uSpeechRec: Julius Speech Recognition

Spring 2019

Michael Novitzky, novitzky@mit.edu
Department of Mechanical Engineering, CSAIL
MIT, Cambridge MA 02139

1 uSpeechRec: Julius Speech Recognition

The `uSpeechRec` application, now in version 3.0, is a module for using the Open-Source Large Vocabulary CSR Engine Julius. It is an interface between Julius and the middleware MOOS. By using `uSpeechRec` along with a dialogmanager, such as `uDialogManager`, a person can use a microphone and simple speech to interact with other MOOS applications.
Key parts needed for `uSpeechRec` are:

1. *Julius Configuration File*: A configuration file, such as Alpha.jconf, specifies important options for running Julius. Many modifications can be made through the configuration file. By default, the configuration file specifies the use of a microphone and is setup for the VoxForge configuration including their acoustic model.

2. *Acoustic Model Files*: An acoustic model represents the relationship between an audio signal and the phonemes that make up speech. Distributed with `uSpeechRec` is an English version of the acoustic model provided by VoxForge.net.
3. *Grammar*: A grammar specification in speech recognition is a set of word patterns and informs the speech recognition system what to expect a human to say. A sample grammar from VoxForge.net has been modified for the purposes of `uSpeechRec`. Explanations on how to modify and create new grammars are described below.

2 Building uSpeechRec

Prior to building `uSpeechRec` you must install some dependencies.

2.1 MAC OS X

We recommend using MacPorts to install dependencies on MAC OS X. You can add the packages for Julius and portaudio:

```
sudo port install julius portaudio
```

You MUST then add to either your .bash_profile, .bashrc, or .profile the corresponding library path:

```
 export LIBRARY_PATH=/opt/local/lib
```

It is necessary because unfortunately MacPorts is not properly adding the library path as of this writing.

2.2 Ubuntu

In order to install the Ubuntu dependencies simply enter into the command line:

```
 $ sudo apt-get install julius julius-dev libportaudio-dev libpulse-dev libasound2-dev
```

2.3 Build

To build on Linux and Apple platforms, execute the build script within this directory:

```
   $ ./build.sh
```

To build without using the supplied script, execute the following commands within this directory:

```
$ mkdir -p build
$ cd build
$ cmake ../
$ make
$ cd ..
```

3 Using uSpeechRec

Typical use of uSpeechRec has it situated in a community in which a human will interact with it using speech. In addition to uSpeechRec another application in the same community must interpret the sentences from uSpeechRec and deterimine how it should interact with the system.

3.1 Typical Module Topology

The typical module topology is shown in Figure 1 below. The uSpeechRec is situated in a community in which speech will be used as a form of interaction. It is typically run alongside uDialogManager and iSay for an interactive experience. The uSpeechRec application subscribes to the variables SPEECH_ACTIVE and SPEECH_PAUSE which perform the same function of muting the microphone input which mimics a paused state. The uSpeechRec publishes the most likely sentence to SPEECH_RECOGNITION_SENTENCE along with the confidence scores to SPEECH_RECOGNITION_SCORE. If any errors in speech recognition occur, uSpeechRec publishes to SPEECH_RECOGNITION_ERROR.

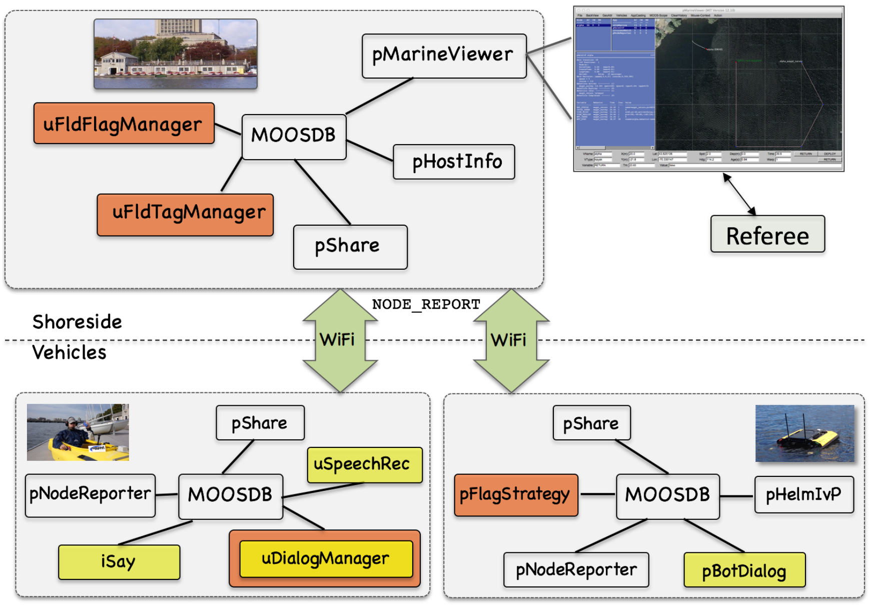

Figure 1: **Typical uSpeechRec Topology:** This module runs in any community in which one would like to use Speech Recognition. It is typically used with the applications uDialogManager and iSay for an interactive experience.

3.2 Pausing uSpeechRec

There may be situations in which you want to pause uSpeechRec. Pausing is accomplished by posting TRUE to the MOOSDB variable SPEECH_PAUSE. uSpeechRec will remain paused until FALSE is posted SPEECH_PAUSE. The status of SPEECH_PAUSE can be seen through AppCasting. Alternatively, by posting the opposite variables to SPEECH_ACTIVE will gain the same effect as pausing.

3.3 Starting uSpeechRec in a Paused State

There may be situations in which you want to have uSpeechRec start in a paused state. Starting uSpeechRec in a paused state is accomplished by inserting StartState = Paused in the .moos file for uSpeechRec. In order to have recognition start then a posting of FALSE needs to be posted to the MOOSDB variable SPEECH_PAUSE.

3.4 Interpreting the Outputs

The most likely output speech recognition sentence by Julius is published to the SPEECH_RECOGNITION_SENTENCE. However, just because it is the most likely sentence based on the acoustic model, vocabulary, and grammar files, does not mean that is confident that is the correct sentence. To get further confidence, starting with uSpeechRec version 3.0, it is best to look at the SPEECH_RECOGNITION_SCORE. Here is an example SPEECH_RECOGNITION_SCORE

sentence= BLUE THREE ATTACK, confidencescores=1:0.999692:0.912241:0.73853:1, score1= -12884.4

The sentence section is a repetition of the most likely recognized sentence. The confidencescores section includes the confidence for each word separated by ':'. The first and last numbers reflect the starting silence and ending silence - which is typically 1. The numbers in between are the confidence, (0.0-1.0], that Julius has that the word is the correct word. This output can be leveraged by uDialogManager to reject a sentence based on a word confidence threshold.

3.5 The SPEECH_RECOGNITION_SENTENCE Format

The available set of sentences that uSpeechRec will recognize is tied to the `.grammar` file and the `.voca` file. The `.grammar` file describes the structure of the available sentences. The `.voca` file describes the vocabulary that comprise the sentences.
Let's take a simple `.grammar` file:

```
S : NS_B SENT NS_E

SENT: NAME COMMAND
SENT: ACK
```

Here we see a simple grammar specification starting with the first line:

```
S : NS_B SENT NS_E
```

Where S is the root and it is defined with NS_B which corresponds to silence with a SENT or sentence and terminated with `NS_E` which is also silence. In the following line `SENT` is defined:

```
SENT: NAME COMMAND
```

A possible definition for `SENT` or sentence is created with the terms `NAME` followed by `COMMAND`. A second definition for `SENT` is created on the following line:

```
SENT: ACK
```

in which `SENT` or sentence is defined as `ACK`.

The words such as `NAME`, `COMMAND`, and `ACK` are defined in the `.voca` file. The `.voca` file uses the syntax % followed by grammar construct such as `NAME`. Below `NAME` are the text such as `ARNOLD` followed by the phonemes such as `aa r n ah l d`. See the following example:

```
% NS_B
<s>        sil

% NS_E
</s>        sil

% NAME
ARNOLD     aa r n ah l d
BETTY      b eh t iy
CHARLIE    ch aa r l iy
TEAM       t iy m

%COMMAND
RETURN     r ih t er n
DEPLOY     d ih p l oy
FOLLOW     f aa l ow
STATION    s t ey sh ah n

%ACK
YES        y eh s
NO         n ow
```

The configuration file for Julius has the `.jconf` extension. As an example we include the configuration block suggested by VoxForge.org:

```
# VoxForge configurations:
-input mic                            # live microphone
-dfa grammar/alpha.dfa
-v grammar/alpha.dict
-h acoustic_model_files/hmmdefs
-hlist acoustic_model_files/tiedlist
-spmodel "sp"                # HMM model name
-multipath
-gprune safe
-iwcd1 max
-iwsppenalty -70.0        # transition penalty for the appended sp models
-smpFreq 16000                # sampling rate (Hz)
-iwsp                        # append a skippable sp model at all word ends
-penalty1 5.0
-penalty2 20.0
-b2 200                # beam width on 2nd pass (#words)
-sb 200.0               # score beam envelope threshold
-n 1
# !!!!!!
```

The key parameters describe where the grammar files are found and the acoustic model can be found. In this example the two grammar file parameters are:

```
-dfa grammar/alpha.dfa
-v grammar/alpha.dict
```

The following parameters describe where the acoutic files are to be found:

```
-h acoustic_model_files/hmmdefs
-hlist acoustic_model_files/tiedlist
```

The rest of the parameters in the VoxForge.org `.jconf` file describe the model parameters which are beyond the scope of this document.

4 Adding New Words To Your Grammar

Because we are using the VoxForge.org dictionary, we are limited to the words and phones contained within. This was the dictionary used to train the acoustic model. However, if there are words within the dictionary you wish to use it is relatively a simple process to add them to the vocabulary list. Let's say you would like to add the word `STATUS` to your list of commands. That way you can ping a robot to get it's status by saying `ARNOLD STATUS`. First, we look into the grammar folder and search through the file `VoxForgeDict.txt` for the word `STATUS`. It seems that we have found two entries with the first being:

```
STATUS          [STATUS]        s t ae t ah s
```

In this example we are running `alpha.voca` as our vocabulary file. We will add just the english text `STATUS` and phones `s t ae t ah s` to the last entry under `%COMMAND`.

```
% COMMAND
RETURN     r ih t er n
DEPLOY     d ih p l oy
FOLLOW     f aa l ow
STATION    s t ey sh ah n
STATUS     s t ae t ah s
```

But we are not finished just yet. We must recompile the Julius grammar called `alpha` using the Perl script `mkdfa.pl`.

```
 $ mkdfa.pl alpha
```

If this worked properly, `mkdfa.pl` converted the `.grammar` and `.voca` file into Julius specific formats of `.dfa` and `.dict` files. Now, when we run the mission we can use the phrase `NAME COMMAND` with `ARNOLD STATUS` as an example.

5 Modifying Grammar

Let's say that we have exhausted the original sentence structure in the `alpha.grammar` file. For example, we really enjoyed using `ARNOLD FOLLOW` to mean that the robot `ARNOLD` should `FOLLOW` me. However, this is a limited vocabulary because we would like to specify an object to some of our commands. For example, I would like for `ARNOLD` to `FOLLOW` another robot such as `BETTY`. At the moment, our `alpha.grammar` does not allow for such a sentence. Let's add this powerful grammar to it! Our original `alpha.grammar` specifies two sentence formats.

```
S : NS_B SENT NS_E

SENT: NAME COMMAND
SENT: ACK
```

Let's add a third sentence format by inserting the following line:

```
 SENT: NAME COMMAND NAME
```

This sentence format allows us to specify any name in our `.voca` file followed by any `COMMAND` with the object `NAME`. Our final `.grammar` file will look like:

```
S : NS_B SENT NS_E

SENT: NAME COMMAND NAME
SENT: NAME COMMAND
SENT: ACK
```

We will need to convert this updated `.grammar` file into something Julius can use.

```
$ mkdfa.pl alpha
```

The Perl script `mkdfa.pl` has taken a `.grammar` file and a `.voca` file and converted them into Julius formats of `.dfa` and `.dict`. Now when running `uSpeechRec`, a user can use the additional sentence `NAME COMMAND NAME` such as `ARNOLD FOLLOW BETTY`. It is up to the user to determine how this new sentence will be used in a second app such as `uDialogManager`.

6 Configuration Parameters of uSpeechRec

The following parameter is defined for `uSpeechRec`. A more detailed description is provided in other parts of this section. Parameters having default values are indicated so.

Listing 6.1: Configuration Parameters for uSpeechRec.

`JuliusConf`: Names the file for the Julius configuration. For example `"JuliusConf = Alpha.jconf"`. See Section 3.5.

`StartState`: Decides the Paused state that `uSpeechRec` will start in. Options are 'Active' or 'Paused', see Section 3.3.

6.1 An Example MOOS Configuration Block

To see an example MOOS configuration block, enter the following from the command-line:

```
$ uSpeechRec --example or -e
```

This will show the output shown in Listing 2 below.

Listing 6.2: Example configuration of the uSpeechRec application.

```
===============================================================
uSpeechRec Example MOOS Configuration
===============================================================

ProcessConfig = uSpeechRec
{
  AppTick   = 4
  CommsTick = 4

  JuliusConf = Alpha.jconf
  //StartState default is 'Active'
  //Can be set to 'Paused'.  Must then be unpaused
  //by posting to the MOOSDB SPEECH_PAUSE=FALSE
  StartState = Paused
}
```

7 Publications and Subscriptions for uSpeechRec

The interface for uSpeechRec, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ uSpeechRec --interface or -i
```

7.1 Variables Published by uSpeechRec

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility.
- SPEECH_RECOGNITION_SENTENCE: The most likely sentence heard by the Julius Speech Recognition Engine. Section 3.5.
- SPEECH_RECOGNITION_SCORE: The most likely sentence heard by the Julius Speech Recognition Engine and the confidence scores in each word. Section 3.5.
- SPEECH_RECOGNITION_ERROR: If Julius Speech Recognition Engine does not produce a result then an error is output here..

7.2 Variables Subscribed for by uSpeechRec

The uSpeechRec application will subscribe for the following four MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- SPEECH_PAUSE: A value of TRUE will pause uSpeechRec while a value of FALSE will unpause/start uSpeechRec.
- SPEECH_ACTIVE: Used as an alternative to SPEECH_PAUSE. A value of TRUE will unpause uSpeechRec while a value of FALSE will pause uSpeechRec.

7.3 Command Line Usage of uSpeechRec

The uSpeechRec application is typically launched as a part of a batch of processes by pAntler, but may also be launched from the command line by the user. To see command-line options enter the following from the command-line:

```
$ uSpeechRec --help or -h
```

This will show the output shown in Listing 3 below.

Listing 7.3: Command line usage for uSpeechRec.

```
==========================================================
Usage: uSpeechRec file.moos [OPTIONS]
==========================================================

Options:
  --alias=<ProcessName>
      Launch uFldHazardMetric with the given process name.
  --example, -e
      Display example MOOS configuration block.
  --help, -h
      Display this help message.
  --interface, -i
      Display MOOS publications and subscriptions.
  --version,-v
      Display release version of uSpeechRec.
```

8 Terminal and AppCast Output

Listing 8.4: Example uSpeechRec console output.

```
 1  ===================================================================
 2  uSpeechRec mokai                                          0/0(682)
 3  ===================================================================
 4  ====================================================
 5  JuliusConf = Alpha.jconf
 6  Recognizer Paused
 7  ===================================================================
 8  Most Recent Events (2)
 9  ===================================================================
10   sentence: BLUE THREE ATTACK, confidencescores: 1 0.999692 0.912241 0.73853 1, score1: -12884.4
11   sentence: NO, confidencescores: 1 0.368944 1, score1: -10499.8
```

Line 2 includes the name or alias that uSpeechRec is using followed by the community in which it resides. The final set of numbers on line 2 include the number of appcasts that have been published. Line 5 includes the `.jconf` file being used by the Julius Speech Recognition Engine for configuration. Line 6 presents whether the speech recognition provided by Julius is 'Paused' or 'Active'. During normal operations Lines 10-11 provide a history of the latest sentences recognized by the Julius Open-Source Recognition Engine. Immediately following the most likely recognized sentence is the negative log-likelihood provided by Julius. If there is a configuration or runtime warning then Line 5 would be replaced with `.jconf` file issues or the possible situation when Julius could not be initialized.