**InMoov**

Deepti Bankapur

ECE 579: Intelligent Robotics II

Final Report

**Abstract:**

The purpose of the project this term was to develop a speech engine that has the ability to communicate and/or interact with a human. For this reason, by working on a speech recognition system we can add an interactive element to our robot InMoov who we named Artemis. Ultimately the goal is to make Artemis have a personality and interact with a human whether it be answering general questions regarding the school (Portland State), the weather or reading notifications from an email account or social media account. There are various possibilities.

This report talks about the process of building a speech recognition system and the issues encountered. The primary focus will be on pocketsphinx, which is the speech engine used. The CMUSphinx website provides the necessary packages and libraries required, such as the acoustic model. To create the language model, I have used the suggested online tool called Sphinx Knowledge Base Tool-Version 3 (<http://www.speech.cs.cmu.edu/tools/lmtool-new.html>). The system that I have built is run on Ubuntu 15.10.

**Introduction:**

Speech recognition is the ability of a software or program to receive and identify words or phrases from specific spoken languages and convert them to a machine readable format. The speech recognition software that I ended up using is called Pocketsphinx. Pocketsphinx is a speech recognition engine and uses the Pocketsphinx and Sphinxbase libraries to function. Their packages need to be installed and unpacked. It can be built on a unix-like environment, and I am using the latest version available which is Ubuntu 15.10. It also requires the following dependencies to be installed and they are: bison, swig, python development package, gcc, etc.

**Flow Chart:**

The following system shown on the flow chart consists of input received from the microphone which then gets decoded from the acoustic and language model and goes through the recognition engine to decipher the phrases in the library that is created.



Figure 1. Proposed structure of speech engine in terms of input data

**Hardware Used:**

The following are the devices used to build and run the speech recognition engine.



Figure 4. Intel NUC5i5RYK







Figure 4.1: Contents inside- 1 M.2 SATA 240GB, 2 Crucial Memory 4GB chips



# Figure 5: EBerry Home Studio Adjustable USB Desktop Microphone

**Working with Pocketsphinx:**

I successfully downloaded, installed and configured pocketsphinx on the Intel NUC. The Intel NUC is basically a portable PC. Since Pocketsphinx can run on an Ubuntu OS, I chose to download the latest version available which is Ubuntu 15.10. I am a fairly new user to Linux operating systems, so I learned how to use particular commands which I would have to use frequently. To test if the audio was being captured, I recorded a 5 second clip and played it back to check if I could hear the output via the audio jack. The NUC has a good sound card because the test file that was played back had minimal noise disturbance. However, there were certain issues I ran into while installing some packages of the software. The following was one of the errors that showed up on the terminal: “Error while loading shared libraries: libpocketsphinx.so.3”. To solve that, I had to export the correct path for the environment variables. Additionally, when trying to run a small test after generating the required files to decipher what is being said, I had to tweek the pocketsphinx\_continuous command to the following statement:

pocketsphinx\_continuous -hmm /usr/local/share/pocketsphinx/model/en-us/en- us -lm 6015.lm -

dict 6015.dic -inmic yes.

The pocketsphinx\_continous command is continuously running and basically captures sound from a microphone or file and converts it to text. Nonetheless, an error showed up saying that the main dict and lm files were missing from a specific folder. That folder did not exist. Therefore, I created the required folder and copied the files to the missing destination.

The dict file generated is the pronunciation dictionary input file and the lm file is the trigram language model input file. I created a list of about 40 to 50 words and put them in a text file so that I could use the lm tool to generate the required files for the speech engine. Moreover, reading specific phrases or concatenating words from the list created, the voice recognition software was able to detect and put together sentences correctly when there was no background noise. The software runs best when the environment is silent. If there is some echo or disturbance, it will display extra words that have not been said. While testing, I recorded a video of the software running to show that it works well in deciphering what had been said.

Meanwhile, since using a Linux based operating software was new to me, I learned various necessary commands that I would be using frequently. I learned how to add or remove files to or from a certain folder. The command cp (for copy) or rm (for remove) should be used respectively. It is necessary to include sudo to get root privileges. For the cp command, I had to include the path of the source file and then the path to the destination folder. Whereas, for the rm command, I just had to include the path to where the file was located. I had to use the rm command because I accidentally put one of the required files in the wrong folder. Below are examples of what I used to move and remove files.

sudo cp Desktop/9645.dic /usr/local/share/pocketsphinx/model/lm

sudo rm /usr/local/share/pocketsphinx/model/en-us/lm

Similarly, to edit files via the Ubuntu terminal, the sudo nano <filename> statement should be used. Again, we use sudo to get root privileges and nano as the file editor. If you type in a filename that does not exist, it will create one for you. File contents will be displayed if it recognizes the file and edits can be made. Exit and save changes made to the file for it to take effect. After testing the example library I created a larger library of phrases/words to test the accuracy of the word detection. I uploaded the file I created to the online tool at Carnegie Melon to generate the required files for the tool to decipher what is said. The following is the link used to generate the required files:

<http://www.speech.cs.cmu.edu/tools/lmtool-new.html>

After that, I tested out the pocketsphinx\_continuous command with the new library and the software was able to recognize what was being said. I also installed the Google Text to Speech (gTTS) to test if the software could recognize written text and read it out in a clear manner. The tool works great and the voice does not sound robotic at all. The gTTS tool creates an mp3 file with either the gTTSmodule or gtts-cli command. I used the gtts-cli command. For this I had to run it from the correct path where the gtts-cli.py file was located. The following are the two statements I could use:

python gtts-cli.py -t "Hello" -l 'en' -- hello.mp3

python gtts-cli.py -f /home/artemis/Desktop/greeting.txt -l 'en' -- hello.mp3

gtts-cli.py –help usage: gtts-cli.py [-h] (-t TEXT | -f FILE) [-l LANG] [–debug] destination

The first statement uses text from the command line, the second one calls a file from a directory which is just a text file with phrases and the last one is the syntax for the statement.

To move further on in the project, the goal is to end up having some sort of conversation. To do this, pocketsphinx can be used with gstreamer and python. Gstreamer is a media framework and can be used for audio streaming and processing. I followed the instructions from the following website:

<http://cmusphinx.sourceforge.net/wiki/gstreamer>.

Even though I set the path for the environment variables for both the gst and pocketsphinx libraries, I had issues finding the pocketsphinx plugin for gstreamer while running the following statement:

gst-inspect-1.0 pocketsphinx

When I checked in my file directory I saw that I had another gstreamer version installed when I installed Ubuntu extras. I was able to get gst-inspect-0.10 pocketsphinx to work by figuring out the correct path for the environment variables. While using gstreamer version 0.10 I was able to run the example python code given on this website:

<http://cmusphinx.sourceforge.net/wiki/gstreamer>

However, the following are the errors I encountered:

Using pygtkcompat and Gst from gi

Traceback (most recent call last):

 File "recognize.py", line 95, in <module>

 app = DemoApp()

 File "recognize.py", line 23, in \_\_init\_\_

 self.init\_gst()

 File "recognize.py", line 44, in init\_gst

 asr.set\_property('lm', '/usr/local/share/pocketsphinx/model/lm/6015.lm')

NameError: global name 'asr' is not defined

By commenting out the two asr.set\_property statements for recognition accuracy, I ran into another error which basically said no element pocketsphinx found. After some research I learned that the problem may have risen when I first installed the pocketsphinx library certain gst-plugin files were not necessarily downloaded.

The following blog, <http://jonathankulp.org/blog.html>, describes the files that may be required to find pocketsphinx for the gst-plugin. These are the files that the author suggests to look for:

pocketsphinx-0.8/src/gst-plugin/.libs/libgstpocketsphinx.a

pocketsphinx-0.8/src/gst-plugin/.libs/libgstpocketsphinx.so

pocketsphinx-0.8/src/gst-plugin/.libs/libgstpocketsphinx.la

I was not able to find these files in the pocketsphinx folder suggested. The solution was finding the correct path to the environment variables. On top of that another gstreamer plugin was missing. There was an issue trying to receive or record input from the microphone. The error that showed up was “GLib-GIO-CRITICAL: g\_bus connection failed”, “Cannot connect to server socket err = No such file or directory”, and “Jack server is not running or cannot be started”. This required Multimedia System Selector (MSS) to be installed as well. After installing MSS, the example python code had no issues running. Basically when the code runs, a graphical user interface (GUI) will show up with an empty text box and button to recognize speech input from the microphone and display it in the text area.



Figure 4. This snapshot shows the GUI that should display text of the input received once the speak button is pressed.

Since the GUI was running, I decided to figure out if I needed to import an ‘asr’ manager since I was previously missing some plugins. I downloaded, extracted and installed the required files for the asr manager because it was integral to improve the accuracy of the of the speech recognition. However, python detected that ‘set\_property’ was not an attribute of object ‘asr’. Below is a snapshot of the error that was generated when I tried to run the speech file.



Figure 5. Asr module error generated.

**Issues:**

There were several hurdles I encountered while trying to utilize the tools and packages available to build a voice recognition engine. Building and developing a speech recognition engine is not simple. A lot of the packages, libraries and software are open source and at times the links to certain files would be missing, broken or outdated. Without the required files, it’s difficult to progress forward with the project. On top of that everyone uses different versions of software and hardware so there are issues with compatibility.

When I started this project I was going to use another text to speech engine called Jasper. Japser is an open source software and the necessary information can be found at the following website:

<http://jasperproject.github.io/documentation/>

Jasper is always on and listening to voice commands. From the voice commands given, the software can deduce what you program it to know. For example, you can ask for information, get notifications from any of the social media or email accounts that are synced, control your home and more. To initiate help from Jasper, you have to say ‘Jasper’ and then follow up with a question. Jasper will then look for keywords in your speech and deduce the best answer for your question. Three weeks in, I learned that I would not be able to use Jasper as the speech recognition tool.

After following the instructions to install the required software I worked on trying to download and configure Jasper on the Raspberry Pi using Method 1 which was the quick install instead of Method 3 the manual install. However, I found out that I had a different version/model of the Pi and the disk image available was only available for a certain version of the Raspberry Pi. This setback cut time into moving further into working on the speech to text recognition. Also, the micro SD card that I had got damaged while formatting, so the card reader that I had did not work, which let a day go to waste because I had to wait another day to get another card reader and reformat the micro SD card. Even though, I found the correct image file for the Pi that I had which was the Raspberry Pi 2, the operating software for Jasper had some issues while configuring the files. Jasper required a user profile, Speech to Text (STT), and Text to Speech (TTS) engine. For example, while trying to configure the Google STT and TTS engine, the terminal threw some errors. In addition, while replaying a recording I could not hear an output from the speakers. There were some files missing from the ALSA library which is used for audio. Since there were critical files missing from the Jasper image installation, I decided it was best to work with a different speech recognition tool instead of taking up more time trying to install, fix the same software or waiting another week receive the recommended hardware if I purchased it.

Even though I decided to use Pocketsphinx I realized that using the Raspberry Pi as the base of the speech recognition tool would not work because I was primarily having issues installing the software required for the audio portion of it. There were errors installing certain files for ‘ALSA-base’ which is used to decipher the incoming audio. On top of that even though I was able to record a short 5 second clip, I could not hear the output from the audio jack. I found out that the OS Raspbian Jessie may not have the ‘ALSA-base’ file, so I tried to use Raspbian Wheezy. This required me to expand the file system because there wasn’t enough space. We expand the file system to use whole space on microSD card. And after expanding the system you have to reboot for the changes to apply. However, even after installing a different version of Raspbian the same issue occurred.

After some research I found out that I could use pocketsphinx on Ubuntu. However, since I did not have an empty external hard drive or a large enough flash drive, I ran Ubuntu on my laptop without installing it. I followed the instructions on the following website and I was finally able to get the software to work:

<http://www.robotrebels.org/index.php?topic=220.0>.

The instructions were detailed and helpful. I also tested out the library listed for speech recognition. It was a short list of five to six phrases. To generate the required files for the tool to decipher what is said, there is an online tool via Carnegie Melon which helps generate the additional files needed for the speech recognition tool. Since I ran Ubuntu on my laptop without installing it, everything I worked on got erased. I planned to get Intel NUC (Next Unit of Computing) from work and install the Ubuntu OS on it and install all the required files and packages on it while building the speech library.

Frustrated that there was not much progress in getting some communication going using gstreamer with python, I decided to see if I could get some interaction if I used a different speech engine. I was suggested to look into a blog called pi robot which uses ROS (Robot Operating System). ROS has frameworks used for robot software development. Since, I had already been able to accomplish certain parts in the project such as being able to build a language model that takes input from the microphone and recognizes if that said word/phrase is in the library, being able to detect what is being said and also getting the gTTS library to work, I did not want to move or jeopardize all of the effort I had put in installing and building pocketsphinx. I decided that I would get another hard drive and install an older version of Ubuntu since I wanted to use a version of Ubuntu that was compatible with the instructions given by the following blog on how to get a speech recognition engine working:

<http://www.pirobot.org/blog/0022/>

The blog is outdated and uses older versions of ROS and Ubuntu and it seems like some of the ROS syntax has changed or is only compatible with older versions. Since I installed ROS jade and Ubuntu 14.04 the following statement uses an older version of ROS (electric):

sudo apt-get install ros-electric-sound-drivers

Apparently ‘sound-drivers’ is not a command anymore and I had to substitute it with ‘audio-common’. I was able to download and install the pocketsphinx driver and packages required. However, I had issues installing the tutorial, which in turn didn’t allow me to test the pocketsphinx recognizer either. Here are some snapshots of the errors I received:



Figure 2. Unable to find the files required to run the tutorial, even though they were retrieved from an online repository.



Figure 3. ROS unable to run example code recognizer.py from .launch file. Log file indicates the following: “use\_sim\_time is not set, will not subscribe to simulated time [/clock] topic”. However, in the particular file required to set ‘use\_sim\_time’ to true does not contain the statement.

In the end, I decided to go back to just using pocketsphinx because I don’t feel that I would accomplish much using the instructions from pi robot. The learning curve for ROS is too steep to be figured out in a week.

**Conclusion:**

In conclusion, trying to build a speech recognition engine is definitely difficult. Even though I had some issues integrating pocketsphinx with gstreamer and python to create two way communication, I learned a lot from debugging certain problems and using a Linux based operating system. A lot of the libraries and packages used require various other plugins to be installed before the software recognizes certain statements. I learned that when the speech recognition tool is running in the background, the surrounding environment matters as well. It is best if the system is in a quiet area. Also, there could be an issue with the microphone used. A higher quality microphone could definitely help the system with deciphering the phrases being said, so when the phrase is shown on the GUI, the correct input shows up. Moreover, due to time constraints and certain plugins missing, I was unable to integrate gTTS with pocketsphinx to create a two way communication.

**Works Cited:**

"CMUSphinx." *CMU Sphinx Downloads [ Wiki]*. 11 Feb. 2015. Web. 19 Mar. 2016. <http://cmusphinx.sourceforge.net/wiki/download>

"CMUSphinx." *Using PocketSphinx with GStreamer and Python [ Wiki]*. 28 Dec. 2015. Web. 22 Feb. 2016. <http://cmusphinx.sourceforge.net/wiki/gstreamer>

Durette, Pierre Nicolas. "GTTS 1.0 : Python Package Index." *GTTS 1.0 : Python Package Index*. 15 May 2014. Web. 19 Feb. 2016. <https://pypi.python.org/pypi/gTTS/1.0>

Ralph. "RobotRebels.org." *Installing PocketSphinx on a Raspberry Pi*. 19 Oct. 2015. Web. 6 Feb. 2016. <http://www.robotrebels.org/index.php?topic=220.0>

Saha, Shubro, and Charlie Marsh. "Welcome to the Jasper Documentation." *Jasper*. Web. 8 Jan. 2016. <http://jasperproject.github.io/#about>.