VADER-based Sentiment Analysis of WhatsApp chats

Stefan Dusnoki

Introduction

This technical report details the inner workings of the sentiment analysis tool I developed. The program takes an exported WhatsApp chat file, validates the content, and by using the VADER library, generates a list of sentiment scores for each user, which is then charted to show trends over time. The report also includes a description of the program's inner workings, some limitations, along with my future plans for the project.

Justifications

I chose WhatsApp for this project, as it is the most popular messaging app among my peers. Therefore, I was able to quickly test my code and get results. VADER[1] was selected as my sentiment analysis library of choice, because it is, as described in its GitHub repository, "specifically attuned to sentiments expressed in social media". Unlike other sentiment analysis tools, it can also handle emojis, which appear ubiquitous in day-to-day communication.

Data Input & Format

The Sentiment Analyzer utilizes exported WhatsApp conversations that compute sentiment trend over time. The input `.txt` file contains one message per line, in the following format: `<Date>, <Time> - <Username>:<Message>`.

How This Works

Three main functions make up the entire program. The first one, `data_validation()`, takes the chat `.txt` file and validates the content, ensuring that no errors will appear when calling subsequent functions. Each message must adhere to the pattern described in the previous section, meaning that we can differentiate between two message types, starter and extension. To do so, we check whether the current line contains a date using the following regular expression: `"\b\d{1,2}/\d{1,2}/\d{2}\b"`. If the message is a starter, simply write it in the output file. If not (the message does not contain the date), remove the last newline character (`\n`), and attach the continuation to the starter in the new file. System messages are also ignored, as they are irrelevant when calculating sentiment scores.

The `sentiment_analysis()` function goes through the resulting `.txt` file and creates a dictionary that includes every user plus a list of sentiment values for each one. Starting from an initial value of 0, every subsequent message results in a sentiment score which is added to the previous value in the list. The resulting list displays a user's sentiment trend over time. Features such as the use of regex and a caching mechanism help the function stay fast and resource-light.

The final function, `get_charts()`, takes the resulting dictionary and creates a chart that displays the sentiment trend of each user over a set period of time.

Limitations

Despite functioning well, certain constraints might inhibit the program's ability to work effectively.

Firstly, successful running of the functions is contingent on the structure of the exported chats. If the official export structure is modified and no longer acts as described previously, the functions will cease to work properly.

Secondly, the program includes a plethora of admin messages that are filtered out, as they do not serve to calculate a sentiment score. As of July 1, 2025, the program filters for 31 admin messages. If there are any not already included in the list simply because I was not able to find them when building the program, that may lead to crashes.

Another weak point in the program's architecture is VADER's inability to properly handle languages other than English. The given text must be translated before passing it to VADER, which may skew the final sentiment scores.

Lastly, despite this not halting the execution of the program, a small amount of data may result in a peculiar looking sentiment chart, with large jumps to new values in a small amount of time or (seemingly) periods of stagnation. Therefore, it is recommended to use a larger amount of chat data to achieve the best results.

Future Plans

I plan to apply the knowledge I gathered from this project in similar systems in the future. By using Discord's API, I plan to create a bot that will calculate sentiment scores on servers, provide daily updates to the existing charts, and update the design of the charts to look more like a candlestick chart.

Closing

In closing, this project allowed me to dig into the world of sentiment analysis and gain some hands-on experience working with VADER, which I plan to learn more about.

I learned how regular expressions can help my program be leaner and faster, which was proven to me when I compared my initial solution to a problem with its regex counterpart using `%prun` (e.g. getting a date object using regex vs getting a datetime object using `datetime.strptime()`, which in comparison is much more computationally expensive).

Furthermore, by using docstrings to explain functionality and keeping the codebase organized, I ensure that everyone can explore the program for themselves in an easy and straightforward manner.

References

[1] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, (Ann Arbor, MI, USA), June 2014.