ASSISTANT

Magnus Cardell
Andrew Lee

Fusing both our ideas here are the main components we would like to accomplish on our project:
1) Being able to activate the application from a suspended state using a keyword (much like "Hey Siri," or "Okay Google").

2) Converting the audio of human voice into a string of text, and passing that into Prolog

3) Using Prolog to parse though a limited natural language library, and to return result in the form of an action, or response audio.

We decided to scrap the idea of Amazon Alexa Specifically because looking through the API as well as their marketing model, it seems that Amazon Voice API is heavily geared towards prototyping Amazon Voice Platform applications rather than more specific applications such as parsing natural language, and passing those values into an engine separate from the Amazon Alexa Product Suite. For the same reason, we decided to forgo using a tool such as IBM Watson Suite.

We found the open source Intelligent Personal Assistant (IPA), Sirius (now renamed Lucida). Sirius depends on numerous open source programs such as CMU-Sphinx, Kaldi, Protobuf, and OpenCV.

However, because designing a IPA was our original intention, we decided to look into how the individual tools worked, and worked to create a way to do just the voice to text conversion at this level (since these tools do not utilize Prolog) and to be able to pass along the string of text into a Prolog interpreter and have Prolog do the rest.

We will now demonstrate the uses of the open source speech regognition software Carnegie Mellon University Sphinx suite. We specifically used a Python module that integrates Sphinx's voice recognition abilities with Python

Magnus
Starting pocketsphinx server waiting for connection
User sends .wav file to server, it is converted to text and send back to the user as text string.

Andrew
I was able to create a script that recognizes the spoken word "forward" and prints it out when it "hears" it.

Furthermore, I found a SWI-Prolog wrapper that allows us to run SWI-Prolog underneath a Python interpreter session, and to pass functors, arguments, data, and querries as you would normally do in Prolog.

While Magnus' code was more robust in the sense that it was able to parse a complex audio file, Andrew's code was able to analyze a real life human voice in real time.

Our next goal is to be able to have Python recognize a start point, and an end point to our query, then to pass that directly into Prolog without having to resort to clunky language wrappers. This will be cruicial in order to utilize the potential performance improvements of natural language processing that Prolog offers us.

Furthermore, we hope to be able to find a way to create and or utilize a natural language library to add a logical element to Prolog to be able to do something other than print out the spoken term data in terminal