Guide to chatterine's underlying logic

Data

The google spreadsheet called 'data' is where the bot's logic is stored.

In the sheet called 'MODIFY_CONTENT_HERE' are all the observations that are fed to the ML engine that helps the bot choose what it should reply to a given user to look like it's understanding what he or she is saying.

There, they can be labeled, firstly, according to the first level of topics under which they should be classifiable, secondly, according to the topics of the second level belonging to the topic to which they have already been ascribed, and lastly, according to the 'intents', or in other words, the most specific semantic categories of the ontological structure, which should stand for the gist of what the bot can respond to what the chatbot user may, according to the annotator, be referring to with what he or she is expressing in what they are sending to the bot. The emission will then appear directly to the right of the fully-labeled observation.

The sheet called 'ontology', true to its name, holds the bot's ontology, namely, all the semantic categories under which the bot's emissions have been grouped, both those directly linked to intents and those that should serve as fallbacks.

The sheet called 'named_entities', I know, I have invested in creativity elsewhere, I promise. Named entities are, according to the logic I have followed, the categories representing the lexicality of the words the bot should understand as semantically equivalent and thus, as interchangeable, given the context.

'Vocabulary' is the sheet where I have stored, first, the most frequently misspelled words of the English language, and secondly, the words I wanted my system to give priority as candidates for replacing misspelled words.

Annotation Tool

There is one last sheet that is generated on executing the code on the google colab page called 'annotation_tool'. The sheet is called '2annotate_by_labels', and is meant to allow annotators to add utterances to labels with more or less intents ascribed to them. This sheet should manually be renamed to 'annotated' once the google colab page has finished running. That way, the next time 'annotation_tool' is run, the observations contained in that sheet, which, by then, should have been filled out completely, will be added to those of the main data set, to be found in 'MODIFY_CONTENT_HERE', together with those that were saved in the csv-file 'obs_from_website', corresponding to the interactions the bot has had with the users of the site.

'Annotation_tool' also trains a logistic regression with our complete data set, to let us inspect thanks to eli5 by which sequences of words the ML model's decision on what seems most representative of which semantic category is most influenced. I believe this information could prove valuable to perform the annotation task as well as the task of increasing or trimming the data contained in the sheets 'vocabulary' and 'named_entities'.

Handling Logic

Compared to *data*, *logic* is chatterine's frontal lobe. Basically, where the action takes place. It is where chatterine decides what to reply taking into account more than just a model's indication on how to proceed. For instance, if the emission the model recommended has already been sent in that conversation, if the conversation has been carrying on for a longer or shorter period of time at

any given point, if the message the user sent contains an email address, if it contains a funny word from the emission that was sent right before by the bot, …

If an emission doesn't meet any of the criteria that would allow it to qualify as a good candidate to act as the bot's reply, a fallback emission of the second level of topics is checked against those same criteria. If no fallback emission could be chosen either, the best fallback emission of the first level of topics will be selected and rendered.

Besides, logic also saves the new user queries alongside the models' predictions on them in a csv-file, so that they can be labeled and added to the main data set, because they are on what the model's performance should be measured.