairs of input and output examples. In particular we are leveraging the BERT network from Google. That is a neural network that is trained to learn natural language just by scraping all the web and understanding how this fits together. In semantic parsing the data is utterance plus the state and plus the program. The agent took and said in the previous sentence and the program. What we expect the neural network to produce which is the next query we want to execute.

And so we said that neural networks can learn from data and can learn anything provided we have the data. But how do we collect this data without annotating. So we propose that once you provide the higher-level specification, we provide the schema and indicating how that column is in that language. We build this tool that's called Genie. Genie comes with a general machine that covers all the dialogues. All the dialogues where the user is trying to get to do something. And what Genie will do is dialogue all the possible things that will talk about restaurants and it's talking about the users looking for information about food place and it's looking to book it. Now the agent responds with the proposal and the user can then follow up and so on.

So how do we automatically convert the high-level specification of the database into natural language dialogues that we can use to train the semantic parser? Well, for example, the database operational of selection of filtering maps through specifying a table that has a parameter and it can be entered in a number of the table. So we can write a set of templates that expresses how that database and that operation is reflected. Of course, natural language doesn't fit the templates. If it did, then we can use the temperature Latinos to understand the user rather than going through this convoluted process.

Our approach is to capture as much of the variety that we can in its domain and dependent manner. For example, we leverage the intrinsic grammar of English so we can capture the different purposes of the sentence. So not only search for restaurants but what kind of restaurants. These are all the different types of sentences. We can also leverage and set a standard type such as people, objects, time, and location. And the ones that map in English. And finally we have this that covers weight, height, age, and length. Then we can apply them to all database that uses those types.

It is not enough though to capture the variety in a domain and dependent way. We also need to capture the specificity of each database. And as I said, we do so by having annotations on every field. Every field can be used in different ways in a sentence that correspond to each different part of speech. So, for example, if we look at the alumniOf, we can see who has a Stanford degree or we can say who was educated at Stanford. Also know that for sound fields we have access to all the different parts of speech. They're all meaningful. For some, like you can see, we see who has Italian food, but this is such a thing where is an Italian restaurant.

And so having specified database, how do we go and having the ability, how do we go and make dialogues and not just search database? Well, we observed that there is a high-level structure that applies to all dialogues where the user is trying to perform some action on the virtual assistant. So let's take this example. This is taken from the Wizard of Oz. The user is looking from the restaurant and first they're going to search and as the user finds the results in different ways (audio breaks). As we capture this general structure shared by all transaction dialogues in an abstract state machine. And this state machine has dialogue acts such a search request and proposing one result or multiple results from an agent in green. All in particular that there is this pass from the instate. We can already see that we can generate many thousands of possible dialogues that we can use to train. And we can also see if we were to understand all the possible dialogues from data that we collected then we would need millions of dialogues. We would need Amazon scale to understanding the state machine.

And so zooming in in the state machine it is dependent. So these templates are sentences of families with placeholders. So this is translated by the abstract machine. From there the agent can decide to propose one or propose multiple results. If the agent proposes multiple results then the user can ask a question or it can just say I don't like any of those, do you have something else. Or I can say I like that, can you help me, for example, book it.

And so this is what happens when you put it altogether giving us more result where in the case a user was looking for a restaurant in Palo Alto. I have Terun and coconuts. And the agent will say, okay, for what day would you like your reservation for? So to recap this part a new technological approach is a high-level specification of the domain of interest and the database schema and a few natural language of ten, 12, 15 around that number for that field and we can

combine that and generate many of these. Then we can use the train that understands not only what we have in the template that understanding in the state machine. It can actually be robust to things that we didn't think of because first of all we started with a neural semantic and so we don't have to cover all possible paths. That's why this approach gives us the best of both worlds. It is robust and generalizable that the system will use. That's why we believe that our approach is a unique opportunity not just to match the proprietary assistant but to surpass it. No other assistant at the moment before the proprietary system has the technology.

And so that was about how we make our technology or build the conversation of AI at a much cheaper cost. Now that we have the technology, can we use it to build something more power and feel more interesting and more useful for users? We start by observing that today's visual assistants are a lot like browsers. We can interact with one scale at a time. Humans don't want just browsers. They want to connect to different sites and different scales. And ideally, they would let the assistant decide in similar situations where the assistant will decide automatically.

So let's look at one example, Bob is one of the eight percent of America that has a disease, and no it's not the coronavirus. Bob has asthma. So he will ask the assistant to help. Bob's doctor would want to be alerted if Bob's peak flow is below 180L/min. So Bob can be alerted and then the virtual assistant can alert Bob.

We observe the language is compositional. Complex sentences and complex commands are built from basic primitives. The favorite command of a Ph.D. student is when my adviser is online on Slack set me to away. Automatically upload all my Instagram pictures to Twitter. What's the total size of the files in my Dropbox? And finally we have the security camera when I'm not home.

And so, again, as you probably guessed, the way we're going to implement this is by translating every utterance into formal language. We have a control contrast and you can see all the commands that are shown it can be presented. Making this a very powerful contrast. And the primitives are taken out by Thingpedia. This has all sorts of services and it currently supports to 129 skills and 272 functions. For every function we classify whether it's a query or it is an action. For everyone we have natural language information associated so we can use our Genie technology to understand all these commands.

And Thingpedia allows us to have the privacy-preserving architecture from Almond. So the system is now running on my own personal device and now the system has access to the personal account and executes the commands privately. And the architecture is sharing, sharing means accessing data that is not only owned by my virtual assistant but it's possibly with other people. Sharing is an important key of what we would want our system to support. So if my virtual assistant can server request on my behalf it can also do it for other people. So we have a federated architecture where it can share with each other. It is generatable because any service can be controlled. It is usable because it is backed by natural language. And finally it is private because there is no direct access to the service.

I don't have time to go into details for this part. But if you're interested, I recommend you look at our paper. And in the next part I'm going to talk to you about what you're most excited about given that this is a GNOME conference, and this is about the desktop. How can we use a virtual assistant on a desktop computer? And the way I see there are three major uses. The first one is control. We can replace long menus and opaque terminals. And we can control apps without

switching. We can also search and access all content and finally as I've shown we wish to support automation by replacing bash scripts with natural language scripts where users will understand the natural language. So we might be able to combine traditional apps.

Going in more detail. Here are some commands. You can lock your PC; you can take a screen shot and the interesting thing there are natural points in these commands. There are standard interfaces. This will allow different components to communicate. So we can use the interface for screen shot. One in an area where the control can be used for, we don't have yet which is Mprints to control music.

The next one is search. There are a lot of queries or like what would be my favorite command on search is just pull up the right presentation. I don't want to look where I pulled the right PDF for this. The presentation is happening to this one, but tracker provides the database to this one. And the schema can be translated to an English database and to semantically query the tracker. We might be able to do the same and leverage the existing search provided interface because the information exposed by the shared provider interface is not semantic enough to be able to compute and talk about it. So this I think the virtual assistant is an opportunity to define a new of the search provided interface that is now semantic and accessible for natural language.

And finally automation. Automation is by doing away with repetitive task. I had a video here, but we have a prototype that does automation on a browser using a browser stems using Puppeteer. You can imagine using other OS primitives from the tool stack who control any application. This is the rough work. At the moment it's a prototype.

And finally to wrap up, where we are and where r we're going next in the Almond project. Well, where we are is, we are the open virtual assistant lab and it is paid for by your American tax dollars. We have a lot of students. Here in the slide I'm showing the actively Ph.D. students, but we have a lot more that have graduated and left.

Our current status is we're at one point something. I think we're at 1.8. We have a web version that you can go and try at your own risk. It is a research prototype. We have this where you can download it at Flathub. Home assistant is this open source and it's a very popular community. We're available at the home assistant Addon and integration. We're an entirely open source component. You can create a new scale in Thingpedia. Currently the depository API is free to use. We are very happy to collaborate.

As I said, we're at 1.0 and we're building to 2.0. It is a research project and we have to support the property types. And as we develop our technology and extend it, we increase the dollar capabilities. We are currently working and dialogue is more complex in comparison to symbols and virtual assistants. So it's going to be hard to make it across the skills. We're going to focus on the top ten skills. We also found that the Almond is a little tough. It's kind of hard to pronounce the Almond. So we're going to rebrand with an easier way for it. We will provide an intuitive speaker and maybe what I hear that it may be the first supported platform of the Almond assistant. And maybe we can have kind of --

>> A quick reminder, we have ten more minutes.

>> Right. I'm almost done. So our timeline is we started this road to Almond 2.0 in June 2020. We got one million dollar from this foundation and we're very happy. This month or in the next couple of weeks we're going to release a new development and a very stable version. We hope to have developer by then that actually works at the beginning of the summer with the goal to having a fully supported 30-party skill platform. We hope to be usable and ready for end users by the second quarter of next year.

Here I have highlighted from some of the community help. On some platforms like home assistants like the browser Alexa, Google, we can add the home voice. Voice is not an active area of research -- we're just using off the shelf technology and we're using proprietary API from Microsoft. Our project started before Deep Speech and before Mozilla. We would like to have help from the wide-open source community to support all the different languages. Here I have listed all the community resources and the talks and links will be available after the talk. And so to recap I believe we can beat, not just match, but beat the proprietary alternatives. Synthesis will order of magnitude cheaper and annotating data and end-user programming will provide new capabilities. And finally we care about privacy and an enthusiasts user community who cares about autonomy and privacy. And, of course, it's all free and open. Thank you.

>> Thank you very much. We have time for questions. So do you want me to read them for you?

>> How many questions do we have because if we have an extra minute, I can do a live demo.

>> Yes, we have two questions and seven minutes.

>> Let's see if I am brave enough. Okay. So we have an extension. I'm gowning to go into preference and enable voice input. So I can say, "computer, tell me a joke." Here comes the joke. "Computer, what time is it? " And that's what it does. That's a very short demo but you can see there's an integration in the shell that you can use in other desktop environments. Any questions?

>> Awesome. The first question is how much work is set to support for a different language?

>> So I didn't support for a different language. At the moment is a lot of work. This is hinted by our grammar inspired. So you need someone who has good knowledge of the machine and the language to build this machine, this templates. But we are working to fix that so it can automatically translate from English to other language. So at the moment the work to build to other languages is substantial. And that's one of the advantages to be closely related to the research site. That's an advantage that we get access to that technology.

>> Great. Next question is any plans for UX work on the client app? The current app?

>> Well, that's an area where we love community help. I have to admit that the GNOME client is something that exists because we love GNOME. It's not something that we would have built unless it would have been me. Assistance on the desktop like we've seen with Cortana they're not very popular. I'm not a UX person let alone a UI person. And that's why some help in improving the client application will help a lot, yes.

>> Yes. Contribution is welcome then. I see two more questions. How does this compare to Microft.ai?

>> Yes, they're based on classical technology. So every point they have to decide on the sentence. It belongs to one of these five or different intents to get good quality. And if you have a classical system like an SVM you might get away with fewer sentences. But the problem of Mycroft or Firefox existing methods they require a lot of work to get there. The short answer is they're not a state of the art.

>> Okay. And the last question is what is current local overhead of RAMs and network data?

>> So if you want to run the neural network locally it's about 500 megs in terms of big footprint. So it's quite a bit heavy. You don't need the GPU. You definitely need it to train it that's why we don't expect at the moment to run the neural network and all the parsing locally. We have to spin up a server, but it still presents privacy because unless you consent to recording then we don't record anything, and nothing is collected unless you consent to it.

>> Great. Thank you very much, Giovanni Campagna.

>> Yes. Thank you for inviting me and offering me the ability to speak. I'm very happy to be here.

>> Great talk. Next, we have a one-hour break and within this break there are office hours with our sponsor, GitLab, and after the break we have the keynote in track one. Then there's another talk in this track. So see you then.

[Lunch break].