**Assignment 3: Building a Crowdsourced Recommendation System**

**New due date: 1st November by 11:59 p.m.**

**High level description:** The objective of this group assignment is to create the building blocks of a crowdsourced recommender system. This system should accept user inputs about desired attributes of a product and come up with 3 recommendations.

For this assignment, scrape reviews of craft beer from beeradvocate.com, mainly because the site is easy to scrape compared to many others. However, if you prefer, you may want to scrape another site for a different product – say, musiciansfriend.com for reviews of musical instruments like acoustic drums or electric guitars. The challenge is that many sites (including musiciansfriend.com provide reviews as the number of stars, which are pictures, and harder to scrape – by contrast, on beeradvocate.com, the rating is given as text, which is easier to access).

**Task A.** Use Web Scraper to extract about 3k reviews of beers from Beeradvocate.com. I would suggest choosing the following link, which shows the top 250 beers sorted by ratings:

<https://www.beeradvocate.com/beer/top-rated/>

The nice feature of the above link is that it is a single-page listing of 250 top-rated beers (avoids the pagination feature, which you need in cases where listings go on for many pages).

Scrape reviews as well as the ratings provided by users for each beer. The CSV output file should be processed to retain 3 columns: product\_name, product\_review, and user\_rating.

**Task B.** Assume that a customer, who will be using this recommendation system, has specified 3 attributes in a beer – e.g., hoppy, malty, dark. Find five of the most important attributes of beer with word frequency analysis. Choose three attributes from these five.

**Task C.** Perform a **similarity** analysis with the 3-attribute set and the reviews. See guidelines below on how to do this.

**Task D.** Perform sentiment analysis on the reviews. While it would be nice to get separate sentiment values for each attribute, it will take a significant amount of additional processing. So we will get an overall sentiment score for each row.

Note that we are NOT using the parser script before sentiment analysis. Since sentiment.py requires a single column file with the data, you have to remove the beer name and similarity score columns in your data file before performing sentiment analysis (though you should retain that information in your master data file). Once you get the compound scores for each row, you can add this information to the master data file.

Create a total score by suitably combining the similarity and sentiment scores.

Since each beer may appear multiple times in the data, get the average total score for each beer.

**Task E.** Sort the evaluation scores from high to low, and **recommend 3 beers** to the customer.

**Task F.** How would your recommendations differ if you ignored the similarity and sentiment scores, and simply chose the 3 highest rated beers from your entire dataset? To answer this question, you need to calculate the average rating for each beer mentioned (remember each beer in your data has multiple reviews, and you need to get the average rating for each beer). Would these three beers meet the requirements of the user looking for recommendations? Why or why not? Justify your answer. Use the user\_rating data and similarity scores to answer this question.

Your submission should include a word or pdf file showing

1. Names of all team members **inside** the document (only one submission per team)
2. The average total scores for the three products you recommended in task E.
3. Your analyses and answer to task F. Make sure you show the ratings, similarity scores and sentiments for the products you recommend in tasks E and F.

**Guidelines**

1. Once you get the data in the csv format, retain three columns, product\_name, product\_review and user\_rating. Save the file as an Excel file, e.g., beeradvocate\_reviews.xlsx since the python script similarity\_script.py wants the data file to be in .xlsx format.
2. Use the attributes.txt file to specify the product attributes – **one attribute per line**, no commas.
3. You will need to run similarity\_script.py for the similarity analysis. This script compares the words in the attributes.txt file with the reviews, and creates a similarity score between 0 and 1 for each review. Check the comments in the script for things to do before running it.

Also before you run the script, you need to do the following to the data file. Make sure the review column in your data file is called Restaurant\_review. Also delete the user\_rating column for running this script (and of course, save the file with a different name so that the user\_rating column is not lost).

1. Now run similarity\_script.py. The output file is similarity\_output.xlsx. First, add the user\_rating column back to this file. Since you have not sorted anything (yet), copy-paste of the user\_rating column should work well.