NUBots communication module – instruction set

Using the software

To use the software run the communication driver class. This should compile and output the starting messages for PocketSphinx and Rasa TensorFlow. After this a continuous loop will run stating “listening” which will listen for commands on the default microphone for the device.

Installation and environment

The installation and environment for the git project is all done through python, utilising the pip package management system.

1. Ensure python 3.7 is downloaded and installed properly with a reference in PATH
    a. Python 3.7 can be found at the following link
        i. https://www.python.org/downloads/release/python-370/
2. ensure that pip is installed. Instruction for this on windows can be found here:
    a. https://phoenixnap.com/kb/install-pip-windows
3. Install the following packages by running the following commands
    a. “Python -m pip install speech_recognition”
    b. “Python -m pip install swig”
    c. “Python -m pip install rasa”
    d. “Python -m pip install tensorflow”
    e. “Python -m pip install ffmpeg”

After these have successfully downloaded and installed, the project should compile and run without any errors. Refer to troubleshooting to look for any known errors in establishing the development and runtime environment. If any errors with unknown packages occur, this can most likely be solved with using the command pip install [package_name] which should resolve any dependencies and requirements that are missing. However, the above list covers all core modules and should encompass the entire system.

ASR model adaptation

To adapt the pocket sphinx model, follow the following steps. This instruction set was generated from the following site:

https://cmusphinx.github.io/wiki/tutorialadapt/#adapting-the-acoustic-model

1. Generate the clips
    a. Each clip must be clear and accurate to have a meaningful impact on the model
    b. They must be of the .wav format
2. Name each clip zero through number of clips that are being used
3. Create a transcript and id file that has the data of each clip contained in each line. The data is of the form
    a. Transcript - <s> this is where the transcription of the spoken clip goes </s> (arctic_0001) where s contains the transcript and the brackets contain the corresponding name of the file, (in this case 0-number of clips) – named Name.transcript where name is user choice
    b. File id – each line containing the name of the .wav file being used ie 0-number of files all on a newline – named Name.fileids where name is user choice

4. Locate the speech_recognition package and make a copy of the sphinx model (to use as a backup)
5. Copy the sample files, transcript file and ID file into the new model directory
6. Ensure that the sample rate data is known (this is up to the user discretion as there are multiple ways to learn this information)
7. Ensure that the sphinx_fe tool has been downloaded from sphinx base (this is up to user discretion and more details on how to find and use this are listed here: https://cmusphinx.github.io/wiki/tutorialadapt/#adapting-the-acoustic-model)
8. Ensure the map_adpat file has been downloaded from sphinx base (this is up to user discretion and more details on how to find and use this are listed here: https://cmusphinx.github.io/wiki/tutorialadapt/#adapting-the-acoustic-model)
9. Run the following command to format and process the user data:
10. Path_to_sphinx_fe_tool -argfile path_to_new_model -samprate 16000 -c path_to_model\Name.fileids -di . -do . -ei wav -eo mfc -mswav yes
11. Run the following command to adapt the model
12. path_to_map_adpapt_tool -moddeffn acoustic-model/mdef -ts2cbfn .ptm. -meanfn acoustic-model/means -varfn acoustic-model/variances -mixwfn acoustic-model/mixture_weights -tmatfn acoustic-model/transition_matrices -accumdir . -mapmeanfn en-us-adapt/means -mapvarfn en-us-adapt/variances -mapmixwfn en-us-adapt/mixture_weights -maptmatfn en-us-adapt/transition_matrices

NLU model commands and use

The following two rasa commands are the only ones that should be needed when utilising this project, any others are most likely used for setting up new project or changing the settings of the current project. (not recommended due to the project dependence on the current format). All information on the rasa suite of technology is available here:

https://rasa.com/docs/

Command 1 – "Python -m rasa train"

This will train a new model based on the current iteration of training data and project configurations. Changing the nlu.yml file with new intents/entities or with new training data will require this command to be run to rebuild the model based on the new information.

Once this command has been run, to integrate the new model into the project, first extract the model tar file and then change the model_name variable to point to the newly extracted folder.

The project will now automatically run and use the new model.

Command 2 – "Python -m rasa test nlu --nlu train_test_split/test_data.yml"

Where train_test_split/test_data.yml points to a separate testing dataset. This command will produce the testing results and illustrate the models performance. Further testing can also be done with other rasa commands including the ability to split the current training data into training and splitting (however this will hinder the models potential as it lowers the available training data).

Troubleshooting and known problems with installation

Swig

There is a known problem wherein installing swig through pip will not properly execute and lead to errors at run time execution. This can be solved by manually download the swig files and adding the to the path of the system directory. This process is outlined here:

https://stackoverflow.com/questions/33745389/swig-not-found-when-installing-pocketsphinx-python

This will enabled PocketSphinx to install and run properly.

Python versioning

Python 3.7 has been carefully chosen for the project due to its compatibility with all the required packages. If a different python version is required this can still work with the project, however care should be taken to ensure that all packages are compatible.

Rasa Packages

Rasa may require the individual packages to be installed to ensure that no runtime errors occur, If this is the case the most important package is the rasa NLU core.