Celine Ta

Project Overview – text_mining_main.py

Using the Pattern module and its built-in functions, I pulled and analyzed news from Google in English, French, and Dutch to roughly quantify how positively or controversially each language-speaking community viewed a input topic or person. Sentence-by-sentence sentiment analysis (from Patter) could be averaged over the content in each language and compared to yield a more holistic, but simplified view of global news and politics. Ideally, the program would be able to gauge not only where the language-speaking population lay on the spectrum of opinions, but also how controversial the topic is within that community.

Implementation

The program can be broken down into two functional parts: 1) language processing and 2) visualization. In the first part, the program asks the user for an input name. The program specifies proper nouns, specifically a person's name, to avoid confusion when Googling say, a French phrase (say, “fromage”) for news in English. Since the Google Translate API is a paid service, the program does not support common nouns well, nor does it account for languages that do not use the Roman alphabet. The program searches the term for English, French, and Dutch articles in Google, analyzes each sentence and averages the positivity and controversy. For the purposes of this program, controversy is calculated as the average of spread (average distance from neutral zero) and subjectivity.

The second part creates a bar graph. While Pattern has some really intriguing functions in pattern.graph, this program does not lend itself to nodes and edges. Instead, the program represents positivity and controversy in a bar graph, separated by language.

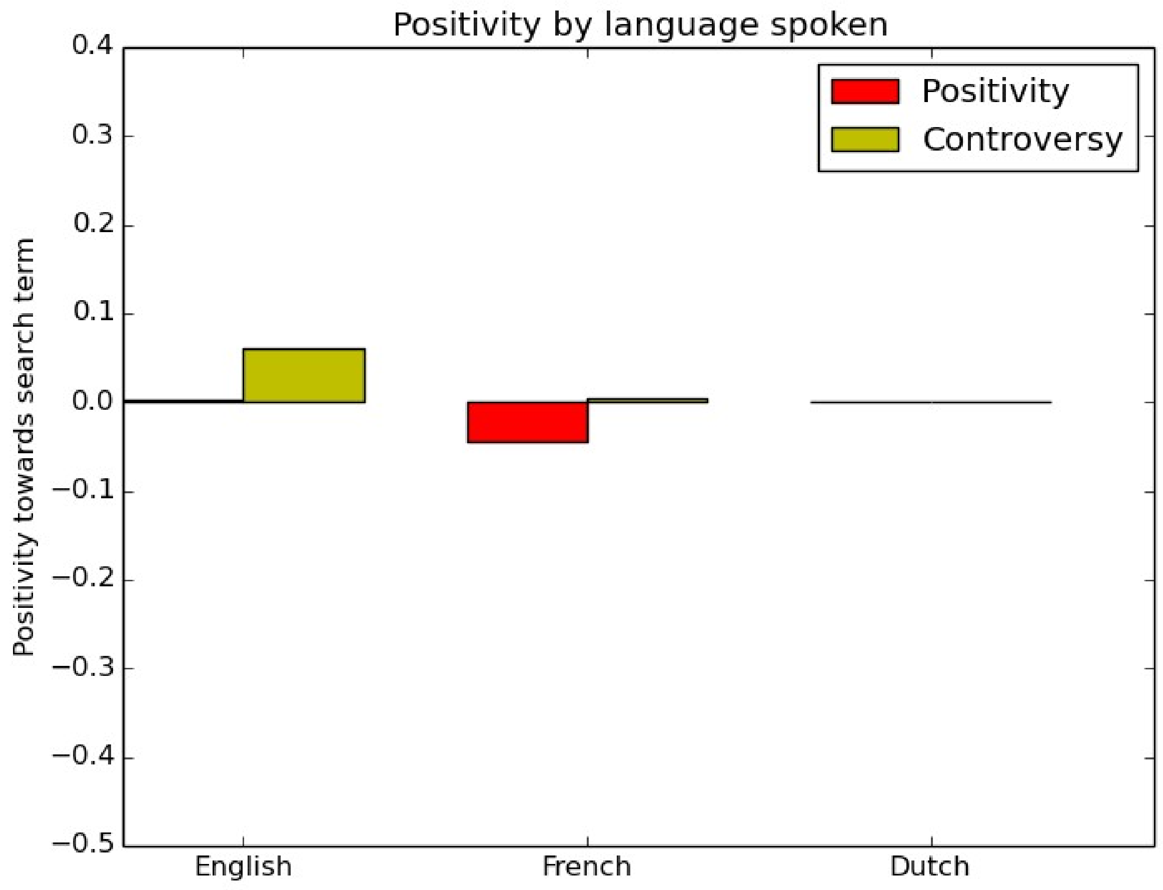
My implementation is one large script, without functionalization. Due to Pattern's structure, two options were available: import all functions from Pattern without differentiating between sentiment functions for each language, or import specific Pattern language packages with clear naming conventions for sentiment analysis. To avoid potential miscalls and readability, I chose to go with the latter, though it was far from ideal. This choice forced the program to repeat many similar tasks using the specific sentiment analysis function for each language as hard-coded repeated loops, rather than a function call. Since we use raw-input() to enter in search terms and it didn't make sense to functionalize anything else, given the program structure, there was no need to pass arguments in or add an extra layer between the user and running the program. If there were occasion for other programs to use this program's functionalities, putting the script into a large function is very quick.

Results

The program allows for any proper noun to be searched, but I searched “Barack Obama” and “Celine” as my two test cases. While America has positive views of Obama overall, for example, France had a slightly negative view. Keeping in mind that the news articles pulled are heavily influenced by recent events, and may also be confounded with national troubles or tragedies like Charlie Hebdo, the president's absence from the multinational rally in Paris does align with this analysis of France-Obama relations.

On the other hand, “Celine” rated highly in positivity in all three languages, most likely due to the PR skills of both the singer and the brand.

While the visualizations of these results were still in process when I reached the daily limit of Google searches, a sample result would look like the following:



Reflection

Overall, Pattern seems very powerful and had some tools that I'd like to play with more. I wish I had read Pattern's documentation outside of what was referenced by the assignment; for example, their .graph package seems very useful. I think I may have been able to do much more interesting and technically challenging work had I seen more of what Pattern could do. As it was, I was uninspired when coming up with project ideas; the scope is a smaller than it could be. That said, there are definite limitations, especially with the paid services and limits on search requests. Between poor planning on my part compounded with these limitations, leading to some challenges with debugging and producing a project on time. Unit testing proceeded as well as it could have, given the lack of functions. The script was built incrementally, starting with the Google search and then sentiment analysis.