Aubrey Dority

Text Mining Project

I analyzed the messages in my Facebook inbox using pattern's sentiment analysis and modality analysis. I wanted to be able to rank my Facebook friends by how 'happy' and 'confident' they had come across in their messages.

In order to do this, I first had to get the information from Facebook. I found a blog which led me to facepy, a package that let me query Facebook's API for information like the ID of the author of a message, the latest snippet of text in a message thread, or the thread ID. Unfortunately, Facebook's API is designed to make it really difficult to access large amounts of information about users, so I couldn't query for actual messages as I had hoped. I tried working around this by getting a list of all the thread IDs in my inbox and then asking for the messages in each thread, only to find that the API wouldn't allow for any sort of loop, and wouldn't print the text corresponding to message IDs anyway.

After a while I admitted to myself that the closest I was going to get would be either taking the latest snippets of text in my inbox along with the ID of their author, or scanning through my friends' newsfeeds and grabbing text from there. I decided to go with the former option after looking at the newsfeeds and seeing that many were changes of status, tagged friend names, or links to articles, which I didn't think would be as relevant as messages from the people themselves.

This led to a fairly straightforward and rather simplistic code, which took the message snippets and their corresponding authors, ran the snippets through pattern's sentiment and modality analyses, and sorted the authors by their snippets' results. Using the returned lists, I was able to determine who among my friends were the most positive and negative, and the extent of their objectivity. The results seemed to emulate a bell curve, with a significant number of zeroes and fewer extreme negative or positive results. However, I found that some zeroes may have been due to the brevity of the snippets analyzed- the snippets were at most two sentences, and if they were too short, then pattern wouldn't be able to find words to analyze.

I was curious as to how pattern was getting the numbers it returned, so at one point I printed the assessments for each snippet (assessments return the word evaluated and the value it returns). One really cool thing I found was that emoticons changed the sentiment of a sentence- the emoticon ':D' had a sentiment value of 1.0 while ':o' had a value of .05. This proved incredibly useful for a project like mine where emoticons were conveying a good amount of the information in some of the briefer messages.

I learned quite a bit from this project but executed it terribly. First off I spent far too much time messing with API queries to try to get the messages I wanted, which didn't end up working and left the rest of the project rushed and a bit messy. It was really interesting to see how information on Facebook could be accessed through Python, but it detracted from the actual results of the project. I also realized after the fact that I didn't unit test per se- instead I'd print out various results as I went so I could see what was happening. This kind of worked for this project, since I wasn't sure exactly what I should expect to see in some cases and wanted to figure out how things were working by looking at what got printed. However, this 'method' led to really terrible time management and results since I was going through every step in turn and getting caught up on roadblocks rather than blocking out what steps I needed to take (along with unit tests to ensure they did what I wanted) and working within that framework. This is something I really have to change in my next project, for the sake of better code and my sanity.