

MATCH 'N MERGE

Jordan Cox

Rob Holbrook

Ramon Lim

Stephanie Zhu

12/08/2017

PITCH

Match N Merge Application - An easy-to use and flexible user interface-driven application takes two different data sets, utilizes machine-learning to develop models to predict column matches between data sets based on generated column value features and then uses Ratcliff/Obershelp pattern recognition to match similar rows ultimately resulting in a combined data set matched by column and row that can be used for understanding data and developing future machine learning models.

Background – Motivation and Problem Statement	3
Approach	3
Column Match N Merge	3
Row Match N Merge	4
User Interface and Shell	4
Results	4
Future Enhancements	4
Summary and Learning Journey	5
Team Contributions	5
Appendix	7
Column Matching: Preparing The Data	7
Launching the Match N Merge Application	8
Running the Match N Merge Application	8
Step 1: Select CSV Data Sets	8
Step 2: Match N Merge Columns	8
Step 3: Review Mappings	9
Step 4: Match N Merge Rows	10
Results With Different Datasets	11
Example 1 - San Francisco Restaurants:	11
Datasets	11
Mappings	12
Output - Results.csv (sample columns)	12
Findings	13
Example 2 - Periodic Table of Elements:	14
Datasets	14
Mappings	15

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Findings	15
Example 3 - Wine:	16
Datasets	16
Mappings	16
Findings	17

Background – Motivation and Problem Statement

In an increasingly interconnected world, people are interested in getting more out of their data than ever before. However, in a typical company, only administrators familiar with database querying languages such as SQL or analysts with domain knowledge of the different data repositories can extract, understand dependencies, and analyze data appropriately. As a result, these skill sets are often a barrier to entry for companies and stakeholders who want to leverage their data but don't necessarily have the skill to do so. Our entire group has dealt with this problem firsthand in industry and see the value of designing an application that can quickly merge and join datasets together without relying on querying languages or deep knowledge of the datasets. Our tool will use machine learning to detect similarities between datasets, remove duplicated columns and rows, and return a clean dataset for use and further analysis. Value for this application ranges from small scale use involving merging of data from disparate datasets to the larger scale use allowing companies in merger and acquisitions scenarios to integrate their data together quickly, seamlessly, and efficiently.

Approach

We took a two-step serial approach to merging data : 1) map the columns and 2) map the rows.

Column Match N Merge

The program currently requires two data sets the user wants to merge. The first dataset will be used to train the machine learning algorithm to create prediction models of matching columns based on features. These models will be applied to the second data set to match columns across datasets.

First, data must be organized into a standardized format. Both datasets must be “flattened” so each row of data is collapsed into one column. Nine new features based on the flattened column will be appended: length, number of letters, number of numerical digits, number of punctuation, number of whitespaces, % of letters, % of numerical digits, % of punctuation, and % of whitespaces.

Secondly, a new table is generated assigning the original column label to each index. This column label is considered the actual “Y” value. Using the first dataset, we train with our nine new features to predict the “Y” value. Our training accuracy with both random forest and support vector machines (SVM) was 90%; for the validation set, we chose to use SVM. Next, using the second dataset, we test with our nine new features and the column labels from the first dataset. The code makes a “Y” prediction for each row of data, the prediction categories being the column label from the first dataset. The predictions and actuals in the second dataset are compared for similarities and we extract a table of each unique pairing. We calculate the relative accuracy to show how many predictions out of total predictions fall into that particular category.

One caveat that must be mentioned is we only show relative accuracy - that is, the algorithm will make a prediction on the second dataset based only on the column labels from the first dataset. If the columns do not match, the algorithm will still make a best guess prediction from the available column labels.

Row Match N Merge

The row alignment section of the program is designed to align rows across the two datasets based on the column mapping identified in the previous step. It starts by opening both datasets as well as the column alignment file. The column alignment file has both the suggested column alignment from the previous section as well as the user feedback on the alignment. The user feedback is used to create a mapping of which columns in the first dataset align with which columns in the second dataset.

The program performs some basic string cleansing by removing all punctuation and converting the strings to lowercase. It then takes the aligned columns for each dataset and creates a new column “bigKey” that combines all the values for each row that are matched in the second dataset as long strings. Each of these strings in the first dataset is then compared to the strings in the second dataset to identify which row is the most like the target string. This value is generated using the sequence matcher function from the difflib function (which applies the Ratcliff/Obershelp pattern recognition algorithm).

The row matcher then creates a new table with the data from the first table, the index of the most similar row in the second table, the alignment score created by the difflib function, and the data from the second dataset that aligns with that row. This data is finally saved to a file.

User Interface and Shell

The user interface (UI) is an easy-to-use application that guides the user step by step. It relies on the Flexx package, HTML, and Javascript interfaces to Python code. Launched from a Jupyter Notebook, it presents a webpage that prompts the user through a four stage process: 1) select the datasets, 2) start the column mapping process, 3) present the column mappings and allow the user to manually override such mappings, and 4) initiate the row matching process and present the user a consolidated dataset matched on column and row. All throughout, status is updated to the user. In the appendix, the “Running the Match N Merge Application” section details the UI process.

Results

We tested the performance against many different data sets. Three of those pairs of datasets are outlined in the “Results With Different Datasets” Appendix section. The conclusion is that predictive “merge n mapping” performance is highly dependent on the features that are initially utilized for matching columns. Generally, our features identified the alpha-numeric mix-up of dataset values so it makes sense that the strongest models were those tied to datasets that had column values somewhat identifiable by its alpha-numeric combination. Conversely, there was poor matching on dataset values that were either mostly numeric or not clearly identifiable by the mix of alpha and numeric characters.

Future Enhancements

The purpose of the Match N Merge algorithm is to provide enterprise solutions to data consolidation problems. In its current form, Match N Merge is only viable for smaller datasets with certain column characteristics. In future iterations of the model, we intend to implement the following changes to improve column matching accuracy and general code efficiency:

- **Create more features:** Additional features will allow for a significant increase in the types of columns the model is able to successfully match.
- **Create shadow features to identify unique columns:** Shadow features will allow the model to identify unique columns, i.e. columns found in only one dataset. Unique columns will no longer be force matched to unrelated columns based on relative accuracy.
- **Fill in missing data using probabilistic joins:** Inferential statistics may be able to extract relationships from both datasets to input missing data values.
- **Expand the number & types of mergeable datasets:** The prototype is only able to merge 2 .csv files. Future iterations would be able to merge multiple .json, .xlsx, RSS feeds, etc. file types.
- **Improve code efficiency:** Improvements to the column match and row merge code blocks would enable increased scalability.

Summary and Learning Journey

From the onset, our team was interested in examining the value that could be gained from exposing the relationship between similar datasets. The initial conception was to use probabilistic joins and machine learning to input missing data values between related datasets. As we shared our collective industry backgrounds, we recognized a common theme that highlighted the tedious nature of managing data from multiple sources. At this point, the focus shifted from missing data towards data consolidation. Specifically, we wanted to create a tool that could use machine learning to automate the process of merging multiple datasets into a single data source with minimal manual intervention.

Despite a unified direction, we realized that progress would be slowed due to a lack of machine learning skills amongst the team. While this approach helped compile a strong composition of working code, it necessitated individual research in order to assemble the blocks of code (e.g. column match, row merge) into a fluid, working model. Further, we determined that the model would require a UI in order to be user-friendly. This required extensive work outside of the curriculum to learn how to build a functional, intuitive interface. Currently, the focus is to learn more sophisticated machine learning techniques that can improve the predictive accuracy of the algorithm and improve the overall efficiency of the code.

Team Contributions

Our team's approach has heavily emphasized group discussion and collaboration. We met at least weekly to seek feedback and advice for each component of the project, discuss use cases of the model, concerns with code, which techniques we should focus our learning on, and assign individual responsibilities that each team member would then report on in the following week. Decision making was, without exception, a group endeavor. While the general decision making and strategy was a collective effort, there was individual components to the code writing as follows:

- **Stephanie:** Primarily responsible for the column matching block of code.
- **Jordan:** Along with Stephanie, contributed to the column match code.
- **Rob:** Compiled the row merge code.
- **Ramon:** Built the UI and performed quality assurance checks of the assembled model.

Appendix

Column Matching: Preparing The Data

	key2
0	Tiramisu Kitchen
1	033 Belden Pl
2	San Francisco
3	CA
4	94104

	y2
0	business_name
1	business_address
2	business_city
3	business_state
4	business_postal_code

	len	let	num	ws	punc	%let	%num	%ws	%punc
0	16	15	0	1	0	0.937500	0.000000	0.062500	0.0
1	13	8	3	2	0	0.615385	0.230769	0.153846	0.0
2	13	12	0	1	0	0.923077	0.000000	0.076923	0.0
3	2	2	0	0	0	1.000000	0.000000	0.000000	0.0
4	5	0	5	0	0	0.000000	1.000000	0.000000	0.0

	0	1	2	3	4	5	6	7	8
0	899	0	0	0	0	10	0	0	0
1	0	329	0	0	0	0	0	595	0
2	0	0	937	0	0	0	0	0	0
3	0	0	0	574	3	0	0	0	0
4	0	0	0	0	618	0	0	0	0
5	10	0	33	0	0	850	0	0	1
6	0	0	0	0	0	0	274	0	0
7	0	0	0	0	0	0	0	895	1
8	0	0	0	0	0	0	0	0	946

Launching the Match N Merge Application

1. Install the Flexx package, <https://flexx.readthedocs.io/en/latest/>
2. Download the latest Match N Merge code by doing a GET on <https://github.com/rholbrook98/matchnmerge>
3. Locate Jupyter notebook, MatchNMerge.ipynb
4. Ensure that the following files do NOT exist in the current Jupyter notebook folder: results.csv, columnMapping.csv, columnMapping_user.csv, filesReady.txt, status.txt. If any of these exists, delete them on the filesystem
5. Open Jupyter notebook, MatchNMerge.ipynb
 - a. Note: Jupyter notebook must be run in Firefox, Mozilla
 - b. Note: Default download file path for Firefox must be the folder where the notebook is located
6. Select Kernel -> Restart & Run All
 - a. Javascript/HTML UI Application will be presented

Running the Match N Merge Application

Step 1: Select CSV Data Sets



Note: This must be run in firefox with the default download file path set to: `///C:/Users/tlm/fuzzy`. Additionally, this file must be executed from Jupyter Notebook, MatchNMerge.ipynb

Step 1: Load data set csv files

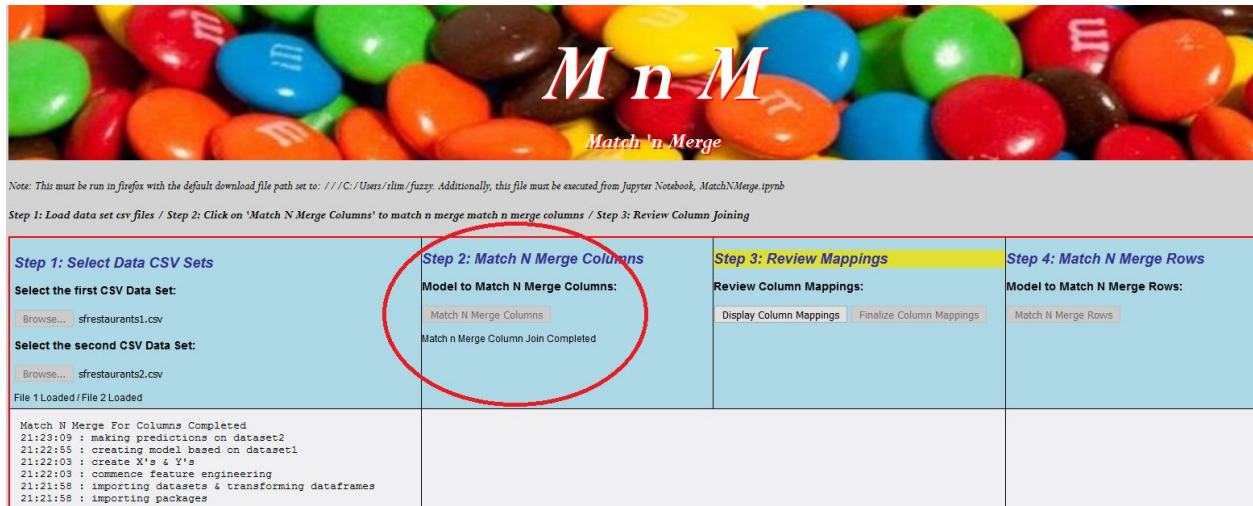
Step 1: Select Data CSV Sets	Step 2: Match N Merge Columns	Step 3: Review Mappings	Step 4: Match N Merge Rows
Select the first CSV Data Set: Browse... No file selected.	Model to Match N Merge Columns: Match N Merge Columns	Review Column Mappings: Display Column Mappings Finalize Column Mappings	Model to Match N Merge Rows: Match N Merge Rows
Select the second CSV Data Set: Browse... No file selected.			

- Save the CSV file to the same folder as the Jupyter notebook (set the default file download path in the Firefox browser to download files to that folder)
- Select the first data set (CSV file)
- Select the second data set (CSV file)

Step 2: Match N Merge Columns

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Click on the **“Match N Merge Columns”** button to start the machine learning modeling process to match columns across the two data sets



The screenshot shows the Match 'N Merge application interface. At the top is a banner with colorful M&M's candies and the text 'M n M Match 'n Merge'. Below the banner is a note about running the application. The main interface is divided into four steps: Step 1: Select Data CSV Sets, Step 2: Match N Merge Columns, Step 3: Review Mappings, and Step 4: Match N Merge Rows. Step 2 is circled in red. The status box at the bottom left shows the progress of the match n merge process.

Note: This must be run in firefox with the default download file path set to: `///C:/Users/rlm/fuzzy`. Additionally, this file must be executed from Jupyter Notebook, `MatchNMerge.ipynb`

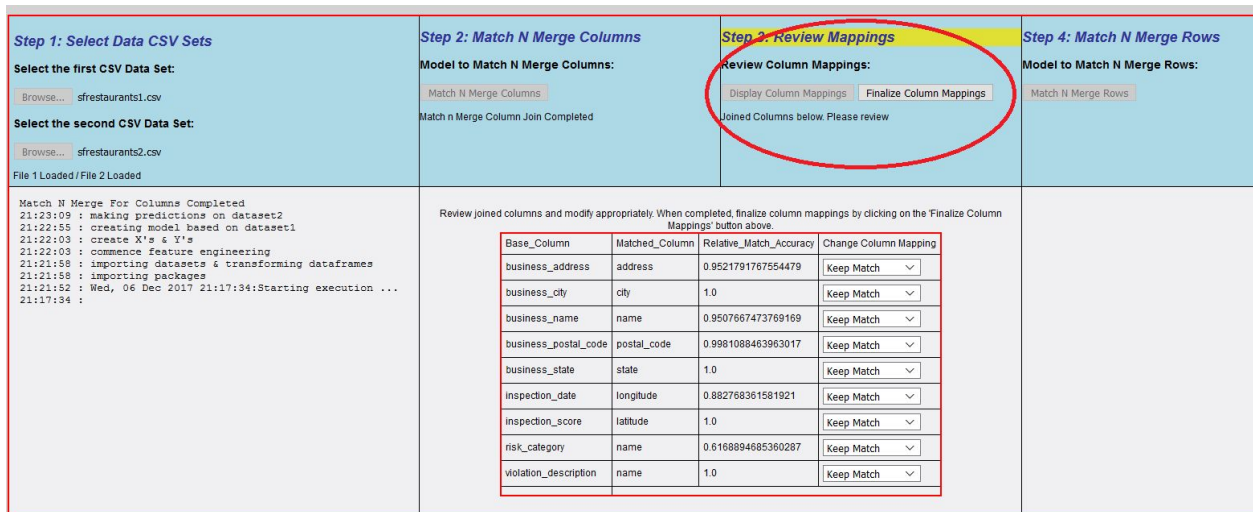
Step 1: Load data set csv files / Step 2: Click on 'Match N Merge Columns' to match n merge match n merge columns / Step 3: Review Column Joining

Step 1: Select Data CSV Sets	Step 2: Match N Merge Columns	Step 3: Review Mappings	Step 4: Match N Merge Rows
Select the first CSV Data Set: <input type="button" value="Browse..."/> <code>sfrestaurants1.csv</code> Select the second CSV Data Set: <input type="button" value="Browse..."/> <code>sfrestaurants2.csv</code> File 1 Loaded / File 2 Loaded	Model to Match N Merge Columns: <input type="button" value="Match N Merge Columns"/> Match n Merge Column Join Completed	Review Column Mappings: <input type="button" value="Display Column Mappings"/> <input type="button" value="Finalize Column Mappings"/> <input type="button" value="Match N Merge Rows"/>	Model to Match N Merge Rows: <input type="button" value="Match N Merge Rows"/>
Match N Merge For Columns Completed 21:23:09 : making predictions on dataset2 21:22:55 : creating model based on dataset1 21:22:03 : create X's & Y's 21:22:03 : commence feature engineering 21:21:58 : importing datasets & transforming dataframes 21:21:58 : importing packages			

- Wait for column modeling background python function to complete
- Status box on the bottom left-hand section of the UI informs you when this has completed
- Once completed, the **“Display Column Mappings”** button is enabled

Step 3: Review Mappings

Click on the **“Display Column Mappings”** button to view the best column matches based on the results of the predictive modeling.












The screenshot shows the Match 'N Merge application interface with Step 3: Review Mappings highlighted. The 'Display Column Mappings' button is circled in red. The status box at the bottom left shows the progress of the match n merge process. A table of joined columns is displayed in the center of the interface.

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
business_address	address	0.9521791767554479	<input type="button" value="Keep Match"/>
business_city	city	1.0	<input type="button" value="Keep Match"/>
business_name	name	0.9507667473769169	<input type="button" value="Keep Match"/>
business_postal_code	postal_code	0.9981088463963017	<input type="button" value="Keep Match"/>
business_state	state	1.0	<input type="button" value="Keep Match"/>
inspection_date	longitude	0.882768361581921	<input type="button" value="Keep Match"/>
inspection_score	latitude	1.0	<input type="button" value="Keep Match"/>
risk_category	name	0.6168894685360287	<input type="button" value="Keep Match"/>
violation_description	name	1.0	<input type="button" value="Keep Match"/>

Mapping table presented to the user:

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
business_address	address	0.9521791767554479	Keep Match 
business_city	city	1.0	Keep Match 
business_name	name	0.9507667473769169	Keep Match 
business_postal_code	postal_code	0.9981088463963017	Keep Match 
business_state	state	1.0	Keep Match 
inspection_date	longitude	0.882768361581921	Keep Match 
inspection_score	latitude	1.0	Keep Match 
risk_category	name	0.6168894685360287	Keep Match 
violation_description	name	1.0	Keep Match 

Column Mapping Table:

Base_Column: Columns from data set 2

Match_Column: Best column match to data set 1

Relative_Match_Accuracy: Relative measure of column matching modeling performance

Change Column Mapping: Legend:

Keep Match -- keep matching of data set 2 column to data set 1 column

Do Not Match -- do not match these two columns (will be excluded from row match)

Data Set 1 Column Name - match specific data set 1 column to data set 2 column

Click on the ***"Finalize Column Mappings"*** button to finalize changes.

Step 4: Match N Merge Rows

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Click on the **“Match N Merge Rows”** button to start the process of matching rows across the two data sets based on the column matches

Step 1: Select Data CSV Sets
Select the first CSV Data Set:
Browse... sfrestaurants1.csv
Select the second CSV Data Set:
Browse... sfrestaurants2.csv
File 1 Loaded / File 2 Loaded

Step 2: Match N Merge Columns
Model to Match N Merge Columns:
Match N Merge Columns
Match n Merge Column Join Completed

Step 3: Review Mappings
Review Column Mappings:
Display Column Mappings | Finalize Column Mappings
Column Mappings Finalized

Step 4: Match N Merge Rows
Model to Match N Merge Rows:
Match N Merge Rows
Match n Merge Row Join Completed

Match N Merge For Rows Completed
21:38:17 : Creating results output file
21:38:17 : Aligning rows
21:36:33 : Starting row merge
21:36:33 : Match N Merge For Columns Completed
21:23:09 : making predictions on dataset2
21:22:55 : creating model based on dataset1
21:22:03 : create X's & Y's
21:22:03 : commence feature engineering
21:21:58 : importing datasets & transforming dataframes
21:21:58 : importing packages
21:21:52 : Wed, 06 Dec 2017 21:17:34:Starting execution ...
21:17:34 :

Review joined columns and modify appropriately. When completed, finalize column mappings by clicking on the Finalize Column Mappings' button above.

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
business_address	address	0.9521791767554479	Keep Match
business_city	city	1.0	Keep Match
business_name	name	0.9507667473769169	Keep Match
business_postal_code	postal_code	0.9981088463963017	Keep Match
business_state	state	1.0	Keep Match
inspection_date	longitude	0.882768361581921	Keep Match

[Download results.csv file](#)

When process has completed, click on the 'Download results.csv file' to view results.

Results With Different Datasets

Example 1 - San Francisco Restaurants:

Datasets

Data Set 2 Column Name	Example Value	Data Set 1 Column Name	Example Value
business_name	Tiramisu Kitchen	business_id	67
business_address	033 Belden Pl	name	Tiramisu Kitchen
business_city	San Francisco	address	033 Belden Pl
business_state	CA	city	San Francisco
business_postalcode	94104	state	CA
inspection_date	5/3/2016 12:00:00 AM	postal_code	94104
inspection_score	82	latitude	37.79112
violation_description	High risk vermin infestation	longitude	-122.404
risk_category	High Risk	phone_number	4156766766

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Mappings

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
business_address	address	0.9521791767554479	Keep Match ▼
business_city	city	1.0	Keep Match ▼
business_name	name	0.9507667473769169	Keep Match ▼
business_postal_code	postal_code	0.9981088463963017	Keep Match ▼
business_state	state	1.0	Keep Match ▼
inspection_date	longitude	0.882768361581921	Do Not Match ▼
inspection_score	latitude	1.0	Do Not Match ▼
risk_category	name	0.6168894685360287	Do Not Match ▼
violation_description	name	1.0	Do Not Match ▼

Output - Results.csv (sample columns)

Note:

- matchCertainty: Ratcliff/Obershelp pattern recognition for string similarities figure (closer to 1 = closer match)
- business_name_ds2 : Business name column from data set 2
- name_ds1: Business name column from data set 1

business_name_ds2	name_ds1	matchCertainty
Tiramisu Kitchen	TIRAMISU KITCHEN	1
CARVER ELEMENTARY	GEORGE'S COFFEE SHOP	0.724137931
Nrgize Lifestyle Cafe	NRGIZE LIFESTYLE CAFE	1
OMNI S.F. Hotel - 2nd	OMNI S.F. HOTEL - 2ND FLO	1
Chico's Pizza	CHICO'S PIZZA	0.988505747
Norman's Ice Cream a	NORMAN'S ICE CREAM AN	0.99270073
CHARLIE'S DELI CAFE	CHARLIE'S DELI CAFE	0.99047619
ART'S CAFE	ART'S CAFE	0.988235294
SUSHI ZONE	SUSHI ZONE	0.989010989
Oza OZa	RHODA GOLDMAN PLAZA	0.795698925
CAFE X + O	CAFE X + O	0.988505747
Oasis Grill	OASIS GRILL	0.988235294
Chowders	CHOWDERS	1
STARBUCKS	STARBUCKS COFFEE	0.914893617

Findings

Generally, from the results.csv, the match certainty (close to 1 is better) provides decent results as indicated by the similarities in business names between the business name column from data set 1 and data set 2.












Example 2 - Periodic Table of Elements:*Datasets*

(first 10 columns shown)			
Data Set 2 Column Name	Example Value	Data Set 1 Column Name	Example Value
number	1	atomicNumber	1
name	Hydrogen	symbol	H
symbol	H	name	Hydrogen
name_symbol	Hydrogen, H	atomicMass	1.00794(4)
pronunciation	/ˈhaʔdrʔdʔn/, HY-drʔ-jʔn	cpkHexColor	FFFFFF
appearance	colorless gas	electronicConfiguration	1s1
atomic_number	1	electronegativity	2.2
group_block	group 1, s-block	atomicRadius	37
period	1	ionRadius	76 (+1)
element_category	diatomic nonmetal	vanDelWaalsRadius	120
atomic_weight	1.008	ionizationEnergy	1312
electron_configuration	1s1	electronAffinity	-73
per_shell	1	oxidationStates	-1, 1
color	colorless	standardState	gas
phase	gas	bondingType	diatomic
		meltingPoint	14
		boilingPoint	20
		density	8.99E-05
		groupBlock	nonmetal
		yearDiscovered	1766

Match 'N Merge – IEOR 290 – Cox, Holbrook, Lim, Zhu

Mappings

First 11 mappings

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
allotropes	oxidationStates	0.5714285714285714	Keep Match 
alternative_name	yearDiscovered	0.4	Keep Match 
alternative_names	oxidationStates	1.0	Keep Match 
appearance	oxidationStates	0.38235294117647056	Keep Match 
at_t(k)	boilingPoint	0.6190476190476191	Keep Match 
atomic_number	atomicNumber	0.9230769230769231	Keep Match 
atomic_radius	oxidationStates	1.0	Keep Match 
atomic_weight	density	0.46153846153846156	Keep Match 
band_gap	electronegativity	1.0	Keep Match 
boiling_point	yearDiscovered	0.7184466019417476	Keep Match 
brinell_hardness	electronicConfiguration	0.45	Keep Match 

Findings















Match N Merge solution based on existing features perform poorly when many cell values are the same for different columns and when there are a lot of numerical values. Although there are some likely matches, there are obvious incorrect mappings between the “best” mapping for Base_Column and Matched_Column.

Example 3 - Wine:*Datasets*

Data Set 2 Column Name	Example Value	Data Set 1 Column Name	Example Value
wine	1	fixed acidity	7.4
Alcohol	14.23	volatile acidity	0.7
Malic.acid	1.71	citric acid	0
Ash	2.43	residual sugar	1.9
Acid	15.6	chlorides	0.076
Mg	127	free sulfur dioxide	11
Phenols	2.8	total sulfur dioxide	34
Flavanoids	3.08	density	0.98
Nonflavanoid.phenols	0.28	pH	3.51
Proanth	2.29	sulphates	0.56
Color.int	5.64	alcohol	9.4
Hue	1.04	quality	5
OD	3.92		
Proline	1065		

Mappings

Match 'N Merge – IEOB 290 – Cox, Holbrook, Lim, Zhu

Base_Column	Matched_Column	Relative_Match_Accuracy	Change Column Mapping
Acid	total sulfur dioxide	1.0	Keep Match 
Alcohol	chlorides	0.8932584269662921	Keep Match 
Ash	total sulfur dioxide	0.7808988764044944	Keep Match 
Color.int	quality	0.5280898876404494	Keep Match 
Flavanoids	total sulfur dioxide	0.8876404494382022	Keep Match 
Hue	total sulfur dioxide	0.9101123595505618	Keep Match 
Malic.acid	total sulfur dioxide	0.8876404494382022	Keep Match 
Mg	total sulfur dioxide	0.5449438202247191	Keep Match 
Nonflavanoid.phenols	total sulfur dioxide	0.8539325842696629	Keep Match 
OD	total sulfur dioxide	0.8820224719101124	Keep Match 
Phenols	total sulfur dioxide	0.6348314606741573	Keep Match 
Proanth	total sulfur dioxide	0.9325842696629213	Keep Match 
Proline	chlorides	0.7584269662921348	Keep Match 
Wine	quality	1.0	Keep Match 

Findings

Match N Merge solution based on existing features perform poorly when all the data are numerical values. There are significant obvious differences between “best” mapping for Base_Column and Matched_Column.