Interaction

1. Description

You can see your VA as composed of different parts:

- his `policy` : meaning the way it reacts to a certain human-utterance, which is based on the activated/hacked/coded Skills he has at his disposal, depending also on his state.
- his 'limb', the way he talks: through the gpt-2 ML model, fine-tuned on the data you fed him.
- his state: encompasses both its memory and some custom parameters (specified by the human before an interaction, randomized or default).

These processes will be detailed below. The exchange with your voice assistant consists of three parts: interaction, growth (with ML), and self-Quest (optional).

Interaction

The direct interaction with your voice assistant has several steps:

1. Voice assistant 'hears' what you say (your voice is converted into text, and in turn classified as intent; "what is being asked of me?")
2. If an intent is detected, it will trigger the corresponding skill, depending on the skills and triggers you will choose for your project. Default implemented skills include Wikipedia, DuckDuckGo, remembering what you said, record/play, laughing, etc. Cf. Below for more details about Mycroft Skills…or customizable Skills.
3. If no intent is detected, VA will respond through machine learning (as a so-called 'fallback Skill' we implemented). The mode of this response and other parameters can be provided as input of the interaction, default ones, or randomized.
4. If specified, your conversation is stored in the memory and used for further training of the machine learning algorithm (Mycroft's expressivity). This conversation is recorded in the file whatVAHeard.txt in the data folder.

Growth

Your voice assistant's verbal way of communicating (in the fallback Skill implemented) will be progressively influenced by your interactions with it and the data you choose to 'feed' it. This is done with the help of machine learning, and more precisely by fine-tuning a gpt-2 language model through the Jupyter notebook on the data you select. To have access to more computational ressources, we use Google Collaboratory to train this model. For this step, look at the Google Drive, in the limb folder, and follow the quick Machine Learning guide there.

Self Quest (optional)

This feature is a trace of our own Unfamiliar Convenient project, and can provide a simple visualisation of what the VA Heard: it is optional for you to use it or not at all. For the ones which have less time available, it is advised to skip the Self Quest and focus on their VA. :)
The voice assistant also grows its knowledge of 'Self', based on your interactions with it. Note that the 'Self' concept of your VA is a dynamic networked concept depending on your interactions with it. The Self is seen as a set of concepts (nodes of a graph), with various weights between and connections between them, called edges. By running selfQuest.py script, words from previous conversations will be extracted and looked up on Wikipedia, and compared to the concepts already present within the 'Self' (semantic proximity analysed). If these concepts are close enough to the Self, they will be appended with a certain importance (weight).

2. GETTING STARTED

1. Once you have defined your project, it is time to interact with your VA, through interact.py
   If it is the first interaction, you can simply run it and test a few skills randomly (only the ones installed by default are activated for now, or the ML fallback skill). Later on, you will customize this interaction (both ML model and skills) to fit your project.
2. To interact with your VA, open the terminal, navigate to your mycroft-core folder and type:
   ./workshop/scripts/runInteract.sh
   Note that by default, your interaction is recorded in whatVAHeard.txt, for your VA to later evolve from it (the ML notably). If you do not want this, open runInteract and change the value of the argument ifEvolve to False.
3. At the end of your first interaction, you can have a look at it at the file /workshop/data/whatVAHeard.txt, to see what he recorded.

   Further: In the script Interact.py is also specified a parameter keepThreshold: this is indicates that above a certain character threshold, the answer of Mycroft is also recorded in the files whatVAHeard.txt, for him to learn from (to train his ML algorithm on). For instance, if Mycroft is answering the weather, you may not want to keep track of this information, but if Mycroft is reading a wikipedia article, you may want to record it. If you do not want any speech of Mycroft to be recorded, you can put this character number very very high.

3. Self Quest (optional)

Hatching 'Self'

1. To 'hatch' a new seedling of a voice assistant, you first have to provide it with a baseline 'idea'. This should come in the form of text. Choose whatever type of writing you want based on the project you have in mind.
2. Replace the contents in /mycroft-core/workshop/data/hatchVA.txt with your text
3. Now right-click anywhere in the Linux window and click Open Terminal
4. Navigate to /mycroft-core/ folder. Now launch the runSelfQuest.sh by typing:
   ./workshop/scripts/runSelfQuest.sh --firstTime=True
   What this will do is activate a virtual environment which allows to access various hidden functions of Mycroft, as well as keeping the project contain; it will automatically launch Mycroft if it is not yet running; Finally, it will launch the required scripts to set up the required bits for Mycroft to start mapping his Self. Beware, this Quest may be long, as he has to compute all the semantic similarities.
5. All (optional) parameters that you can enable are:
   --firstTime(=True/False): Indicates if it is the first time you start a selfQuest.
   --walkNetwork(=True/False): whether at the end creation of the self graph, VA should 'walk on the network' speaking aloud a few concepts.
   --audibleSelfQuest(=True/False): Indicate if the SelfQuest should be audible. If yes, you can hear part of what Mycroft is doing (look for words on wikipedia etc.).
   --visualizeGraph(=True/False): Indicate if the self Graph should be visualised at the end.
   --ifMLDrift(=True/False): Indicates if in between the quests of concepts your VA shall do a machine-learning drift (Only make sense if the quest is audible).
   --lengthML(=number): Character length of the machine-learning drifts (if any)
   --nSimMax(=number): When the VA looks for a concept on Wikipedia, nSim indicates the number of words it will try to match with this new word, to see how similar they are.
   --nSearch(=number): the maximum number of words the VA should look for.
   --lengthWalk(=number): how many concepts are spoken out during the 'walk on network'
   --finetuned_ML_model(=True/False): if you want to use your fine tuned ML model or the default gpt-2. If you did a fine tuning, say True, else false.
   --path_finetuned_ML_model: path to where your fine tuned ML model is, if you trained it.

Self-Mapping

4. Later on, once you interacted enough with your VA, you can decide to launch a Self Quest via ./workshop/scripts/runSelfQuest.sh. Before launching this script, open it, modify its arguments as you like (cf. step above where they are detailed), and save. Do not forget here, to put back firstTime to False!
5. Once you are ready, go in your terminal, navigate to /mycroft-core and type
   ./workshop/scripts/runSelfQuest.sh
   Beware, this Quest may be long if the number of words looked upon is high.

4. CUSTOMIZATION

The framework given is just a starter. You can now feel free to tweak it, tune it, and add some skills, or any other patch of code. Below, we detail a few options for the customizations, more or less difficult (as precised), with some links and references given. Before going on further, some terminology:

_utterance - Phrase spoken by the User, after the User says the Wake Word.
_intent - Mycroft matches utterances that a User speaks with a Skill by determining an intent from the utterance. For instance, for 'Hey Mycroft, what's the temperature like?' the intent will be identified as *temperature* and matched with the *Weather Skill*. One Skill can have several intents.
_dialog - A dialog is a phrase that is spoken by Mycroft. Different Skills will have different dialogs, depending on what the Skill does.
-FallBack Skill: a Skill that is designated to be a 'catch-all' when Mycroft cannot interpret the Intent from an Utterance.
-handle_nameIntent: is the name of the function Mycroft in each skill use to handle each intent.

1. Recording Time

The maximum timeout for recorded user message (RECORDING_TIMEOUT) is 10s by default. In case your project involve you speaking longer than 10s to Mycroft, this can be increased in (in ResponsiveRecognizer Class):

`/mycroft-core/mycroft/client/speech/mic.py`

Also you can alter other parameters like RECORDING_TIMEOUT_WITH_SILENCE, etc.
Alternatively, you can add it (overwrite) in the config files:

`/mycroft-core/mycroft/configuration/mycroft.conf`

NB: Adding parameters to .conf enable to modify them via the client interface later notably.

2. ML-FallBack Skill

In the proposed framework, the default fallback skill invoke a ML model to react to what you say if no other skill is triggered. This gpt-2 model, as explained in the guide dedicated, can be fine tuned on any data you feed to him. There are also a few other options in this skill we prepared that you can customize according to your project. For this, navigate up to the folder mycroft-core/opt/mycroft/skills/fallback-MLdrift/. Open the file __init__.py. There, you can alter different parameters of the ML drift defined in the preambule of the code: lengthDrift, number drifts, etc. Some are classic parameters of ML language model --others are omitted for simplification, but you are welcome to check further about gpt-2. Notably, one customization available: you can modify and choose a few '**modes**' (replace their name) according to your project. These will slightly influence the turn of the machine learning algorithm, as they will be fed to the machine learning as context. For each of these modes, you have to choose a few sentences and overwrite modeSeeds. For each mode, choose also its probability (in probaMode).

NB: Also, as soon as you have fine-tuned your own ML, do not forget to change here the parameter finetuned_ML_model=True. And check that the path pointing to your ML model is correct.

3. Mycroft Skills

(no coding skills required)

Some skills are installed by default with Mycroft.
Default Skills are listed here along with the commands to use to trigger them.

More **'certified' skills** can be found in the Skills Marketplace: pick the one(s) fitting your project.
Here the corresponding Github repository for the ones looking at code.
For all these, you can uninstall and install them very easily, by voice or command line through the MSM Mycroft Skill Manager once you launched mycroft.

```
(1) launch mycroft virtual env: $ source ./.venv/bin/activate
(2) add a skill: $ msm install skill-name
(3) remove a skill: $ msm remove skill-name
```

NB: Uninstall skills you do not want to use avoid some unwanted triggers.

If you want to go further, you can also have a peek at all the skills available on Github, even the ones which have not been officially tested/marketed. Here you can look at the ones in 'pending Market' or 'not in Market', classified by themes. Of course, with these, you have the risk to run into some issues, but you can try them out! For these, you would need to download them manually, and put them in your folder /mycroft-core/opt/mycroft/skills/ , then start mycroftto test them.

We advise you at the beginning to focus only on a few skills which could fit your project.

4. Custom Trigger / Hack Skill

(no or few coding skills required / some coding)

If you want to use an existing skill, yet modify how to trigger it, you can go in the skill folder to find it, go in the skill-name/vocab/ subfolder, and modify the .voc or .intent files you find related to the intent you want to modify.

```
Ex: if I want that Mycroft laugh whenever I say 'human', I go to the
vocab file of Mycroft laugh and add to the Laugh.intent a line with
written human
```

To understand better about intent, --for some more complex skill-- you may have to look at intent parser: the way Mycroft identify and handle your intent.
There are two possible used by Mycroft which can be used in a Skill:
Padatious, a bit easier to use.
Adapt, a bit more flexible and widely used in the community.
A tutorial about Intent in Mycroft can be found here too.

You can also **hack** the skill further:
(1) modify the .dialog files in the skill-name/dialog/ subfolder, you can change the way Mycroft will answer for some skills.

Ex: If I want when I ask the weather that it answer me 'It's hot. ´Global warming, init?", I go to the file current.hot.dialog and replace by this line what is written. Actually, the weather Skill has a whole bunch of other dialog to check...

(2) (if you code) change what the skill does: by modifying the handle function(s) in _init_-py, you are changing how Mycroft react to a certain intent.

Ex: If I want the Audio skill record several files, or that the remember skill spit something randomly from the VA memory, its a few lines of code ahead.

Here more about the skill structure in Mycroft in general.

5. Create new skills

(some coding skills required, depending on skill complexity)

Before, you can start thinking about what you want to create. Think in terms of "If VA hear …. then VA do/look-for-data-of/ …. and say ...".
Ie the components to decide are:
(1) What words will the Human need to speak to activate the Skill? >in /vocab/.
Except for fallback Skill.
(2) What data will the VA need to look for? > coded in handler function in _init_.py .
Your Skill may need additional packages, dependencies, API, etc.
(3) What will the VA speak in response? > in /dialog/ .

In case you want to venture in creating your own skills, you should have a look further at the documentation beforehand, or at least the skill structure in Mycroft. Then, you can start with a template, or a simple skill 'hello world', or with this to create a FallBack Skills. You can also look at some Github examples to get the idea of how skills are coded.
A tutorial here to build a hello-world skill.

Also, there is also a ready-made Mycroft tool mycroft-skills-kit (msk) to create your own Skills you can use. Here is an explanation. You can do it through the terminal directly, once you started Mycroft.

Nota Beme:
_ Once created, the skill should be placed in /opt/mycroft/skills/MySKILL Not in /mycroft-core-mycroft/skills/MySKILL
_ To understand better about Intent Parser (Padatious, Adapt), cf. 3. 'Custom Trigger' section too.
_The utterance string received from the speech-to-text engine is received all lowercase. As such any string matching you are trying to do should also be converted to lowercase. Compound words like "don't", "won't", "shouldn't" etc. are normalized by Mycroft - so they become "do not", "will not", "should not". You should use the normalized words in your .voc files.

6. Custom your wakeUp Work

This is a possible option with Mycroft, yet require some extra training.

5. References-further

Full Documentation

About Skills-settings Skill settings allow users to customize their experience or authenticate with external services. Learn how to create and use settings in your Skill.
Skill settings provide the ability for users to configure a Skill using a web-based interface

Conversation