DATA301 Project Final Report

Student Name: Moses Bernard Velano
Student ID: 83396373

Abstract / Summary

Building on from the Project Proposal, I proceeded on using the same dataset which was BeerAdvocate dataset. The research question that I have stated in my Proposal was looking at "what is the relationship between a customer's text reviews and a beer's features to the overall impression of the most popular beer". The algorithms that mentioned in the Proposal was using the Term Frequency - Inverse Document Frequency (TF-IDF) and Cosine Similarity. The intended result of this project was specifically determining the best beer which was very dependent on the customer queries on text reviews, ratings on the aspects of the beer, and their overall impression. The TF-IDF algorithm was used to extract and determine the importance of a term word within a collection of documents in the BeerAdvocate dataset. That TF-IDF algorithm should show three categories (negative, neutral, and positive) from overall impressions and associating this with the term words from a customer's text review towards a specific beer. The Cosine Similarity algorithm was then used after in order to find the similarity of beer's aspects/features (appearance, aroma, palate, and taste) and comparing it to another beer from a customer's review. The significance of the result is it can help beer drinkers to easily choose the most liked beer by recommendation, promotion, and displaying the text reviews for customers to see.

Introduction

Background

The BeerAdvocate data set provides various characteristics that determine the attributes of a beer. An example of this is, the Average By Volume (ABV) which refers to the percentage of alcohol given a volume of a beer. We also get the name of the beer (name), unique beer ID (beerId), where it was made (brewerId), classification of a beer (style). The data set also provides individual score ratings on the beer such as appearance, aroma, palate, taste, and the overall impression. It also informs us about the specific point in time when a customer user submitted a review of a particular beer (time), the reviewer's user profile ID name (profileName), and personal comment reviews (text).

We can use TF-IDF in our research to help us find each term frequency of words from the customer text reviews and relate this back to the overall categorized impression of a beer. This is achievable because we know that TF-IDF algorithm weighs the significance of a specific term frequency word of single reviews from a customer user by looking at how frequent a word appears in that review (Term Frequency). We then evaluate how frequent a term word appears across all user reviews which is all document in the collection (Inverse Document Frequency). Then we just multiply TF * IDF. The cosine-similarity algorithm is necessary in our research to help us understand the relationship of a user text reviews to the associated score ratings they gave on a beer's features (appearance, aroma, palate, and taste).

Motivation

This research question is relevant to me because I do not like drinking any type of beers and someday I would like to try a beer that best fits my desire. As a student at University of Canterbury we know that there is a high volume of students who drink beers. Therefore, I want to look at the most popular beer in the BeerAdvocate data set and understand the pattern of a customer's text reviews relating to the overall impression of a beer and the score ratings of a beer based of its features. We can connect the findings from our algorithms to the factors that motivate UC students into purchasing beers. This research can help beer drinkers find the best beers that satisfy their preferences when it comes to beers. It can also help others into avoiding a very bad tasting beers, in that way buyers do not waste any money and undrunk beers. Otherwise, other UC students who do not like drinking beer and are wanting to try it in future, we can use the result from this research to help find the best beer that would certainly satisfy their preferences.

Research Question or Hypothesis

The research question is "what is the relationship between a customer's text reviews and a beer's features to the overall impression of the most popular beer.". This is relevant to our data set because this can improve accuracy and effectiveness of a beer company, a liquor store, or for frequent beer buyers when trying to get the most out of a beer. By using TF-IDF this will help identify the most important term words in each user text reviews. Which will be categorized in three categories of negative, neutral, and positive coming from the score ratings of overall impression a user gave. While cosine similarity algorithm measures the similarity of a user's score reviews on a beer's features between another user's review on a beer's features. With both algorithms, this allows future people to benefit from the given information as they will likely have a faster, easier and improve their decision-making when selecting a beer that best match their preferences. Additionally, this can help promote the best beer in the market as it includes user text reviews that explains the aspects of the beer.

Experimental Design and Methods

Before applying these algorithms, I first needed to import the BeerAdvocate dataset into Google Collab. Then I implemented specific code that decompressed the file that have been compressed using the "gzip" compression into a normal JSON file format for achievable latter processes. The following steps include:

setting up local Spark Cluster → load BeerAdvocate dataset into an RDD → PySpark Sampling (this is where I took a sample fraction from the original dataset) → importing 'stopwords' which is useful for the TF-IDF algorithm → verify attributes of the dataset to ensure accuracy → cleaning and filtering the desired dataset stored in JSON-line.

Finally, I then performed TF-IDF algorithm which started off with computing the Term Frequency (TF) for each term words in a user's beer review. The expected output for TF should consider how frequently a word appears from a singular review in the document (where individual user review = a collection of documents). The output that I was expecting met my expectations, thus the result showed something like this:

[(('term word', 'review/overall'), Term Frequency value), ……]

[(('big', 4.5), 0.1799506984387839), (('thanks', 4.5), 0.0780608052588332), (('n', 4.5), 0.0036976170912078883), ……]

Afterwards, I proceeded to take the Inverse Document Frequency (IDF) for each term word in all documents (all the user's beer reviews). This is basically trying to find out whether a specific word used in a beer review would also appear in another user beer review. The output that I was expecting met my expectations, thus the result showed something like this:

[('term word', Inverse Document Frequency value), ……]

[('big', 8.800694349462107), ('thanks', 9.655473215017361), ……]

Finally, I computed TF * IDF score for each term word in BeerAdvocate dataset and collect/display the overall impression score ratings that a user gave scaling from 1-20. The output that I was expecting met my expectations, thus the result showed something like this:

('negative 5.0', [(6.592480285889554, 'beer'), ……
('negative 4.5', [(6.592480285889554, 'beer'), ……
.
.
.
('negative 1.0', [(6.592480285889554, 'beer'), …….

The following explanation was my initial plan from the Project Proposal:
The Cosine-similarity algorithm will support the reasoning for finding the most popular beer in the BeerAdvocate dataset. But before we use this algorithm, we would need to extract the attributes of a beer's appearance, aroma, palate, and taste and put it as a list of integers. Next, we perform the cosine-similarity algorithm to calculate the cosine similarity between two list of integers (this is the user's score reviews on the characteristic of a beer). Finally, we can connect the Cosine-similarity and TF-IDF algorithms together. Finally, we hope to retrieve a result of the most frequent words used from a user's text reviews and the cosine similarity of users on a beer's characteristic which hopefully would lead us to finding the overall best beer.

But during the implementation of Cosine Similarity, after receiving the grade/comment from the Project Proposal, and few advices from my friends and tutor I realised that finding the 'best beer' was not possible as the result from the TF-IDF were only showing maximum of '5 review/overall' rating impression from user reviews.

Similar but, different approach:
Though, I still continued to extract the attributes of a beer's appearance, aroma, palate, and taste and put it as a list of integers. Then took the mean average of the attributes from all documents (user reviews) that had the same 'review/overall' impression ratings. Afterwards, I performed cosine-similarity algorithm to calculate the cosine similarity from a specific 'review/overall' rating impression (which contains the mean average of all attribute rates) to

another specific ‘review/overall’ rating impression. Hence, this was the output that I was wanted to proceed with:

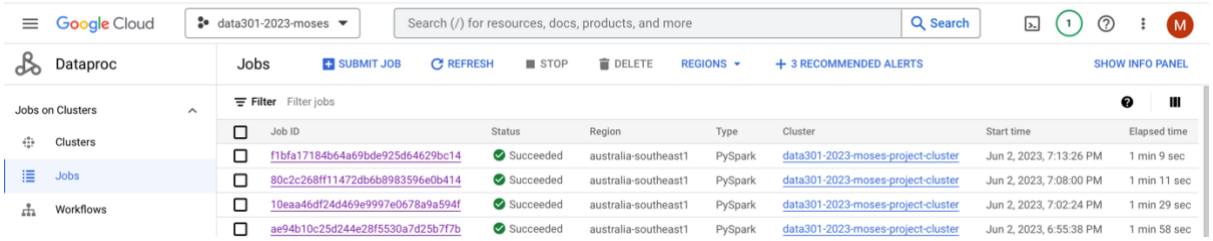
```
Cosine similarity between IDs 1.0 and 1.0: 1.0
Cosine similarity between IDs 1.5 and 1.0: 0.9843091327750998
Cosine similarity between IDs 2.0 and 1.0: 0.9819805060619657
Cosine similarity between IDs 2.5 and 1.0: 0.9983374884595828
Cosine similarity between IDs 3.0 and 1.0: 0.9819805060619657
```

Further explanation of how of this will be explained in the Result and Conclusion part of the report.

Specific code or Libraries created or used to implement the methods	Brief Explanation
`sampled_rdd = user_reviews_df_ids.sample(False, 0.02, seed = 42)`	This is one way to get random sample records of the original dataset of our BeerAdvocate which contains about 589528 documents.
`def overall_splitter(ratings):`	Splits the overall review ratings made by a user into a three categories. That is negative (score from 0-6), neutral (score from 7-13), and positive (score from 14-20). This is important as it will be used in the result of TF-IDF algorithm.

`import statistics`
`import math`
 	
`def mean_value_computation(rates):`
	Computing all the mean average value of the Beer aspects made by user queries. Gets the appearance, aroma, palate, taste and then average all of it per 'review/overall'.
	
	This will be used later in the cosine similarity, wherein comparison between the first lowest 'review/overall' to the first top 5 lowest 'review/overall'.

Results



For P=1 BeerAdvocate Dataset with 1 core

Spark jobs take ~60 seconds to initialize resources. DISMISS

```
{4.5: 2434, 4.0: 4343, 3.5: 2161, 5.0: 714, 1.0: 92, 2.5: 332, 1.5: 107, 2.0: 232, 3.0: 1035}
[(('big', 4.5), 0.1799506984387839), (('thanks', 4.5), 0.0780608052588332), (('n', 4.5), 0.0036976170912078883)]
[(1.0, (3, 2, 2, 2)), (1.5, (3, 3, 2, 2)), (2.0, (3, 3, 3, 3)), (2.5, (4, 3, 3, 3)), (3.0, (4, 4, 4, 4)), (3.5, (4, 4, 4, 4)), (4.0, (4, 4, 4, 4)), (4.5, (5, 5
[((1.0, 1.0), 1.0), ((1.5, 1.0), 0.9843091327750998), ((2.0, 1.0), 0.9819805060619657), ((2.5, 1.0), 0.9983374884595828), ((3.0, 1.0), 0.9819805060619657)]
elapsed time is 71.2781274318695
23/06/02 06:57:31 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@1acd00e4{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

Output is complete

For P=4 BeerAdvocate Dataset with 4 cores

Spark jobs take ~60 seconds to initialize resources. DISMISS

```
{4.5: 2434, 4.0: 4343, 3.5: 2161, 5.0: 714, 1.0: 92, 2.5: 332, 1.5: 107, 2.0: 232, 3.0: 1035}
[(('big', 4.5), 0.1799506984387839), (('thanks', 4.5), 0.0780608052588332), (('n', 4.5), 0.0036976170912078883)]
[(1.0, (3, 2, 2, 2)), (1.5, (3, 3, 2, 2)), (2.0, (3, 3, 3, 3)), (2.5, (4, 3, 3, 3)), (3.0, (4, 4, 4, 4)), (3.5, (4, 4, 4, 4)), (4.0, (4, 4, 4, 4)), (4.5, (5,
[((1.0, 1.0), 1.0), ((1.5, 1.0), 0.9843091327750998), ((2.0, 1.0), 0.9819805060619657), ((2.5, 1.0), 0.9983374884595828), ((3.0, 1.0), 0.9819805060619657)]
elapsed time is 59.14844512939453
23/06/02 07:03:51 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@6b4893af{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

Output is complete

For P=8 BeerAdvocate Dataset with 8 cores

Spark jobs take ~60 seconds to initialize resources. DISMISS

```
[(('big', 4.5), 0.1799506984387839), (('thanks', 4.5), 0.0780608052588332), (('n', 4.5), 0.0036976170912078883)]
[(1.0, (3, 2, 2, 2)), (1.5, (3, 3, 2, 2)), (2.0, (3, 3, 3, 3)), (2.5, (4, 3, 3, 3)), (3.0, (4, 4, 4, 4)), (3.5, (4, 4, 4, 4)), (4.0, (4, 4, 4, 4)), (4.5, (5, 5
[((1.0, 1.0), 1.0), ((1.5, 1.0), 0.9843091327750998), ((2.0, 1.0), 0.9819805060619657), ((2.5, 1.0), 0.9983374884595828), ((3.0, 1.0), 0.9819805060619657)]
elapsed time is 42.140639305114746
23/06/02 07:09:08 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@5cfc902d{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

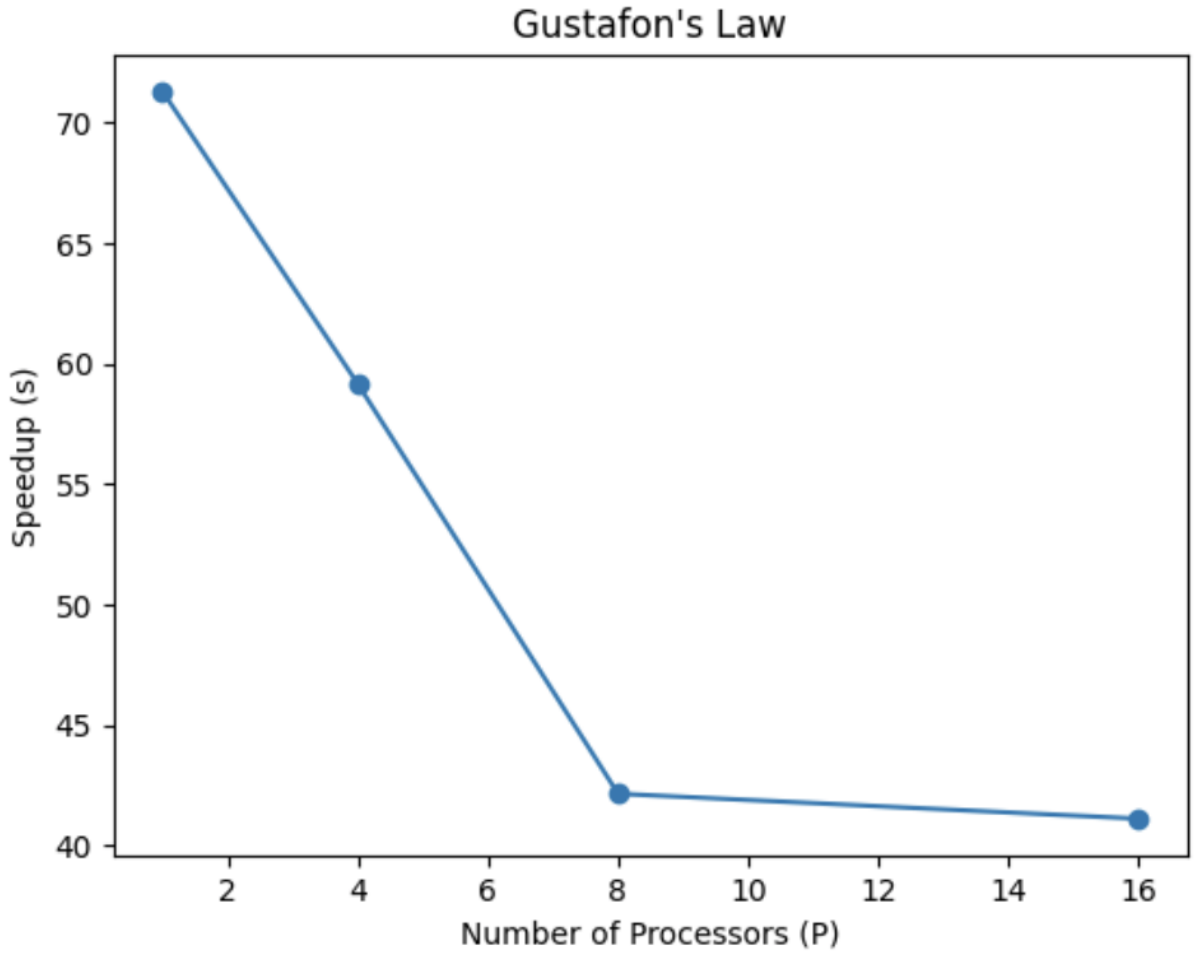
Output is complete

For P=16 BeerAdvocate Dataset with 16 cores

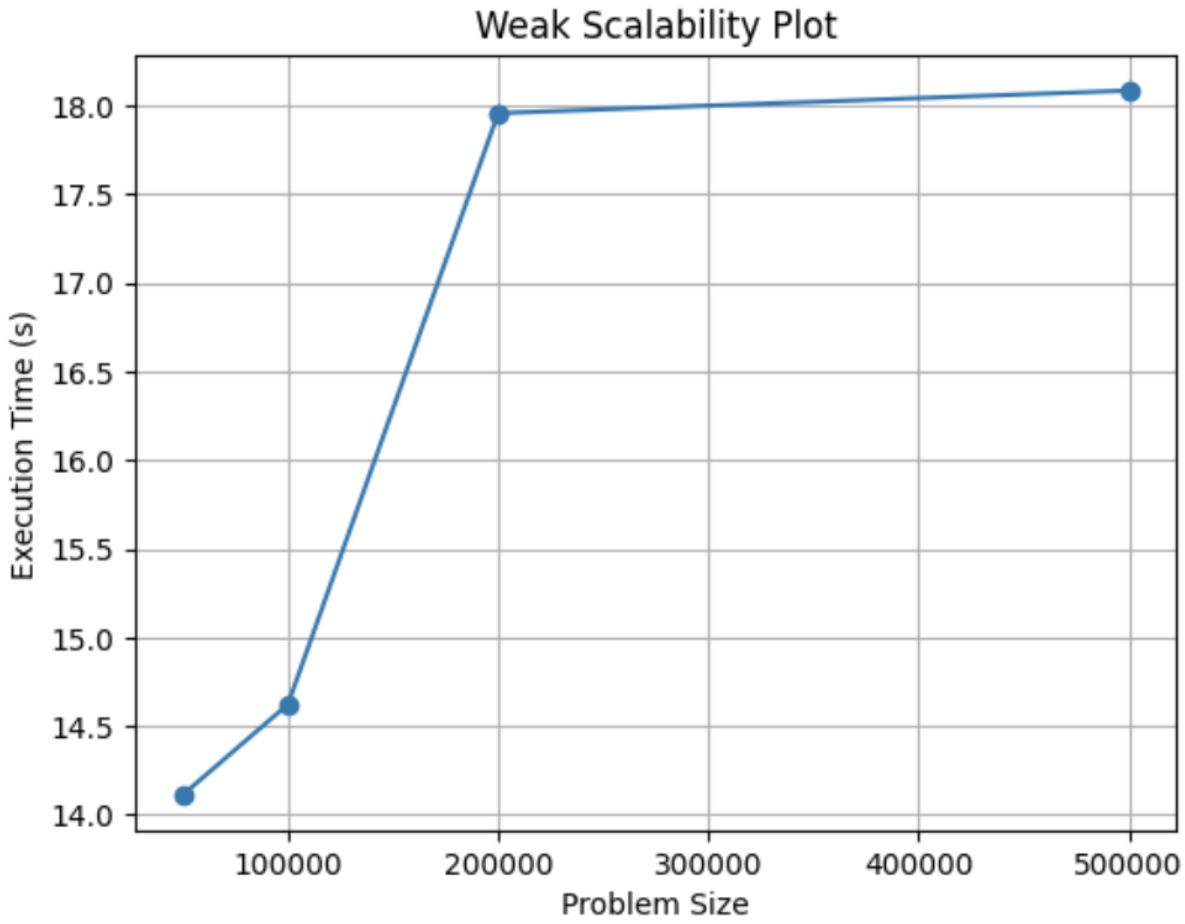
Spark jobs take ~60 seconds to initialize resources. DISMISS

```
{4.5: 2434, 4.0: 4343, 3.5: 2161, 5.0: 714, 1.0: 92, 2.5: 332, 1.5: 107, 2.0: 232, 3.0: 1035}
[(('big', 4.5), 0.1799506984387839), (('thanks', 4.5), 0.0780608052588332), (('n', 4.5), 0.0036976170912078883)]
[(1.0, (3, 2, 2, 2)), (1.5, (3, 3, 2, 2)), (2.0, (3, 3, 3, 3)), (2.5, (4, 3, 3, 3)), (3.0, (4, 4, 4, 4)), (3.5, (4, 4, 4, 4)), (4.0, (4, 4, 4, 4)), (4.5, (5, 5
[((1.0, 1.0), 1.0), ((1.5, 1.0), 0.9843091327750998), ((2.0, 1.0), 0.9819805060619657), ((2.5, 1.0), 0.9983374884595828), ((3.0, 1.0), 0.9819805060619657)]
elapsed time is 41.10582995414734
23/06/02 07:14:32 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@24f5ea3d{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

Output is complete



Gustafson’s Law focuses on the potential benefits of scaling up the problem size and effectively use the parallel processing power. Therefore as the amount of a file text increases (parallelizable work) the amount of processor cores at each step can be effectively applied along with the increase size of the file. In our example, as we increase the size of the file, we were able to effectively use the available processor cores in order to benefit from speedup.



Weak scalability refers to the ability of a system or algorithm to efficiently handle larger problem sizes as the number of processors or computing resources increases. Therefore, as I added more processors to the system, the size or complexity of the problem being solved also increases proportionally. In a system with weak scalability, the computational load is distributed among multiple processors, and each processor works on a separate portion of the problem. The objective is to ensure that each processor handles a consistent workload as the system grows. By achieving this balance, the system can effectively handle increasingly larger problems without significant delays. Though, the time increased a little as I increased the size of the file, we can conclude that it was still able to complete in a reasonable amount of time. By attaining weak scalability, we can leverage the capabilities of multiple processors to address a larger problems while maintaining efficiency and avoiding notable declines in performance.

In the hypothesis I was not able to answer my initial question due to unforeseen circumstances. But I was able to overcome this by reversing my research question to “what is the relationship between a customer’s text reviews and a beer’s features to the overall impression of the least popular beer”. As explained earlier, Cosine Similarity was used in order to compare all the mean averaged attributes according to a specific ‘review/overall’. The output that I wanted looked like this:
[((1.0, 1.0), 1.0), ((1.5, 1.0), 0.9843091327750998), ((2.0, 1.0), 0.9819805060619657), ……]

Equivalently:

[(('review/overall', 'review/overall'), 'Cosine Similarity Value from the mean average value of beer's attributes')]

After, that I took the smallest value of 'review/overall' and compare using Cosine Similarity to the least 5 'review/overall'. The reason for choosing the least/smallest value of 'review/overall' is because most of the output after computing the TF-IDF was only showing 'review/overall' value up to 5 the maximum. Thus, it did not make sense to proceed with my initial research question because there was barely 'review/overall' value that exceeded 10 or 15. Hence, after getting some advice from fellow students and tutor, I decided to look at the least 'review/overall' value and compare that to the other least top 5 review/overall' values. In this way, it can better answer to my new research question "what is the relationship between a customer's text reviews and a beer's features to the overall impression of the least popular beer". With the little change I made to my methodologies, I can then simply answer the research question. Using the TF-IDF still serves the same purpose. In this way we can see each term words made by users to a beer and associate that to why they gave a low 'review/overall' rating. After that, we can use cosine similarity to get the least favourite beer (the lowest 'review/overall' rating) and use Cosine Similarity to compare it to the other least top 5 'review/overall' values. We can look at how similar those attribute rates (appearance, aroma, palate, and taste) made by users from the very least 'review/overall' compared to the least top 5 'review/overall' values. Hence, associate the result of TF-IDF by identifying the most important term words in each user text review that are categorized by 'review/overall' rating into negative, neutral, positive and Cosine Similarity use the mean averaged value of beer's attributes according to the 'review/overall'. Overall, we can look back and forth to why a user gave those text review and attribute ratings on a specific beer according to the 'review/overall' value. This is a good way for future buyers to avoid any bad tasting beers by looking at the review texts and the mean average values.

Conclusion

No, because after doing the TF-IDF algorithm, I was not expecting for the 'review/overall' to be mostly below a value of 5. It was hard to procced with my initial research question because none of the user (documents) rated the overall impression of a beer more than 10 or 15. Therefore, it was not necessary to look the 'most popular beer'. Though, when I reversed the question into looking at the 'least popular beer' I was able to find ways that could answer this question while keeping all the algorithms useful.

This approach allows future buyers to gain a better understanding of potential issues or drawbacks with specific beers. By considering the review texts and mean average attribute ratings, potential customers can make informed decisions and potentially avoid purchasing beers that are likely to have a negative taste or overall experience. Furthermore, TF-IDF and cosine similarity techniques provides a systematic way to analyse and compare the relationships between customer reviews, beer features, and overall impressions. This methodology can be extended to further explore the connections between textual feedback, attribute ratings, and overall ratings in different beer categories or across various products. Overall, the implications of my results suggest that by utilizing the TF-IDF approach and comparing the least popular beers, it is possible to gain valuable insights into the reasons behind low ratings, identify important terms associated with negative impressions, and provide future buyers with more informed decision-making tools in selecting beers based on customer reviews and mean average attribute ratings.

I could have a further analysis on least popular beer, by delving deeper into the characteristics and attributes of the least popular beers and analysing additional factors such as pricing, geographical distribution, or brewery information. This could provide insights into specific aspects that contribute to their low overall impressions. Sentiment Analysis would also be considered by incorporating sentiment analysis techniques it determines the sentiment (negative, neutral, or positive) expressed in the text reviews. This would provide a more detailed understanding of the emotional aspect of customer feedback and its relationship to overall ratings. Lastly, integration of external data sources, such as social media or demographic information, to enrich the analysis. This could provide additional context and insights into consumer behaviours and preferences, enabling a more comprehensive understanding of the relationship between text reviews, attributes, and overall impressions. By incorporating these future questions or directions, I can further enhance the depth and breadth of my project, expanding its applicability and contributing to the understanding of customer perceptions and preferences in the beer industry.

Critique of Design and Project

One of the methods in the software implementation that I should have worked on better was being able to know that I should have took a sample before doing any algorithms. Because during week 10, as I was doing the IDF part of the TF-IDF algorithm, I was not able to make it work properly. And after considering help and advices from my friends, lecturer, and tutor, they have suggested to take a small fraction of documents from the original document size. I should have seek advices earlier at most instead of being stuck for 5 days figuring out what is wrong. Although, after that by taking a fraction of documents from the original dataset, the IDF have worked completely, hence I'm assuming that the dataset was way too large for it to handle.

Also, based on the provided information, one part of the design or method that could potentially benefit from a different approach is the cleaning and filtering of the dataset. While the proposal mentions cleaning and filtering the desired dataset, it does not provide specific details about the criteria or steps involved in this process. The lack of clarity regarding the cleaning and filtering methods can lead to potential issues or limitations in the analysis. In order for me to address this, it would have been better to outline specific cleaning and filtering techniques that could enhance the quality and relevance of the dataset. For example, considering the BeerAdvocate dataset, it might be beneficial to remove duplicates, handle missing values, eliminate irrelevant columns, and apply text pre-processing techniques such as removing stop words or performing stemming. Clearly defining and implementing these cleaning and filtering steps would improve the accuracy and reliability of the subsequent TF-IDF and cosine similarity algorithms. Additionally, providing a rationale for the chosen cleaning and filtering techniques would help justify their inclusion in the process. Explaining why certain attributes or instances are deemed irrelevant or how specific pre-processing techniques improve the analysis would add depth to the methodology. Furthermore, documenting any challenges or limitations encountered during the cleaning and filtering process would provide valuable insights for future researchers. Overall, by incorporating a more detailed and explicit approach to cleaning and filtering the dataset, my research project would have a stronger foundation for accurate analysis and more reliable results.

Reflection

- DATA301 Lab 2
- DATA301 Lab 4
- DATA301 Lab 3 Part 2
- DATA301 Lab 4 Part 2
- DATA301 Sample Project Code
- Visual Help for Spark Transformations and Actions

Through this project researcher, I learned about the application of TF-IDF and Cosine Similarity in text analysis, the importance of data cleaning and filtering, working with JSON-lines (uncompressing a compressed file), and the potential for extracting meaningful insights from user reviews to make informed decisions. The project also demonstrated the potential for integrating different algorithms to obtain more comprehensive results and highlighted the relevance of considering user preferences and feedback in product evaluation.

References

- Provide any citations and/or links to notebooks, datasets, etc
- Provide a list of student names that you have worked directly with on the project.
- You do not need to list code you have gotten from the class project forum on learn whether it was posted by an instructor or another student.

References

Apache Spark. (n.d.). JSON Files. https://spark.apache.org/docs/latest/sql-data-sources-json.html

Geeks for Geeks. (n.d.). Removing stop words with NLTK in Python. https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

Laerd Dissertation. (2012). How to structure quantitative research questions. https://dissertation.laerd.com/how-to-structure-quantitative-research-questions.php

McAuley, J., Leskovec, J., Jurafsky, D. (2012). International Conference on Data Mining (ICDM). https://cseweb.ucsd.edu/~jmcauley/datasets.html#multi_aspect

Python. (2023). re — Regular expression operations. https://docs.python.org/3/library/re.html

Singh, A. (2015). CSE 225: *Assignment 1 Winter 2015*. Retrieved from https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/wi15/Alok_Singh.pdf

Spark By Examples. (2023). Average of List in Python (7 Examples). https://sparkbyexamples.com/python/average-of-list-in-python/

Spark By Examples. (2023). PySpark Random Sample with Example. https://sparkbyexamples.com/pyspark/pyspark-sampling-example/#:~:text=PySpark%20sampling%20%28pyspark.sql.DataFrame.sample%20%

28%29%29%20is%20a%20mechanism%20to,is%20the%20syntax%20of%20the%20sample%20%28%29%20function.

I also got my ideas from the Labs, Lectures, and the sample project code.

My mates Sudarshan Malla and Daniel Nashad in implementing/loading the BeerAdvocate dataset. I also got help and advice with my TF-IDF, Google Cloud, and asked for opinions for my Cosine Similarity.