Overview of the function of the code (i.e., what it does and what it can be used for)

This project is a Chrome extension for sentiment analysis on recipe blog comments. Sometimes, not all the comments on recipe blogs have a star rating, or the ability to rank comments from most to least liked (or vice versa). This makes it difficult to tell from a quick glance, if the recipe has (overall) more positive or negative reviews. Additionally, users must often scroll all the way to the bottom of the blog to find the comments, and then read through them to determine the sentiment. This takes time.

This project aims to provide a summary of the reviews' sentiment for a particular recipe. The idea is to give the user a quick glance, to be able to tell if a recipe is overall more liked or disliked.

Documentation

The software was implemented in three main steps.

1. The first step was to scrape the latest comments from the given website (scrape_me.py). Each recipe blog is structured differently in terms of layout, HTML elements, and so on- it would be a much larger project to be able to scrape all the recipe blogs in existence. I opted to focus on one popular blog, Sally's Baking Addiction. I used the HTML elements on the page to select and save the contents and star rating of each comment (if it existed), excluding any additional metadata and excluding the responses to user comments from Sally or her team. This anonymized data was written into a local CSV file with two columns- Comment and Rating. If there was no rating, it was stored as 'None'.

2. The second step was to write and train a Naive Bayes classifier (naive_bayes.py). The classifier was written using Python, specifically with the pandas library. The Kaggle dataset titled "Sentiment Analysis" by the author "Gemmin" was used (https://www.kaggle.com/code/gemmin/sentiment-analysis/input?select=reviews.csv). This dataset includes recipes and their reviews from the website Food.com. Specifically, the reviews.csv file was used to train the classifier- this data was the closest I could find to classify recipes using sentiment analysis.
   The reviews.csv data was cleaned and modified, so it had only two columns- Ratings and Reviews. The text was also preprocessed to remove non-alphanumeric characters, and to make it all lower case. This data was then split into training and

testing data by an 80/20 split. The data was modified to be in the form of a bag-of-words representation before it was used to train the classifier. The testing set was evaluated to a 75% accuracy score.
The same transformation and preprocessing of data was applied to the scraped comments. The classifier was run against this new data, and the predictions are saved into a file "sentiment_results.json" into three columns- Comments, Predicted Ratings and Actual Ratings. These are visible upon running the script, so specific results can be seen by the user.

3. The third step was to write a Chrome extension that would display the results of the sentiment analysis. The background.js file ensures the logic is executed only when the icon for the extension is clicked. Then it sends the data from "sentiment_results.json" to the content.js script. This script averages the predicted ratings, creates a popup dialog box with the average of the ratings, and displays an appropriate message according to the result.

Usage of the software

This software can be used by downloading the folder titled "project_code" from GitHub. The reviews.csv from the above Kaggle link (https://www.kaggle.com/code/gemmin/sentiment-analysis/input?select=reviews.csv) must also be downloaded and added to the same folder- I was unable to upload it to GitHub due to the size of the file.

Open your machine's Terminal, navigate into the project directory (into "project_code"). Run the following scripts:

1) python scrape_me.py <URL of a Sally's Baking Addiction recipe, eg https://sallysbakingaddiction.com/strawberry-cake/ >

2) python naive_bayes.py

3) Open Chrome and go to chrome://extensions/. Click on the "Load unpacked" button on the top left, and upload the whole folder "project_code". Ensure a new Extension card is created (titled Sentiment Analysis Extension), reload it if needed. Once it is active, click the extensions icon (puzzle piece) at the top of the browser to view the new extension. Pin it using the pin icon. A green icon should now be visible at the top of the browser. Navigate to

the same recipe link (reload it if the webpage is already open), and click the green extension icon. A popup dialog box of the average predicted sentiments of the comments of the current recipe should now be visible.

Reflection/Improvements

The original proposal for this project was a Chrome Extension for Sentiment Analysis of Restaurant Reviews. However, in my initial exploration of implementing this topic, I realized that Yelp was the most unified source to gather restaurant reviews from. Upon coming to this conclusion, I also realized that Yelp has a pretty good way to filter for reviews and their sentiment- each comment has a star rating, and you can filter by rating number, see the number of reviews for each rating number, etc. It did not make sense to add sentiment analysis to this already informative tool.

I then decided to pivot to a similar topic- sentiment analysis on recipe blogs' comments. Not all comments have a rating attached to them, and it is a widely known, notorious phenomenon of needing to scroll for a very long time to get to the bottom of a recipe blog post (even to just find the recipe itself, let alone the comments). It seemed like a better fit, and a more helpful tool, to add a sentiment analysis Chrome extension to blogs, rather than to Yelp.

One improvement this project could use is the volume of comments analyzed per recipe. Some of the more popular recipes have 30+ pages of comments. I attempted to analyze all of these, but my scraping algorithm could not handle the volume and crashed due to timeouts. I decided to stick with the latest page of comments for this project, but it would definitely provide the user with more information if all the comments could be analyzed.

Additionally, the Naive Bayes algorithm could be improved in some ways. Adding more data to train the classifier on could yield more accurate classification results. A different classifier altogether may yield better results as well. Given more time, I would definitely look into either of these options.