CS513 Phase-II Report: Wine Reviews

Team18
Yunfei Ouyang   Sicheng Jiang
yunfeio2@illinois.edu   sj62@illinois.edu
July 30, 2023

Abstract

This project focuses on the analysis of wine reviews using a combination of OpenRefine and Python for data cleaning and preparation. The dataset consists of wine reviews from various sources and requires thorough cleaning to ensure data quality and suitability for analysis. OpenRefine is utilized to perform initial data cleaning tasks such as removing duplicates, clustering entries, and handling missing values. Subsequently, Python is employed to further preprocess the data, including re-indexing, numeric normalizations. The cleaned and preprocessed dataset is then ready for various use cases. This study highlights the importance of proper data cleaning and processing to extract meaningful insights from large and unstructured wine review datasets.

0. Dataset and Use Cases

0.1 Dataset

The dataset is Winery-Kaggle, it contains 130k wine reviews with variety, location, winery, price, and description. Source: https://www.kaggle.com/datasets/zynicide/wine-reviews

0.2 Dataset Structure

The "wine_reviews" dataset contains following columns:

1. id: Unique identifier for each wine review (integer primary key).
2. country: Country of origin for the wine (string).
3. description: Brief textual description of the wine by a sommelier (string).
4. designation: Specific vineyard within the winery where grapes are grown (string).
5. points: Wine rating on a scale of 1 to 100 (integer). Only reviews with scores of 80 or higher are included.
6. price: Cost of a bottle of wine (integer).
7. province: Province or state from which the wine originates (string).
8. region_1: Wine growing area within a province or state (string).
9. region_2: More specific sub-region within a wine growing area (string, can be blank).
10. taster_name: Name of the sommelier or wine taster who provided the review (string).
11. taster_twitter_handle: Twitter handle of the sommelier or wine taster (string).
12. title: Title of the wine review (string).

13. variety: Type of grapes used in making the wine (string).
14. winery: Name of the winery that produced the wine (string).

0.3 Use Cases

a. "Zero Cleaning" U0: Data Cleaning is not necessary

U0: We want to investigate country <country> that contains the highest rating <points> wines, and report all countries.

Data cleaning is not necessary for U0 because it is based on the <points> column and the <country> column, where <country> has no missing data. We can simply ignore any missing values in the <points> column since we are only concerned with identifying the maximum rating among all <points>.

b. "Main" U1: Data Cleaning is necessary and sufficient

U1: We aim to analyze the reviews of the taster <taster_name, taster_twitter_handle> with the most contributions (i.e., the one who has written the most reviews) in the US <country>. We want to determine the number of reviews, average rating <points>, and the minimum and maximum prices <price> for each wine variety <variety> in the US that the taster has reviewed.

U1 involves analyzing multiple metrics for each wine variety in the US from the taster with the most significant contributions. It provides a comprehensive view of taster influence and wine variations. Data cleaning is essential for this use case to ensure the accuracy and reliability of the analysis. Without proper data cleaning, the presence of missing or inaccurate data could result in biased outcomes and unreliable conclusions. For instance, incomplete or inconsistent entries in the points, price, or taster-related columns could distort the average points, price ranges, and the number of reviews attributed to a specific taster, potentially leading to misleading insights. By performing data cleaning, we can mitigate these issues and ensure that the analysis provides meaningful results for understanding wine reviews.

c. "Never Enough" U2: Data Cleaning is not sufficient

U2: We want to investigate popular wines, and since our user enjoys French wines, we would like to recommend some French wines from the Deutz winery to them.

At first glance, it may seem possible to recommend some wines to the user. However, we realize that relying solely on the reviews of wines may not provide robust enough recommendations. To make a more informed recommendation, we need to investigate additional factors such as the wine's sales history and the duration it has been on the market, as well as its performance compared to other wines. Unfortunately, these crucial factors are currently lacking in our data, and as a result, data cleaning alone will not be sufficient for U2.

1. Data Cleaning Methods

We utilized OpenRefine and Python for the data cleaning process. In the following steps, we will describe the rationale behind each cleaning operation and its relevance to U1.

1.1 Data cleaning with OpenRefine

1. **Remove Null <country>:** We removed rows with missing values in the "country" column as it is essential for our use case (U1) to analyze wines from US. Without a country value, we would not be able to link wines to specific country.
2. **Remove Null <designation>:** The "designation" column contains information about the specific vineyard where the grapes were grown. We removed rows with missing values in this column so that remaining data is informative.
3. **Remove Null <price>:** For U1, we want to analyze the prices of wines, and the "price" column is crucial for this purpose. By removing rows with missing prices, we ensure that our analysis is not biased.
4. **Replace Null <region_1> with <province> value:** Since province have no missing data, and "region_1" 16% null data, we filled empty "region_1" values with the corresponding "province" values to ensure data completness.
5. **Merge <region_1> and <region_2> to new column <region>**: After fill in empty cells of "region_1" with "province" value, we combined the "region_1" and "region_2" columns into a new column called "region" using a comma separator. This process allowed us to mitigate the issue of 61% null values in "region_2," improving data quality.
6. **Remove ending ',' for <region> column**: As part of data cleaning, we removed trailing commas from the "region" column for merging with empty "region_2".
7. **Remove column <region_1>**: We removed the "region_1" column after merging it with "region" in step 5 to simplify the dataset and remove redundant information.
8. **Remove column <region_2>**: After merging "region_1" and "region_2" in step 5, we removed the "region_2" column to simplify the dataset and remove redundant information.
9. **Remove Null <taster_name>**: For U1, we want to attribute reviews to specific tasters, so we removed rows with missing values in the "taster_name" column.
10. **Replace Null <taster_twitter_handle>**: Since there is a significant amount of missing data (24%) for the "taster_twitter_handle," removing these entries would result in a substantial loss of data. To ensure a consistent dataset for U1, we decided to replace the missing values with the magazine's Twitter handle, "@WineEnthusiast," considering that our data is scraped from the WineEnthusiast website.
11. **Fingerprint Cluster <designation>**: We used fingerprint clustering to merged 993 similar "designation" clusters. This step helps reduce data redundancy.
12. **N-gram=2 Cluster <designation>**: We further improved "designation" column by performing n-gram clustering, manually excluded 19 clusters, and merged 182 out of 201 clusters. This step enhances data consolidation for U1 analysis.

13. **Fingerprint Cluster <region>**: We applied fingerprint clustering to the "region" column, merged 7 clusters. This step allows us to group similar regions together, facilitating region-based analysis.
14. **Fingerprint Cluster <title>**: We used fingerprint clustering to merge similar "title" values, merged 80 clusters. This step simplifies analysis on wine titles.
15. **N-gram=2 Cluster <title>**: We further performed n-gram clustering on the "title" column, merged 9 clusters.
16. **Fingerprint Cluster <winery>**: We applied fingerprint clustering to the "winery" column, merged 20 clusters. This step allows better winery-based analysis.
17. **N-gram=2 Cluster <winery>**: We further improved winery clustering by performing n-gram clustering, merged 15 out of 16 clusters.
18. **<id> to Number:** We converted "id" column to numeric data.
19. **<points> to Number**: We convered "points" column to numeric data.
20. **<price> to Number**: We convered "price" column to numeric data.
21. -40. **Trim Leading and Trailing & Collapsing Consecutives Whitespaces**: We cleaned textual data in all non-numeric columns, removing leading and trailing whitespaces and collapsing consecutive whitespaces. This step ensures data uniformity and consistency throughout the dataset, improving data quality for analysis.

By performing these data cleaning steps, we ensure the accuracy and reliability of the dataset, making it suitable for our use case (U1) focused on analyzing tasters' reviews of wines by country, points, price, and other relevant attributes. The cleaning process addresses missing data, enhances data integrity, and reduces data redundancy, enabling meaningful insights for wine enthusiasts.

1.2 Data Cleaning with Python

After processing the data with OpenRefine, we use python to normalize the points and reset the index of the data.

1. **Normalize <points>**: Since the data only includes wines with ratings from 80 to 100, we decided to normalize the scores to a range of 0 to 10 to provide a more user-friendly representation. This normalization allows for easier comparisons and interpretations of wine ratings.
2. **Reset index for <id>**: After completing the data cleaning process, we found it essential to reset the index of the "id" column to improve data organization and enhance data readability. By resetting the index to a sequential numbering system (1, 2, 3, ...), we achieve a more structured dataset, making it easier to access specific rows and facilitating further data analysis.

Python data cleaning involves normalizing numeric data and re-indexing rows. While not essential for our use case U1, it remains valuable. Normalization transforms data into an intuitive range for analysis and visualization. Re-indexing facilitates row-based data indexing, enabling support for various use cases.

1.3 IC violations with Python

We have also defined the following IC violation Checkers for our data using Python

1. **IC_empty_country**: Verifies if there are empty fields in the <country> column.
2. **IC_empty_points**: Verifies if there are empty fields in the <points> column.
3. **IC_empty_price**: Verifies if there are empty fields in the <price> column.
4. **IC_empty_taster_name**: Verifies if there are empty fields in the <taster_name> column.
5. **IC_empty_taster_twitter_handle**: Verifies if there are empty fields in the <taster_twitter_handle> column.
6. **IC_empty_variety**: Verifies if there are empty fields in the <variety> column.
7. **IC_is_numeric_points**: Verifies that all entries in the <points> column are numeric.
8. **IC_is_numeric_price**: Checks that all values in the <price> column are numeric.
9. **IC_points_range_points**: Ensures that all values in the <points> column fall within the range of 80 to 100 before the points normalization.
10. **IC_points_range_price**: Checks all values in the <price> column are greater than zero.

These data integrity checks are crucial for U1 and in general, as they ensure the accuracy and reliability of the dataset used for analyzing wine reviews by country, points, prices, and other relevant attributes. Empty or non-numeric fields could lead to biased results and hinder meaningful insights, making these checks essential for creating a consistent and reliable dataset.

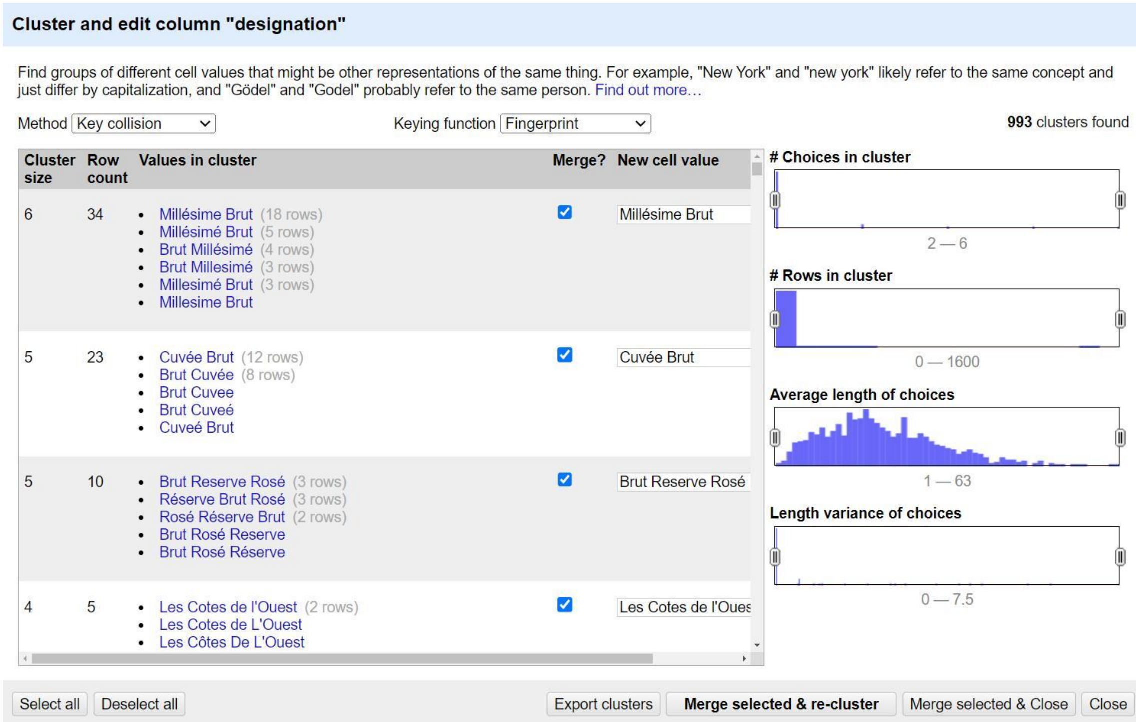
2. Document data quality changes

We provide a summary of data changes from OpenRefine and Python below. We also showcase the data quality improvement using the IC-violation afterward.

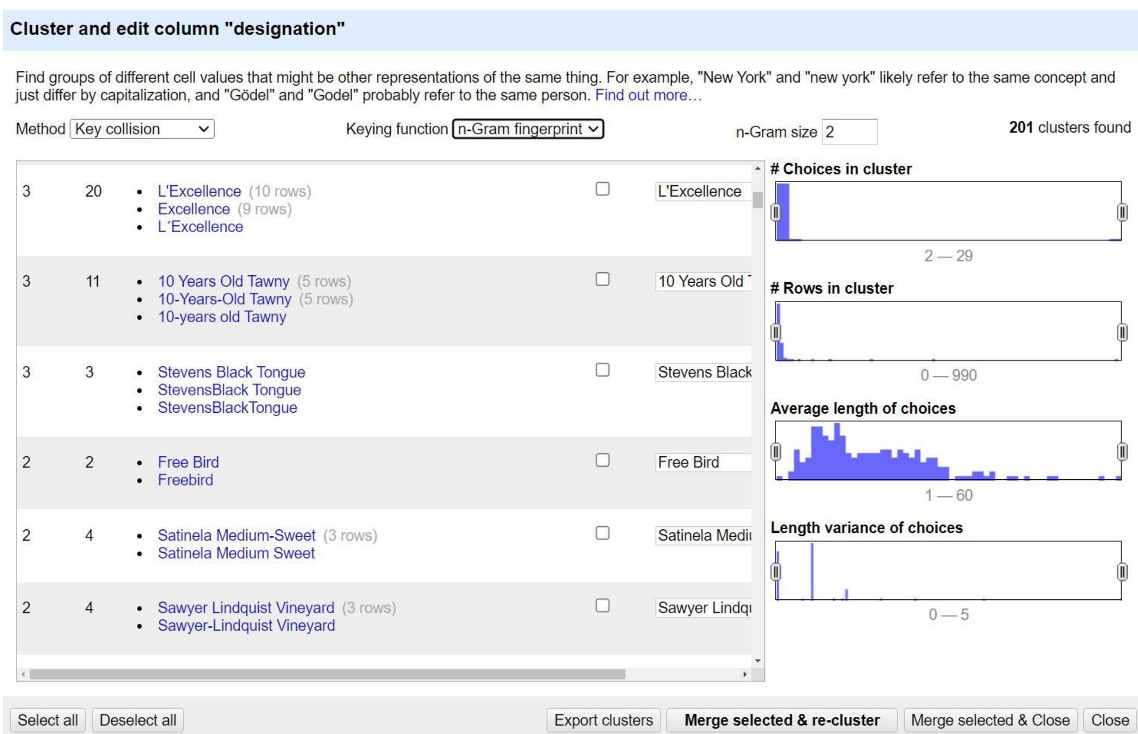
2.1 Column Changed from OpenRefine:

1. **Remove Null <country>:** Remove 63 rows.
2. **Remove Null <designation>:** Remove 37,454 rows.
3. **Remove Null <price>:** Remove 6,306 rows.
4. **Replace Null <region_1> with <province> value:** Text transform on 15,973 cells in column "region_1" join "province".
5. **Merge <region_1> and <region_2> to new column <region>**: Create new column region based on column "region_1" by filling 86,148 rows with "region_2".
6. **Remove ending ',' for <region> column**: Text transform on 52,064 cells in column region.
7. **Remove column <region_1>**: Remove column "region_1".
8. **Remove column <region_2>**: Remove column "region_2".
9. **Remove Null <taster_name>**: Remove 16,229 rows.
10. **Replace Null <taster_twitter_handle>**: Text transform on 3,833 cells in column "taster_twitter_handle".

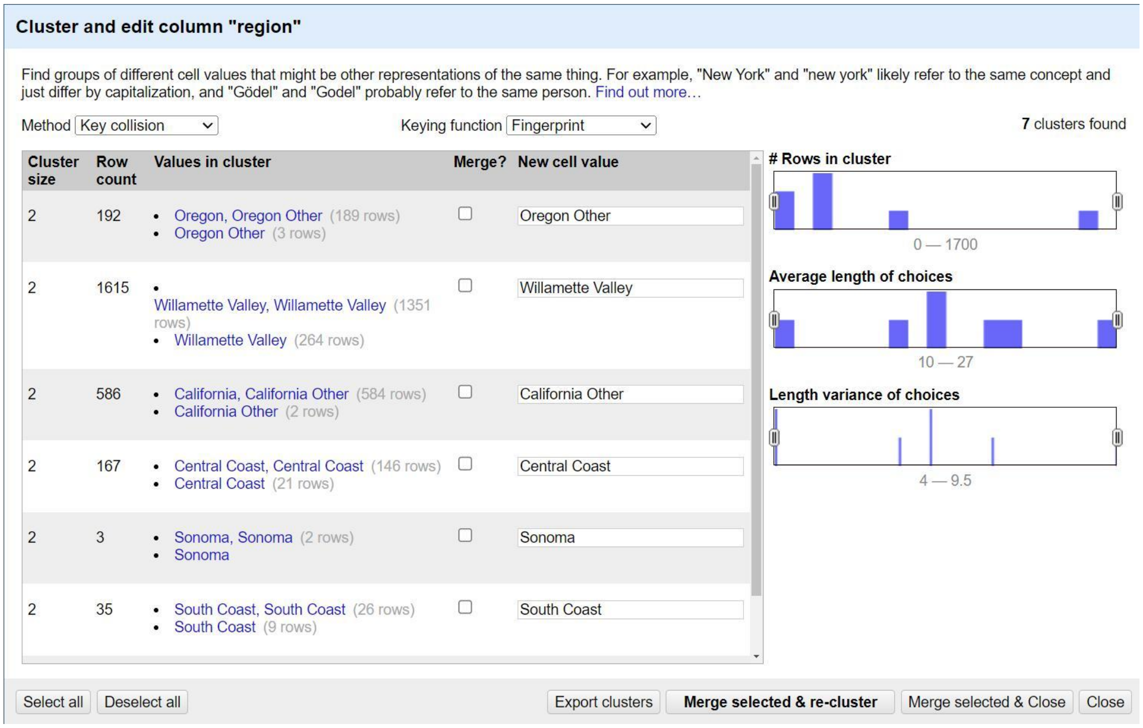
11. **Fingerprint Cluster <designation>**: Merge 993 clusters. Edit 8,637 cells.



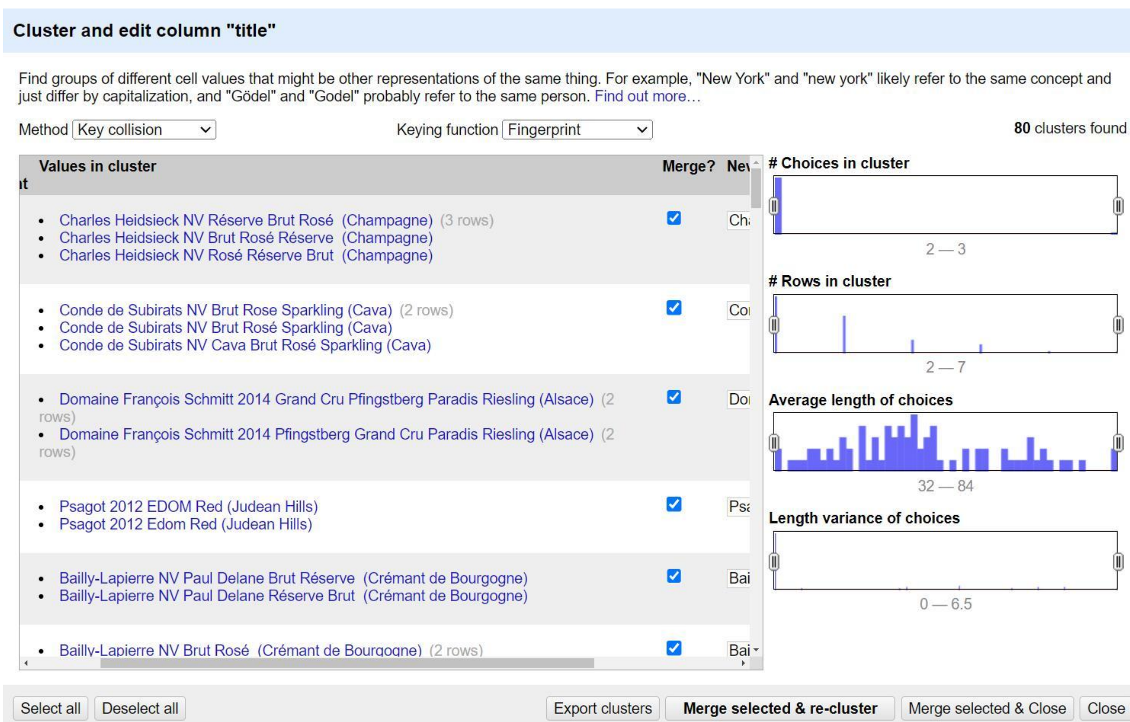
12. **N-gram=2 Cluster <designation>**: manually exclud 19 clusters, and merge 182 out of 201 clusters. Edit 2,800 cells.



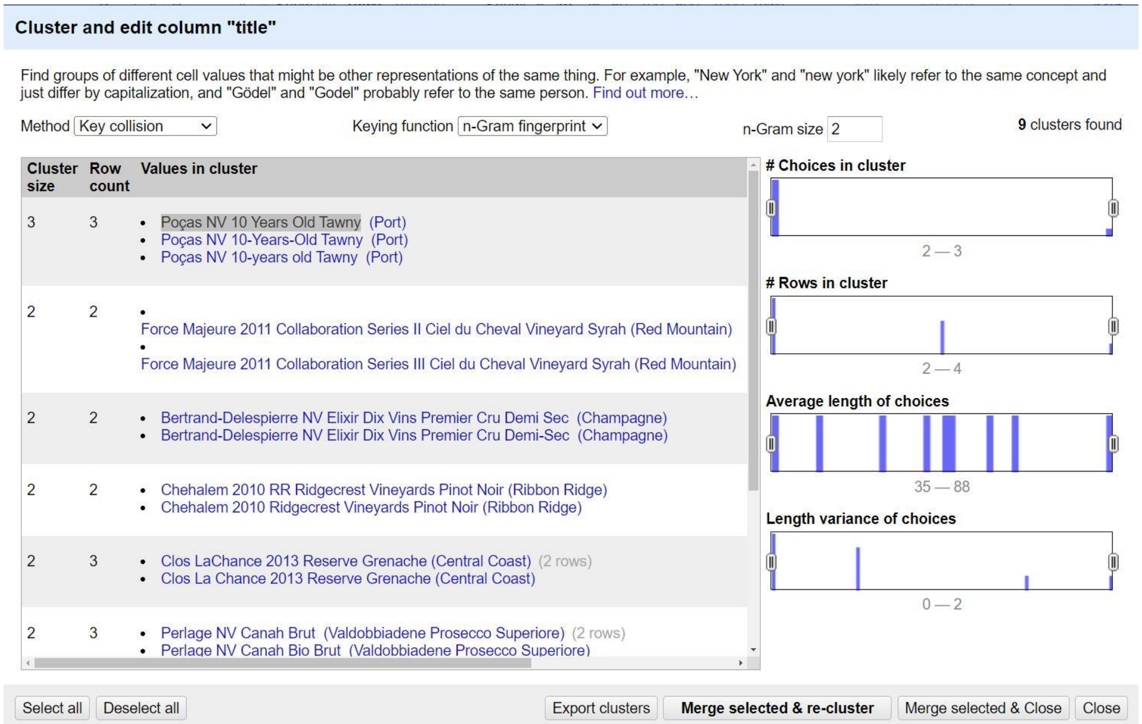
13. **Fingerprint Cluster <region>**: Merge 7 clusters. Edit 2,753 cells.



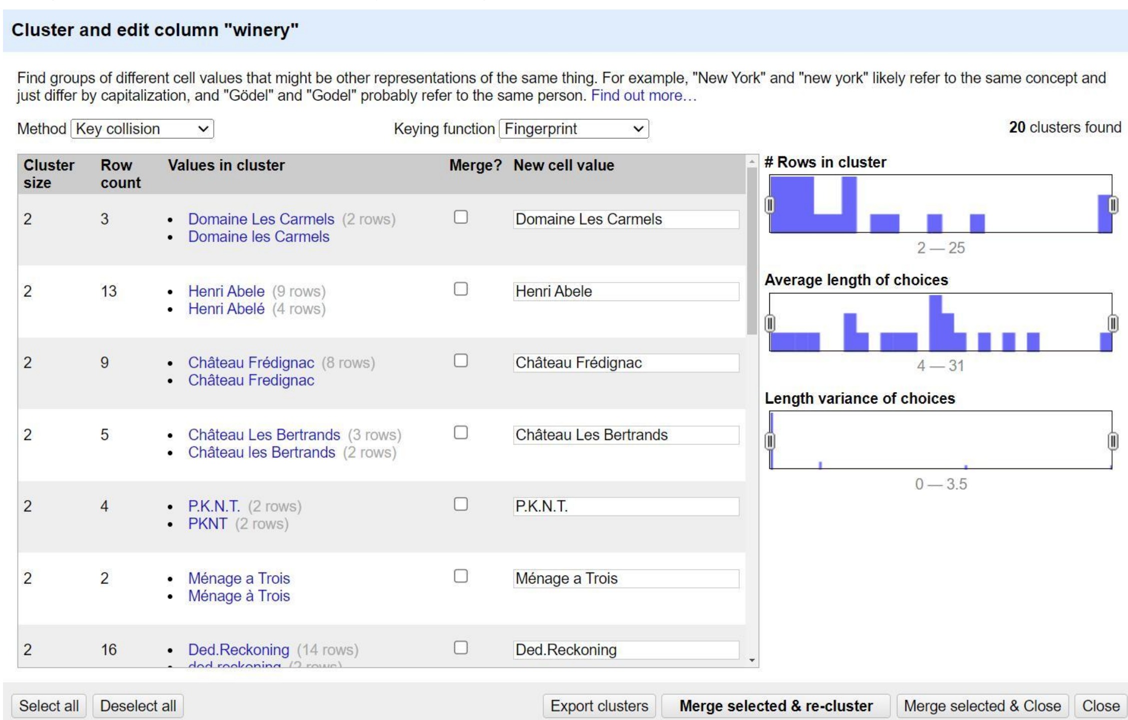
14. **Fingerprint Cluster <title>**: Merge 80 clusters. Edit 230 cells.



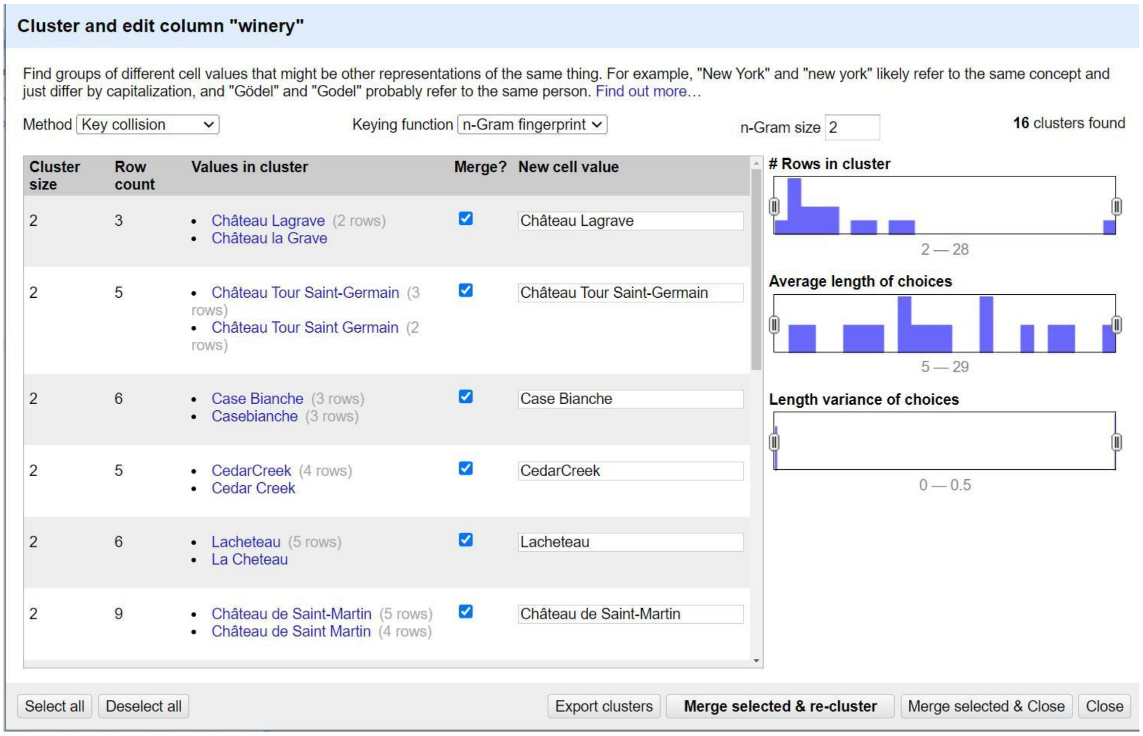
15. **N-gram=2 Cluster <title>**: Merge 9 clusters. Edit 23 cells.



16. **Fingerprint Cluster <winery>**: Merge 20 clusters. Edit 157 cells.



17. **N-gram=2 Cluster <winery>**: Merge 15 out of 16 clusters. Edit 108 cells.



18. **<id> to Number:** Text transform on 69,919 cells in column "id".
19. **<points> to Number**: Text transform on 69,919 cells in column "points".
20. **<price> to Number**: Text transform on 69,919 cells in column "price".
21. -40. **Trim Leading and Trailing & Collapsing Consecutive Whitespaces**: Text transform on 22,812 cells.

Before data cleaning: 129,971 rows. After data cleaning: 69,919 rows

2.2 Column Changed from Python:

1. **Normalize <points>**: Text transform on 69,919 cells in column "points".
2. **Reset index for <id>**: Text transform on 69,919 cells in column "id".

Before data cleaning: 69,919 rows. After data cleaning: 69,919 rows

2.3 Data Quality Improvement: IC violation with Python

We have also defined the following IC violation Checkers for our data using Python

1. **IC_empty_country**: 63 rows.

129908	129952	US	This Zinfandel from the e		90	22	California	Chiles Valle	Napa	Virginie Boo	@vboone	Houdini 201	Zinfandel	Houdini
129909	129967	US	Citation is given as much		90	75	Oregon	Oregon	Oregon Oth	Paul Gregut	@paulgwine	Citation 200	Pinot Noir	Citation
129910	913		Amber in co	Asureti Valle	87	30				Mike DeSim	@worldwin	Gotsa Famil	Chinuri	Gotsa Family Wines
129911	3131		Soft, fruity ε	Partager	83					Roger Voss	@vossroger	Barton & Gu	Red Blend	Barton & Guestier

2. **IC_empty_points**: 0 rows.

3. **IC_empty_price**: 8,996 rows.

15840	France	The wine is a velvet glov		96	2500	Bordeaux	Pomerol		Roger Voss	@vossroger	Château Pét	Bordeaux-st	Château Pétrus
98380	France	A superb wine from a gre		96	2500	Burgundy	La Romanée		Roger Voss	@vossroger	Domaine du	Pinot Noir	Domaine du Comte Liger-Belair
80290	France	This ripe wine shows ple		88	3300	Bordeaux	Médoc		Roger Voss	@vossroger	Château les	Bordeaux-st	Château les Ormes Sorbet
1844	Argentina	Cinnamon and licorice gi		83		Mendoza Pr	Mendoza		Michael Sch	@winescha	Cascada Pea	Cabernet Sa	Cascada Peak
7584	Argentina	A roasted, le	Reserve	85		Other	Neuquén		Michael Sch	@winescha	Alpataco 20	Merlot	Alpataco
12614	Argentina	Minerally m	Piedra Negr	86		Mendoza Pr	Uco Valley		Michael Sch	@winescha	François Lur	Pinot Gris	François Lurton

4. **IC_empty_taster_name**: 26,244 rows.

111475	US	Cinnamon, r	Estate Grow	88		California	Sonoma Co	Sonoma	Virginie Boone	@vboone	Cline 2013 E	Pinot Noir	Cline
112216	US	Soft on the	Cellar Club	85		California	Calistoga	Napa	Virginie Boone	@vboone	Sterling 201	Cabernet Sa	Sterling
112533	US	Chocolate-\	Reserve	88		California	Napa Valley	Napa	Virginie Boone	@vboone	Cosentino 2	Cabernet Sa	Cosentino
118073	US	Beautifully f	Rodgers Cre	93		California	Sonoma Co	Sonoma	Virginie Boone	@vboone	Landmark 2	Chardonnay	Landmark
119540	US	This lovely understated \		92		California	Coombsville	Napa	Virginie Boone	@vboone	Sodaro 201	Cabernet Sa	Sodaro
128678	US	Floral and ju	Limited Pro	90		California	Knights Vall	Sonoma	Virginie Boone	@vboone	Summers 20	Malbec	Summers
129214	US	Bright and li	Riserva	86		California	San Francisc	Central Coast	Virginie Boone	@vboone	Tamás Estat	Sangiovese	Tamás Estates
31530	US	Packaged in a cute yello		84	4	California	California	California Other			Bandit NV C	Chardonnay	Bandit
64590	US	There's a lot going on in		86	4	California	California	California Other			Bandit NV M	Merlot	Bandit
110255	US	A good everyday Merlot		84	4	California	California	California Other			Bandit NV M	Merlot	Bandit
100469	Australia	This bargain-basement A		81	5	Australia Ot	South Eastern Australia				Banrock Sta	Shiraz-Cabe	Banrock Station
3167	Italy	Packaged in	Mini	86	5	Veneto	Prosecco				Anna Spinat	Glera	Anna Spinato
102853	Italy	Definitely not Zinfandel-		83	5	Southern Ita	Puglia				Terrale 199	Primitivo	Terrale
102859	Italy	Terrale tran	Bianco	82	5	Sicily & Sard	Sicilia				Terrale 199	White Blenc	Terrale
102861	Italy	A thin peach nose is the		81	5	Lombardy	Oltrepò Pavese				Belmondo 1	Pinot Grigio	Belmondo

5. **IC_empty_taster_twitter_handle**: 26,244 rows.

111475	US	Cinnamon, r	Estate Grow	88		California	Sonoma Co	Sonoma	Virginie Boone	@vboone	Cline 2013 E	Pinot Noir	Cline
112216	US	Soft on the	Cellar Club	85		California	Calistoga	Napa	Virginie Boone	@vboone	Sterling 201	Cabernet Sa	Sterling
112533	US	Chocolate-\	Reserve	88		California	Napa Valley	Napa	Virginie Boone	@vboone	Cosentino 2	Cabernet Sa	Cosentino
118073	US	Beautifully f	Rodgers Cre	93		California	Sonoma Co	Sonoma	Virginie Boone	@vboone	Landmark 2	Chardonnay	Landmark
119540	US	This lovely understated \		92		California	Coombsville	Napa	Virginie Boone	@vboone	Sodaro 201	Cabernet Sa	Sodaro
128678	US	Floral and ju	Limited Pro	90		California	Knights Vall	Sonoma	Virginie Boone	@vboone	Summers 20	Malbec	Summers
129214	US	Bright and li	Riserva	86		California	San Francisc	Central Coast	Virginie Boone	@vboone	Tamás Estat	Sangiovese	Tamás Estates
31530	US	Packaged in a cute yello		84	4	California	California	California Other			Bandit NV C	Chardonnay	Bandit
64590	US	There's a lot going on in		86	4	California	California	California Other			Bandit NV M	Merlot	Bandit
110255	US	A good everyday Merlot		84	4	California	California	California Other			Bandit NV M	Merlot	Bandit
100469	Australia	This bargain-basement A		81	5	Australia Ot	South Eastern Australia				Banrock Sta	Shiraz-Cabe	Banrock Station
3167	Italy	Packaged in	Mini	86	5	Veneto	Prosecco				Anna Spinat	Glera	Anna Spinato
102853	Italy	Definitely not Zinfandel-		83	5	Southern Ita	Puglia				Terrale 199	Primitivo	Terrale
102859	Italy	Terrale tran	Bianco	82	5	Sicily & Sard	Sicilia				Terrale 199	White Blenc	Terrale
102861	Italy	A thin peach nose is the		81	5	Lombardy	Oltrepò Pavese				Belmondo 1	Pinot Grigio	Belmondo

6. **IC_empty_variety**: 1 rows.

102282	US	This variety	Estate Wilri	90	35	Washington	Naches Heig	Columbia Valley	Sean P. Sullivan	@wawinere	Wilridge 2013 Estate Wilridge	Zweigelt	Wilridge
86909	Chile	A chalky, dusty mouthfe		88	17	Maipo Valley					Carmen 1999 (Maipo Valley)		Carmen

7. **IC_is_numeric_points**: 0 rows.
8. **IC_is_numeric_price**: 0 rows.
9. **IC_points_range_points**: 0 rows.
10. **IC_points_range_price**: 0 rows.
11. **IC violation check after data cleaning:** No IC violations are found.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Column	country	description	designation	points	price	province	region	taster_nam	taster_twitt	title	variety	winery	normalized_points	
2	1	Portugal	This is ripe a	Avidagos	87	15	Douro	Douro	Roger Voss	@vossroger	Quinta dos A	Portuguese	Quinta dos A	3.5	
3	2	US	Pineapple ri	Reserve Lat	87	13	Michigan	Lake Michig	Alexander P	@WineEnth	St. Julian 20	Riesling	St. Julian	3.5	
4	3	US	Much like th	Vintner's Re	87	65	Oregon	Willamette	Paul Gregut	@paulgwine	Sweet Chee	Pinot Noir	Sweet Chee	3.5	
5	4	Spain	Blackberry a	Ars In Vitro	87	15	Northern Sp	Navarra	Michael Sch	@winescha	Tandem 201	Tempranillo	Tandem	3.5	
6	5	Italy	Here's a brig	Belsito	87	16	Sicily & Sard	Vittoria	Kerin O'Kee	@kerinokee	Terre di Giu	Frappato	Terre di Giu	3.5	
7	6	Germany	Savory driec	Shine	87	12	Rheinhesser	Rheinhesser	Anna Lee C.	@WineEnth	Heinz Eifel 2	Gewürztram	Heinz Eifel	3.5	
8	7	France	This has gre	Les Natures	87	27	Alsace	Alsace	Roger Voss	@vossroger	Jean-Baptis	Pinot Gris	Jean-Baptis	3.5	
9	8	US	Soft, supple	Mountain C	87	19	California	Napa Valley	Virginie Boo	@vboone	Kirkland Sigr	Cabernet Sa	Kirkland Sigr	3.5	
10	9	Germany	Zesty orang	Devon	87	24	Mosel	Mosel	Anna Lee C.	@WineEnth	Richard Böc	Riesling	Richard Böc	3.5	
11	10	Argentina	Baked plum	Felix	87	30	Other	Cafayate	Michael Sch	@winescha	Felix Lavaqu	Malbec	Felix Lavaqu	3.5	
12	11	Argentina	Raw black-c	Winemaker	87	13	Mendoza Pr	Mendoza	Michael Sch	@winescha	Gaucho And	Malbec	Gaucho And	3.5	
13	12	Spain	Desiccated I	Vendimia Se	87	28	Northern Sp	Ribera del D	Michael Sch	@winescha	Pradorey 20	Tempranillo	Pradorey	3.5	
14	13	US	Ripe aroma:	Vin de Mais	87	23	Virginia	Virginia	Alexander P	@WineEnth	Quiévremor	Red Blend	Quiévremor	3.5	
15	14	Italy	Delicate arc	Ficiligno	87	19	Sicily & Sard	Sicilia	Kerin O'Kee	@kerinokee	Baglio di Pia	White Blenc	Baglio di Pia	3.5	

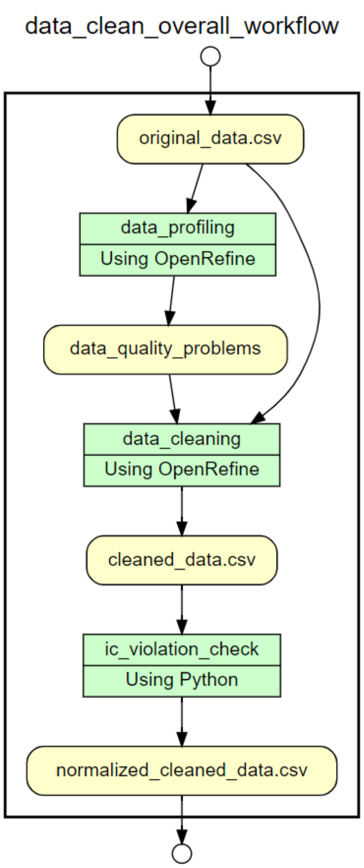
We can see that there were IC violations when running the query on the original data, but after the data cleaning, all IC violations are resolved. This demonstrates a significant improvement in data quality.

2.4 U1 Query with Python SQL

After data cleaning, we successfully executed the U1 query:

```
most_significant_reviewer: "Virginie Boone": @vboone, num_of_reviews: 6739
                         variety  num_of_reviews  avg_points  min_price  max_price
                        Abouriou               1    2.500000       75.0       75.0
                       Aglianico               1    3.000000       58.0       58.0
                        Albariño               8    4.687500       18.0       35.0
               Alicante Bouschet               2    4.500000       30.0       36.0
                      Alvarelhão               1    2.500000       18.0       18.0
                          Arneis               2    4.000000       17.0       38.0
                         Barbera              24    3.687500       11.0       35.0
                    Black Muscat               1    4.500000       25.0       25.0
        Bordeaux-style Red Blend             219    4.835616       20.0      350.0
                  Cabernet Blend              12    4.833333       18.0      100.0
                  Cabernet Franc              78    4.166667       20.0      140.0
           Cabernet Franc-Merlot               2    4.500000       38.0      125.0
              Cabernet Sauvignon            1148    4.956882       10.0      625.0
Cabernet Sauvignon-Cabernet Franc              2    3.250000       45.0       48.0
       Cabernet Sauvignon-Merlot               5    4.800000       30.0       95.0
   Cabernet Sauvignon-Sangiovese               1    5.000000       50.0       50.0
        Cabernet Sauvignon-Syrah               5    4.900000       28.0       70.0
  Cabernet Sauvignon-Tempranillo               1    3.500000       18.0       18.0
                        Carignan               3    4.166667       28.0       30.0
```

3. Workflow Model



3.1 Outer Workflow W1

The overall data cleaning workflow can be summarized as follows:

1. original_data.csv: This is the initial dataset containing wine reviews, which may have various data quality issues such as missing values, inconsistencies, and formatting errors.
2. data_profiling Using Open Refine:

The original_data.csv is imported into OpenRefine, where data profiling is performed to gain insights into the data quality problems. OpenRefine identifies issues like missing values, whitespace, and inconsistent data formats.

3. data_quality_problems: Based on the data profiling results, data quality problems are identified, such as missing values in certain columns, non-numeric data in numeric columns, or out-of-range values.
4. data_cleaning using OpenRefine: OpenRefine is used to clean and transform the data by addressing the data quality problems. Operations like removing null values, replacing missing data, standardizing formats, and clustering similar entries are performed to improve the data quality.

5. cleaned_data.csv: After data cleaning, the dataset is exported as cleaned_data.csv, which now has improved data quality, consistency, and completeness compared to the original dataset.
6. ic_violation_check using Python: In this step, Python scripts are utilized to perform IC (Integrity Constraint) violation checks on the cleaned_data.csv. These checks verify if the data adheres to specific rules, such as ensuring numeric columns contain valid numerical values within certain ranges.
7. normalize_cleaned_data.csv: After the IC violation checks, the cleaned_data.csv is further processed to normalize ratings and re-index ids. Normalization converts the data into a standardized range, making it more intuitive for analysis and visualization.

Overall, this workflow demonstrates the step-by-step process of data cleaning, data profiling, addressing data quality issues, and ensuring data integrity, ultimately leading to a more reliable and accurate dataset for analysis and querying.

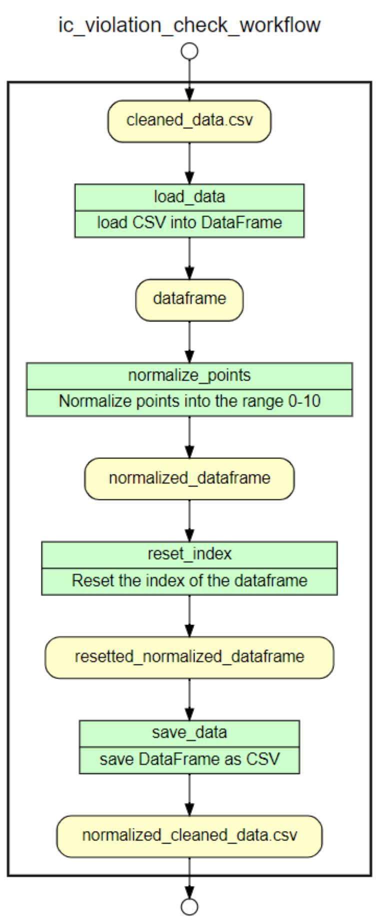
3.2 Inner Workflow W2

The Inner Workflow Diagrams provide a visual representation of the workflow, where the narrative description are provided in section **1. Data Cleaning Methods**.

a. OpenRefine Workflow

Since there are 40 steps in the OpenRefine Workflow, it is too large to fit here, so we provide an image URL for your reference:
https://github.com/jsc723/cs513-proj/blob/main/workflows/W2_OpenRefine/graphviz.png

b. Python Workflow



4. Conclusions & Summary

In this project, we focused on data profiling and data cleaning using OpenRefine and Python to analyze wine review data. We designed a comprehensive workflow, consisting of 42 steps, to clean and transform the dataset into a consistent and reliable format. By utilizing tools like YesWorkflow and or2pw, we visualized our workflow, facilitating better understanding and communication.

Throughout the project, we encountered challenges, such as handling missing data and addressing IC violations. We learned the significance of data integrity checks, ensuring the accuracy and reliability of our dataset for meaningful analysis. Additionally, we realized the importance of flexibility in our assumptions to accommodate unexpected data characteristics.

Concluding the project, we successfully achieved our goals in data cleaning and designed an efficient Python script for query purposes. We improved our understanding of data cleaning techniques and gained practical experience with various data cleaning tools. Moving forward, we plan to extend our use case to create a more interactive system, empowering users to explore wine reviews efficiently.

Overall, this project provided valuable insights into data cleaning methodologies and their relevance in data analysis, reinforcing the importance of thorough and systematic data preparation to derive meaningful insights and make informed decisions.

5. Contribution

In this project, Yunfei worked on data profiling and data cleaning using OpenRefine and the OpenRefine part of the detailed workflow W2. Sicheng worked on the data cleaning and IC violations using Python, the workflow W1 and the Python part of the detailed workflow W2. We contributed equally to the project