Project Documentation: r/AmItheAsshole Sentiment Analysis

Team Information:

Captain (solo project): Samantha Ernst (sdernst2)

Project Overview:

For my course project, I developed a program that takes the link to a r/AmItheAsshole post and renders a color-coded word cloud based on sentiment analysis (negative/neutral/positive each corresponding to a certain color), a bar graph with the top emotions in replies, and another bar graph of the percentage of each of the 5 possible judgments. In a subreddit as controversial and divided as AITA, this program gives users an overview so that they can see at a glance what the majority of commenters are thinking and feeling. It can be used by casual viewers and avid readers of the subreddit alike.

Implementation Documentation:

All of the code for this program is in the Jupyter notebook *CS 410 Course Project.ipynb.* The program gets post and comment information from PRAW (The Python Reddit API Wrapper). It first fetches the post that the user has provided the url of. It goes through all comments on the post and sorts through which ones have actual judgments and which lack a judgment or are a reply to a comment. Only comments that make judgments on the post are considered. The text from these comments is pre-processed, using NLTK to tokenize the strings, remove stopwords, lemmatize the words, and remove the judgments themselves from the list of words.

NLTK is then used again to tag each word with its part of speech, and these tags are used to separate out the words that will be used for the emotion detection sentiment analysis and the polarity sentiment analysis. Notably, only adjectives are used for emotion detection sentiment analysis, because words like nouns do not inherently carry any emotion and often skew the results in the wrong direction.

The judgments from each comment are counted up at this stage, which is then used to make the bar chart showing relative percentages of each judgment in the comments. The polarity of each comment is also analyzed, and each meaningful word in the comment has that polarity score added to its own score. A word's total polarity score will come from its context and will signify whether the word was used in majority positive, negative, or neutral contexts.

Next, NRCLex is used to get the emotions related to selected words. These emotion values are added up and the top emotions for the post are displayed, alongside lists of the top words that contributed to this emotion. Word contribution is weighted by log(1 + freq /

max(sorted_comment_words.values()). This normalizes the frequency in terms of the maximum word frequency and makes sure that the weighting doesn't grow too quickly as frequency increases. A constant factor of 0.1 is also added when frequency is above 1 in order to give extra weight to words that were used more than once, and therefore are more likely to be relevant or a common thought among commenters.

The WordCloud library is used to generate a word cloud of the top meaningful words in the comments of the given post. The color of each word comes from whether its average context polarity is positive or negative. Polarity values are in the range of 0 to 1, so average polarity values are as well. In the case where anger is one of the top emotions, negative polarity words are displayed as red; when sadness is one of the top emotions, they are displayed as blue. Positive polarity words are always yellow, and neutral (or controversial) words that have an average polarity close to 0 are gray.

Usage: *(see CS 410 Tutorial Presentation* for link to video)

Access to the data about the Reddit post requires an api key (client id, client secret, and user agent) for PRAW. To get this information, register for a Reddit account and then go to https://old.reddit.com/prefs/apps/. Put the api key information into a .env file in your google drive and one of the code blocks will access these from your drive once you give it permission. If you don't feel comfortable with that, you can also paste them directly into the code block as long as they are passed into the PRAW authentication!

In order to run the program, open it up in Google Colab. This will require you to sign in with a Google account. Input the desired r/AmItheAsshole url in the first code block as a string. Run each code block in order by pressing the 'play' button and waiting for a checkmark to appear at the left side of the block, indicating that it has finished executing. The first few blocks are installs and set up utility functions that will be used later in the code. Then comes the initial processing, and then the last three code blocks output the graphs and word cloud.

Contribution:

This was a solo project, so I completed all of the work myself.

Resources Used:

PRAW documentation:
https://praw.readthedocs.io/en/stable/code_overview/praw_models.html

Part of speech tagging using NLTK:
https://www.geeksforgeeks.org/part-speech-tagging-stop-words-using-nltk-python/

Emotion detection sentiment analysis using NRCLex: https://pypi.org/project/NRCLex/
and https://www.geeksforgeeks.org/emotion-classification-using-nrc-lexicon-in-python/

Sentiment analysis with NLTK's VADER:
https://realpython.com/python-nltk-sentiment-analysis/

Setting up word cloud: https://www.datacamp.com/tutorial/wordcloud-python

Setting up a Matplotlib bar graph with bars below the y-axis:
https://stackoverflow.com/questions/60519582/matplotlib-bar-chart-negative-values-below-x-axis

Self Evaluation:

I completed everything that I intended to and I am very happy with the results of my project! I had to change a few things as the project went on due to time constraints, finding a more efficient method, etc. but was able to stay true to the initial goal and concept I laid out in the proposal.

Some things that I changed along the way are:

- I switched from the idea of web scraping to using PRAW in order to get the necessary data about the post and its comments. I looked into web scraping methods like Selenium and BeautifulSoup, but found that for my uses, PRAW was much more efficient, less complicated, and already had everything ready for me to use. I hadn't been aware of its existence when I planned out my project proposal originally.
- I also used a bar graph rather than a chart to display the top emotions in the comments. I felt that this was much more visually appealing and gave me the chance to differentiate emotions by applying specific colors for each emotion. I also ended up outputting the top (maximum 10) words for each emotion, along with their frequencies, in order to give the user context about what exactly was contributing to these emotions in the comment section.
- I decided to use a bar graph instead of a pie chart to display the percentages of each judgment in the comments for comparison. The layout is much neater and has fewer glitches based on the dynamic input it is receiving, and still allows for the user to get a sense of proportions at first glance.

Ultimately, I have already used this program a lot and find it really interesting to see what the program is able to conclude from the data (and what it may fall short on, given the inherent shortcomings of part of speech tagging, sentiment analysis, etc.).