ARGUS Chatbot Documentation

Peter Wright

**Summary:**

The ARGUS chatbot is a chatbot that reports to the user the emotions it has found in the conversation based on the path similarity of the words used by the user. The program uses the Natural Language Toolkit (NLTK) python library to find the similarity of words and part of speech. The emotion recognition is based on the emotion wheel by Robert Plutchik. This allows the emotions to be mapped graphically as points on a grid and then reported to the user in an easy to read format. While this program is created to be a chatbot, most of the python files created (all except ARGUSChatbot.py) were created to be used on any text entry.

**How to Run Program:**

1. Enter command prompt
2. Navigate to directory holding ARGUSChatbot.py
3. Make sure you have installed the python libraries chatterbot, nltk, and mathplotlib
4. Type “python ARGUSChatbot.py” to run the program

**External Python Libraries:**

* Natural Language Toolkit
* Mathplotlib
* Chatterbot

**Informational Sources:**

Donaldson, Melissa. April 27, 2017. “Plutchik’s Wheel of Emotions – 2017 Update”. Six Seconds.

<http://www.6seconds.org/2017/04/27/plutchiks-model-of-emotions/>

Bird,Steven, Ewan Klein, and Edward Loper. July 1, 2015. “Natural Language Processing with Python”. <http://www.nltk.org/book/>

**Class: EmotionPoint:**

The EmotionPoint object was created from a need to store a coordinate in both cartesian coordinates and polar coordinates. Plutchik’s emotion wheel works best when points are stored as polar coordinates, but tracking them and finding their average works best when they are cartesian coordinates. So the EmotionPoint class stores both values for each point and has built in functionality to update all the other values if either the x, y, r, or phi value is changed. The class also has a function to return the emotion associated with the point based on the Plutchik emotion wheel.

**Class: EmotionGraph:**

The EmotionGraph object is holds a list of the stored EmotionPoints, the average of all the EmotionPoints, a matplotlib image graph, and a dictionary holding the emotion and where it will be plotted. The actual graphing is done is done by matplotlib in the object’s plot method. Other methods are used to configure the average EmotionPoint and add EmotionPoints to the graph based on keywords (emotions used in Plutchik’s emotion wheel) or using EmotionPoints. The EmotionGraph also has two different ways of plotting that allow either displaying the average or the path the user took.

**Class: SynLink:**

The SynLink object is used to find the closest emotion to a word. The SynLink holds a dictionary of nltk lemmas used for checking path similarity through NLTK’s synsets. Synsets are a tree structure that connects synonyms in a hierarchical structure of sets. Other programs can send it strings through its methods and it will return the closest emotion to the word given or a string “none” if it was unable to find anything.

**Class: Response:**

The Response object is a container for the user’s input string as well as an EmotionGraph object and a SynLink object. The Response splits up and iterates over the input sentence and then parses it using a SynLink object. It then saves the emotions returned by its SynLink to its EmotionGraph. At the end of it’s construction it outputs the EmotionGraph in the plotting the average format.

**ARGUSChatbot:**  
 The ARGUSChatbot file is not an object but a driver file to run the created classes in tandem with a chatterbot. The program first trains the chatbot and then displays the introduction, stating how the program displays information and how to end the conversation. The conversation happens in a while loop that is only broken when the user responds with “goodbye”. Each response given is saved as a Response object and has its average point saved for final display at the end. At the end of each iteration of the loop the Response object is deleted to clear out the data from the previous response. At the end the program displays how the emotions changed throughout the conversation through a message at the bottom and an EmotionGraph.

**Screenshots:**







