OpenRefine

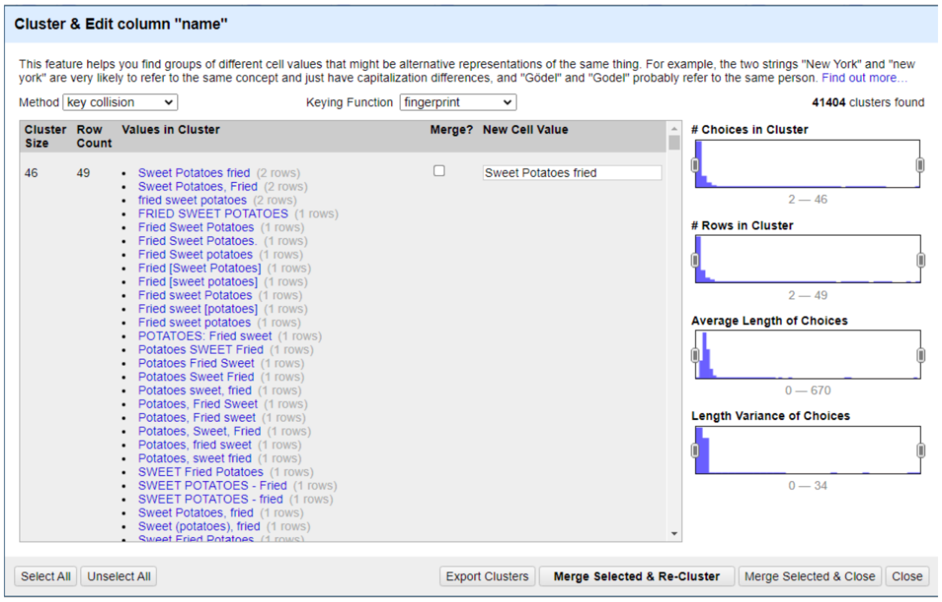
Yixuan Zhang

Introduction:

Data cleaning is a vital process that needs to be considered before any kind of data analysis, since not all the data available are pre-processed “clean” data. Text related data sometimes especially require extra steps of data cleaning, to make sure we can finally get a result as accurate as possible. OpenRefine is a handy tool we can use to clean our dataset. As a Java-based open-source software developed by Google, it is easy to find and easy to use. Besides text data, OpenRefine can also help us clean other types of data like numerical, boolean, or date. But I will focus on discussing the clustering function of OpenRefine since I feel this is most helpful with text cleaning.

Body:

One typical, simplified scenario we might be facing is, counting the occurrence of words or phrases. Let’s say if I want to count the word dinner from a data set. There are lots of possibilities of what the simple word “dinner” looks like in our data set: “dinner”, “DINNER”, “Dinner”, “dinnner” as a typo, or the word sticking with any punctuation like “dinner,”,“dinner!”,“Dinner.”. There are also some more complicated scenarios. Maybe instead of counting dinner, we want to count how many times “fried sweet potatoes” shows up in our data set. In this case, we might consider any phrase that truly means “fried sweet potatoes” to be counted as “fried sweet potatoes”. Like “Sweet Potatoes fired”, “Sweet Potatoes, Fried”, “Potatoes Fried Sweet”, maybe we should consider all of them representing the item that we are looking for.



Cluster & Edit column "name"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more…

Method: key collision   Keying Function: fingerprint   **41404** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
46	49	• Sweet Potatoes fried (2 rows) • Sweet Potatoes, Fried (2 rows) • fried sweet potatoes (2 rows) • FRIED SWEET POTATOES (1 rows) • Fried Sweet Potatoes (1 rows) • Fried Sweet Potatoes. (1 rows) • Fried Sweet potatoes (1 rows) • Fried [Sweet Potatoes] (1 rows) • Fried [sweet potatoes] (1 rows) • Fried sweet Potatoes (1 rows) • Fried sweet [potatoes] (1 rows) • Fried sweet potatoes (1 rows) • POTATOES: Fried sweet (1 rows) • Potatoes SWEET Fried (1 rows) • Potatoes Fried Sweet (1 rows) • Potatoes Sweet Fried (1 rows) • Potatoes sweet, fried (1 rows) • Potatoes, Fried Sweet (1 rows) • Potatoes, Fried sweet (1 rows) • Potatoes, Sweet, Fried (1 rows) • Potatoes, fried sweet (1 rows) • Potatoes, sweet fried (1 rows) • SWEET Fried Potatoes (1 rows) • SWEET POTATOES - Fried (1 rows) • SWEET POTATOES - fried (1 rows) • Sweet Potatoes, fried (1 rows) • Sweet (potatoes), fried (1 rows) • Sweet Fried Potatoes (1 rows)	☐	Sweet Potatoes fried

It is impossible to deal with such a problem manually when we are facing a large dataset. The clustering function of OpenRefine will be a powerful tool for us to use to solve such a problem. OpenRefine provides two clustering methods, key collision and nearest neighbor. There are six keying functions provided for key collision clustering and two method provided for nearest neighbor clustering.

For the key collision, fingerprint is definitely the best keying function to start with. Fingerprint as the keying function is like creating a bag-of-word, eliminating all the whitespace, punctuation and uppercase lowercase difference. Simple but powerful, this is the fastest way to normalize most similar items. If lots of similar items are actually permutations of certain groups of words, fingerprint keying function can identify them with minimal false positives. Since this method is generally powerful, using it to normalize lots of items first will decrease the running time of following clustering algorithms, because, we can filter out some items once we feel we might have already normalized all the related items, there will be no more items related to items that we already seen. Filtering out massive amounts of data of course will significantly decrease the running time.

N-gram fingerprint as the keying function can help us cluster items similar but not exactly spelling the same. This is a keying function that can help us detect typo or different spelling in different cultures like “color” and “colour” or incorrect spaces. Users can adjust the “n” value to create n-grams of “n” size.

There are four more keying functions for the key collision method: Metaphone3, Cologne, Daitch-Mokotoff, and Baider Morse. These four keying functions are phonetic clustering keying functions. These functions will cluster items based on how they “sound”. This type of keying functions good at identifying spelling errors. Group words with similar pronunciation sometimes work well. Metaphone3 is English based phonetic algorithm, cologne is German based phonetic algorithm, Daitch-Mokotoff and Baider-Morse are Slavic and Yiddish based phonetic algorithms. We do not exactly know what kind of language that the original dataset is based on. It might yield good results if we just apply all of them. Maybe one keying function will miss part of the result we want so running all of them will partially solve the problem.

Nearest neighbor will cluster items based on similarity and similarity threshold. With levenshtein distance as similarity measurement, openrefine will calculate the number of changes need to be made to let one item completely transform to another one. PMM as similarity measurement uses compression to determine similarity of two items, it is more effective if items that need to be clustered are relatively long. Nearest neighbor clustering is more flexible than key collision since in nature it tolerates and compares differences. Items with some amount of difference might be actually the same thing, that is the case nearest neighbor will be better than simple key collision. But the problem of the nearest neighbor is, since this type of algorithm calculates differences, the time complexity is O(N^2). Nearest neighbor will be significantly slower than key collision when the data set is large. OpenRefine itself comes with a partial solution, user can assign a parameter K to the algorithm, and OpenRefine divides the whole data set into blocks with a certain “blocking size”. This is like a mixture of kNN and key collision. Distance will only be calculated within these blocks, so the total number of calculations needed is reduced.

Summary

OpenRefine is a powerful tool for data cleaning, especially the cluster function can be a great tool to group and normalize text data. But there are two things that need to be noticed. First, data cleaning is a subjective task, even OpenRefine sometimes will give suggestions that fit our needs, the whole task cannot be automated and needs human involvement to make sure the result is desired. Second, clustering could be a long process. Running time is highly based on the size of the original data set. Clustering may also consume lots of memory and computing power.

Reference

OpenRefine. openrefine.org. Accessed 5 Nov. 2022.

OpenRefine User Manual | OpenRefine. 22 July 2022, docs.openrefine.org.

Phonetic Matching | Apache Solr Reference Guide 7.4. solr.apache.org/guide/7_4/phonetic-matching.html. Accessed 5 Nov. 2022.