New Skill 2: Regular Expressions

In today's video, we're going to continue working on our first skill, and I'm going to show you how to use regular expression files in your skill processing. Let's get started. **PYTHON REGEX**

Review and Overview

So in the last video, we tested our launcher skill. We had it set up to launch one of four programs, depending on what we said, and now we're going to look at that and think about what would happen as we started adding programs to this list. So right now, we just have our intent that requires a launch keyword and then optionally any of the other program keywords that we manually input. And then based on that, it's going to launch one of those programs. But if we add – as we add more programs, that can get a little bit cumbersome, so we're going to look at using regular expressions to filter out a program name from a message, without telling it what the options of that name are.

Creating a Regex File

So to do that we have our regex directory inside of our skill, and then our language underneath that, similar to our vocab and dialogue. And this one I've named program.rx, and what we're going to do with that is if we look at our launch keywords, I'm going to say one of those – any of those – so launch, lunch, or open, and then the word that follows that is our program. And this follows the Python conventions for regular expressions, I'll put a link to that in the description here. But what this means, it's going to look for one of these three words and then whatever comes next is going to be our program.

Adding to Skill init and Modifying Intents

So looking back at our init file, we'll see there are some changes. We've moved our options for each program into our init. So instead of the .voc files we had before, now we just have these lists of words that match for each program, and we've actually removed the .voc files that we are previously using. Our intent builder also looks a little different accordingly. So we still have required our launch keyword, and now we also have required a program, which it gets from our regex file we just looked at, and then we also have this optionally Neon, which we'll take a look at in a moment. But what this means now is it needs to see you say "launch something" or "Neon launch something" before it activates. So the word launch should no longer just call this skill.

Handling False Positives

So moving into our intent, we added this if statement, and if we look at this, we're saying: if we are skipping wake words and we have Neon in our message, then we'll continue. Or if we are not skipping wake words then we will continue. So you'll see this in a number of skills, it's useful to cut down on the number of false activations, especially a word like launch or lunch will quite probably come up in everyday conversation, and this way you won't have Neon interrupting and saying "I couldn't find that", every time it hears launch or lunch. So for this to work we've added our Neon.voc file here next to our launch keyword. If we

take a look at that it basically just contains neon and then some other common transcriptions, similar to how we have lunch in our launch one.

Changes to Our Function

So looking back at our function here, we've changed this around so program is now getting extracted from the message data, whereas before we were assigning it based on some other words in our message data. So now we've changed the logic to: we assume we have a match, we go through, and if we get one of the programs out of the list that we defined earlier, then we launch that program in the same way. So if Chrome then Chrome, else if Nautilus then Nautilus, terminal, Text Edit, etc. and then we've added this else, so if the program wasn't in any of these lists, then we say we did not find a match. If we did find a match, we speak dialogue the same way we did previously, and now if we don't find a match, we instead of using the speak statement we were before, we have another dialogue. If we look at that dialogue, file it's going to tell us "I could not find a program named", and then whatever program it heard. So, this is nice because you get a more specific response. Also, you'll find maybe a program that should have matched, kind of like how we had launch and lunch earlier, and it'll tell you what it heard, and you can say okay that should be added to one of these lists. So with that we're going to give this a test.

Testing our Changes

Okay so now let's give those changes a test. If I were to say launch Chrome, we shouldn't have anything happen because we're skipping Wake words, and I didn't say our keyword. If I say neon launch Chrome, (launching Chrome) then we get Chrome as we would expect. So I'm just gonna close that and go ahead and enable our wake words, so we can give that a try. So now if I say hey neon, launch Nautilus (launching Nautilus) we get our file explorer and even if I say hey Neon, Neon launch File Explorer (launching File Explorer) it'll still launch the file explorer. So, regardless of the Neon in front when we have wake words enabled, it will do- it will still parse out whatever is after our keyword. So, I'll close that out and then try something we don't have. Hey Neon, launch Neon (sorry I could not find neon) and that's exactly what we expected.

Summary and Closing

So that's an example of how you can use a regex file to simplify some of your intent building, and also how to get some more data out of the messages. So in the next video, we're going to look at instead of using Adapt, moving to Padatious and that will simplify things a little bit more here as well. Thanks for watching this Neon AI tutorial. Be sure to head over to neongecko.com for more information, including a written transcript of this tutorial and any code snippets we may have used. See you next time.