**Analytics for Unstructured Data (F2023) Assignment 2**

**Building a Crowdsourced Recommender System**

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

Obtain reviews of craft beers from beeradvocate.com. I would suggest using the following link, which shows the top 250 beers sorted by ratings:

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

**Task A.** Extract about 5-6k reviews.

**Task B.** Assume that a customer, who will be using this recommender system, has specified 3 attributes in a product. E.g., one website describes multiple attributes of beer (but you should choose attributes from the actual data)

<https://www.dummies.com/food-drink/drinks/beer/beer-for-dummies-cheat-sheet/>

* **Aggressive (**Boldly assertive aroma and/or taste)
* **Balanced:** Malt and hops in similar proportions; equal representation of malt sweetness and hop bitterness in the flavor — especially at the finish
* **Complex:** Multidimensional; many flavors and sensations on the palate
* **Crisp:** Highly carbonated; effervescent
* **Fruity:** Flavors reminiscent of various fruits **or Hoppy:** Herbal, earthy, spicy, or citric aromas and flavors of hops o**r Malty:** Grainy, caramel-like; can be sweet or dry
* **Robust:** Rich and full-bodied

Use the above attributes as examples only, for a word frequency analysis of beer reviews is a better way to find important attributes in the actual data.

**Assume that a customer has specified 3 attributes of the product as being important to him or her.**

**Task C.** Perform a **similarity** analysis using cosine similarity (without word embeddings) with the 3 attributes specified by the customer and the reviews.

The similarity script should accept as input a file with the product attributes, and calculate similarity scores (between 0 and 1) between these attributes and each review. That is, the output file should have 3 columns – product\_name (for each product, the product\_name will repeat as many times as there are reviews of the product), product\_review and similarity\_score.

**Task D.** For every review, perform a sentiment analysis (using VADER or any LLM). In case you have to change the default values of words in the VADER lexicon, use this article: <https://medium.com/swlh/adding-context-to-unsupervised-sentiment-analysis-7b6693d2c9f8>

**Task E.** Create an evaluation score for each beer that uses both similarity and sentiment scores.

Now **recommend 3 products** to the customer.

**Task F.** How would your recommendation change if you use word vectors (e.g., the spaCy package with medium sized pretrained word vectors) instead of the plain vanilla bag-of-words cosine similarity? One way to analyze the difference would be to consider the % of reviews that mention a preferred attribute. E.g., if you recommend a product, what % of its reviews mention an attribute specified by the customer? Do you see any difference across bag-of-words and word vector approaches? Explain.

Note that the article doesn’t claim that bag-of-words will always be better than word embeddings for recommender systems. It lays out conditions under which it is likely to be the case. That is, depending on the attributes you use, you may or may not see the same effect.

**Task G.** How would your recommendations differ if you ignored the similarity and feature sentiment scores and simply chose the 3 highest rated products from your entire dataset? Would these products meet the requirements of the user looking for recommendations? Why or why not? Justify your answer with analysis. Use the similarity and sentiment scores as well as overall ratings to answer this question.

**Task H.** Using the top four attributes of beer (from word frequency analysis), calculate the lifts between these attributes and any 10 beers in your data. Choose one beer, and find the most similar beer (among the remaining 9) using the lift values. Explain your method.