MIDS W205

Lab #	10	Lab Title	OpenRefine -- Introduction
Related Module(s)	10	Goal	Get you started on OpenRefine and Edit Distance
Last Updated	9/27/15	Expected duration	60 minutes

Introduction

This Lab has 3 parts. The first two are involves using OpenRefine to clean up some data files. The third one involves calculating the Levenshtein distance between two strings.

OpenRefine is an open source tool for working with bad data. In this Lab we will give you a quick tour of how you can use it to clean data. This is purposely a short introduction to just get you introduced to the tool. If you want a more comprehensive tutorial you can follow any of the tutorials listed in the resources section. The second part of this lab involves understanding Levenshtein distance calculation using the dynamic programming method.

For the OpenRefine portion we will be using two data sets. One data set has earthquake data and one contains customer complaint data. You can access the dataset from the Github (both datasets are available if you clone or pull) or use the links in the text below. The links are also in the resources sections later in this document.

The first data set contains customer complaints; you can download that data set [here](#).

The second data set is the eq2015 data set which data about earthquakes of magnitude 3 or more during the first 6 months of 2015. You can download the Earthquake data set [here](#). You can fine an earthquake data attribute glossary [here](#).
OpenRefine is to a large extent menu driven. But it also allows you to use a language for doing certain types of transformations.

Open Refine

The basic idea in OpenRefine is that you think of exploring your data in terms of patterns, called facets. OpenRefine also has functions for doing transformations of data. These transformations can be expressed in the language GREL although there are a few other options as well. As an example, you can decide to create a new column based on an existing column but with a transformation applied to the data. GREL allows you to match in regular expressions, but also do common operations like trimming blanks, splitting strings etc. In addition it has control structures such as if-statements. You can even have OpenRefine call out to URLS and insert the results in a column. OpenRefine also support fuzzy matching (clustering) of attribute values. It will suggest values to merge and will let you choose which one you like to use.

Instructions, resources and prerequisites

For Step-1 and Step-2 install OpenRefine from here

For Step-3 make sure you have a working Python installation.

Below are a number of resources that are referred to in this lab, or may be of general interest to you during or after this lab.

Resource	What
http://openrefine.org/	This is where you download OpenRefine.
http://arcadiafalcone.net/GoogleRefineCheatSheets.pdf	A short description of OpenRefine commands.
http://enipedia.tudelft.nl/wiki/OpenRefine_Tutorial	Another tutorial on OpenRefine.
http://davidhuynh.net/spaces/nicar2011/tutorial.pdf	Another tutorial on OpenRefine.
http://schoolofdata.org/handbook/recipes/cleaningdatawithrefine/	Programming guide for the Spark Context object. Here you can find actions available on the Spark Contexts.
https://github.com/OpenRefine/OpenRefine/wiki/General-Refine-Expression-Language	GREL is the language used in OpenRefine for data refinements. This is a reference guide for the GREL language.
http://earthquake.usgs.gov/eart	Explanation of the Earthquake data.
https://pypi.python.org/pypi/python-Levenshtein/0.12.0	A Levenshtein module you can use to check your results in a Python shell.
https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth	A good quick read on some clustering methods.
Earth Quake Data set	https://github.com/UCBerkeleyISchool/w205labs-exercises/blob/master/lab_10 /dataset/eq2015.csv
Earthquake Data Glossary.	http://earthquake.usgs.gov/earthquakes/feed/v1.0/glossary.php#net
Customer Complaints Data.	https://github.com/UCBerkeleyISchool/w205labs-exercises/blob/master/lab_10/dataset/Consumer_Complaints.csv

Step-1. Wrangling the Customer Complaints Data

Uploading data
After you started OpenRefine you can pick a data set. For this first step choose the Customer Complaints Data set.

Google refine _A power tool for working with messy data._

Create Project										
Open Project										
Import Project										

Create a project by importing data. What kinds of data files can I import?

TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported. Support for other formats can be added with Google Refine extensions.

Get data from	Locate one or more files on your computer to upload:
This Computer	Choose Files Consumer_Complaints.csv
Web Addresses (URLs)	Next »
Clipboard	
Google Data	

Once the data is read you can inspect it. In this case it looks ok. But lets say that it would have been tab separated rather than comma separate, then OpenRefine would not have identified the structure correctly. You have the opportunity to look at the data here and confirm it is ok. In this case we think it looks good and we click the "Create Project" button.

Google refine _A power tool for working with messy data._

Create Project — Open Project — Import Project

« Start Over   Configure Parsing Options   Project name Consumer_Complaints csv   **Create Project »**

	Complaint ID	Product	Sub-product	Issue	Sub-issue	State	ZIP code	Submitted via	Date received	Date sent to company
1.	1354490	Debt collection		Cont'd attempts collect debt not owed	Debt is not mine	OH	44077	Web	04/30/2015	04/30/2015
2.	1355160	Student loan	Non-federal student loan	Dealing with my lender or servicer		NJ	8807	Web	04/30/2015	04/30/2015
3.	1355730	Credit reporting		Incorrect information on credit report	Account status	IL	60618	Web	04/30/2015	04/30/2015
4.	1355607	Debt collection	Other (phone, health club, etc.)	Disclosure verification of debt	Right to dispute notice not received	WA	98133	Web	04/30/2015	04/30/2015
5.	1354249	Bank account or service	Checking account	Problems caused by my funds being low		AL	35127	Web	04/30/2015	04/30/2015
6.	1354326	Bank account or service	Checking account	Account opening, closing, or management		TX	78575	Web	04/30/2015	04/30/2015
7.	1351925	Bank account or service	Checking account	Account opening, closing, or management		FL	34677	Web	04/29/2015	04/29/2015
8.	1352573	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt was paid	NV	89143	Web	04/29/2015	04/29/2015
9.	1354227	Debt collection	Medical	False statements or representation	Indicated committed crime not paying	FL	32792	Web	04/29/2015	04/30/2015
10.	1354200	Debt collection	Credit card	False statements or representation	Indicated committed crime not paying	AZ	85304	Web	04/29/2015	04/30/2015
11.	1352929	Debt collection	Other (phone, health club, etc.)	Cont'd attempts collect debt not owed	Debt is not mine	NC	27534	Web	04/29/2015	04/29/2015
12.	1354191	Bank account or service	Checking account	Problems caused by my funds being low		CA	90044	Web	04/29/2015	04/30/2015
13.	1354115	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt is not mine	TX	77449	Web	04/29/2015	04/29/2015

Parse data as

Character encoding [   ]   Update Preview

CSV / TSV / separator-based files

Line-based text files
Fixed-width field text files
PC-Axis text files
JSON files
RDF/N3 files
XML files
Open Document Format spreadsheets (.ods)
RDF/XML files

Columns are separated by
◉ commas (CSV)
◯ tabs (TSV)
◯ custom ,
Escape special characters with \

☐ Ignore first   0   line(s) at beginning of file
☑ Parse next   1   line(s) as column headers
☐ Discard initial   0   row(s) of data
☐ Load at most   0   row(s) of data

☑ Parse cell text into numbers, dates, …
☑ Quotation marks are
☑ Store blank rows
☑ Store blank cells as nulls

Version 2.5 [r2407]

Help
About

Also note that we specified that the first line should be parsed as column headers.

Creating a project

Next creating the project can take a little time since there are more that 300,000 lines in this file.

Once the project is created you can see that it has 384498 rows.

Check states with text facet

If you select text facet for state you will see a summary in the left column pane. It indicates you we have 62 different state value (?). Try to figure out why.

SUBMISSION 1: How many rows are missing value in the state column? Explain how you came up with the number?

Checking zip codes

Try the text facet on zip codes, what happens? You can see that there are 24748 different zip codes in this data set. Is that reasonable? Eye ball the data, does all zip codes look valid?

Now try the numeric facet. With the numeric facet the zip code attribute is treated is a numeric value. What would you say the scalar type is for zip codes, can be treated as a numeric attribute? Histogram below shows the distribution when the attribute is treated as numeric. By unchecking numeric you can get a list of row that are missing.

SUBMISSION 2: How many rows with missing zip codes do you have?
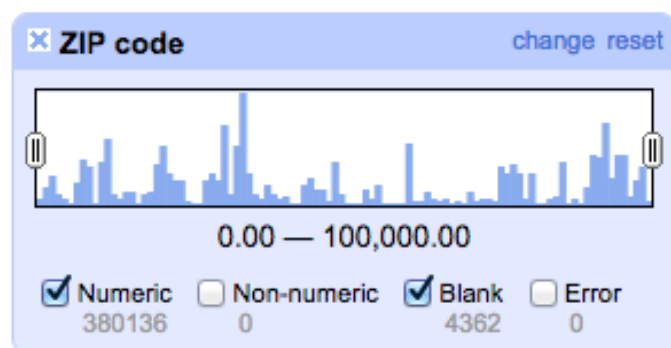
One way of filling in missing values is to take the previous value and use that to set subsequent empty cells. In OpenRefine it is called fill-down. Find a row that is blank. Apply fill down to the fill down by:

```
Edit Cell->Fill Down
```

What happened to the empty cell?

64.	1349391	Credit reporting		Credit reporting company's investigation	Problem with statement of dispute	TX	75181	Web	04/27/2015	04/27/2015
65.	1349369	Debt collection	Other (phone, health club, etc.)	Cont'd attempts collect debt not owed	Debt was paid	CO	80122	Web	04/27/2015	04/27/2015
66.	1347783	Bank account or service	Checking account	Account opening, closing, or management		CO	60008	Web	04/27/2015	04/29/2015
67.	1347685	Credit reporting		Incorrect information on credit report	Account status	KY	40219	Web	04/27/2015	04/27/2015
68.	1347775	Mortgage	Conventional fixed mortgage	Loan servicing, payments, escrow account		OH	43551	Web	04/27/2015	04/27/2015
69.	1347700	Debt collection		Cont'd attempts collect debt not owed	Debt is not mine	NY	12303	Phone	04/27/2015	04/28/2015
70.	1347687	Credit reporting		Incorrect information on credit report	Reinserted previously deleted info	VT	5468	Web	04/27/2015	04/27/2015

If you need to undo the operation, switch to the Undo/Redo tab. Select the previous state for the data. In this example I went back to state 2. As you can see in this screen shot row 151 has a missing zip code, so presumably the fill downs for Zip code and State has been un-done. Observe that the list in Undo/Redo can look different for you if you have been issuing more or different commands.

	Complaint ID	Product	Sub-product	Issue	Sub-issue	State	ZIP code	Submitted via	Date received	Date sent to con	Com
151.	1347504	Bank account or service	Cashing a check without an account	Making/receiving payments, sending money		CA		Web	04/26/2015	04/29/2015	Zions Bancorp
152.	1347486	Debt collection	Other (phone, health club, etc.)	Disclosure verification of debt	Right to dispute notice not received	VA	23238	Web	04/26/2015	04/26/2015	Enhance Recovery Company
153.	1347479	Debt collection	Credit card	Cont'd attempts collect debt not owed	Debt was paid	GA	30281	Web	04/26/2015	04/26/2015	HSBC
154.	1347475	Debt collection	Medical	Disclosure verification of debt	Not given enough info to verify debt	NJ	7712	Web	04/26/2015	04/26/2015	Common Financial Systems,
155.	1347474	Debt collection	Medical	Disclosure verification of debt	Not given enough info to verify debt	NJ	7712	Web	04/26/2015	04/26/2015	Senex Se Corp.
156.	1347492	Debt collection	Credit card	False statements or representation	Attempted to collect wrong amount	MA	1568	Web	04/26/2015	04/26/2015	Lustig, G Wilson, P
157.	1347490	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt is not mine	IL	60502	Web	04/26/2015	04/26/2015	ATG Gr LLC
158.	1347544	Credit reporting		Credit reporting company's investigation	Investigation took too long	NC	28574	Web	04/26/2015	04/26/2015	Equifax

Undo / Redo 2 — 384498 rows

- 0. Create project
- 1. Fill down 0 cells in column ZIP code
- 2. Fill down 0 cells in column ZIP code
- 3. Fill down 4362 cells in column ZIP code
- 4. Fill down 5377 cells in column State

● A3: Transforming zip code column. Lets create a new column called "ZipCode5" with all zip Codes that contain 5 digits preserved. All other rows should have the zip code 99999. (Technically speaking the 4-digit zip codes may be valid zip codes; we do this more to illustrate transformations).

Transformations are generally expressed in some language. OpenRefine supports a few alternative languages for transform, we will be using GREL. You can find a link to a language reference in the resources section. For this simple transformation we will be using an if-statement.

expression	result
if("internationalization".length() > 10, "big string", "small string")	big string
if(mod(37, 2) == 0, "even", "odd")	odd

For the Zip code column select:

        Edit Column -> Add column based on this column.

You will get the dialogue below. Insert the name of the new column and the expression:

        If(value.length() > 4, value, "99999")

This expression states that if the length of value is more than 4 insert value, otherwise insert the string "99999".

Look at the result, this it do what you wanted? What seems to be wrong with that? What happens if you instead insert a numeric value using the following expression?

        If(value.length() > 4, value, 99999)

Add column based on column ZIP code

New column name   ZipCode5

On error   ⦿ set to blank ○ store error ○ copy value from original column

Expression   Language [Google Refine Expression Language (GREL) ⬍]

`if(value.length()>4,value,"99999")`   No syntax error.

Preview   History   Starred   Help

row	value	if(value.length()>4,value,"99999")
1.	44077	44077
2.	8807	99999
3.	60618	60618
4.	98133	98133
5.	35127	35127
6.	78575	78575
7	24077	24077

Add column based on column ZIP code

New column name: ZipCode5

On error: ⦿ set to blank ○ store error ○ copy value from original column

Expression   Language: Google Refine Expression Language (GREL) ⇕

```
if(value.length()>4, value,99999)
```
No syntax error.

Preview   History   Starred   Help

row	value	if(value.length()>4, value,99999)
1.	44077	44077
2.	8807	99999
3.	60618	60618
4.	98133	98133
5.	35127	35127
6.	78575	78575
7.	24677	24677

OK   Cancel

You should have the same type for all cells in the created column. As an example the result should look something like the following.

All	Complaint ID	Product	Sub-product	Issue	Sub-issue	State	ZIP code	ZipCode5	Submitted via	Date received
66.	1347783	Bank account or service	Checking account	Account opening, closing, or management			60008	60008	Web	04/27/2015
91.	1348023	Money transfers	International money transfer	Other transaction issues				99999	Phone	04/27/2015
116.	1348625	Credit reporting		Credit reporting company's investigation	Inadequate help over the phone		777	99999	Web	04/27/2015
286.	1345142	Credit reporting		Unable to get credit report/credit score	Problem getting my free annual report		22043	22043	Web	04/24/2015
322.	1345678	Bank account or service	Checking account	Problems caused by my funds being low				99999	Referral	04/23/2015
329.	1343115	Credit reporting		Unable to get credit report/credit score	Problem getting my free annual report		19428	19428	Web	04/23/2015

SUBMISSION 3: If you consider all zip codes less than 99999 valid zip codes. How many valid and invalid zip codes do you have respectively?

Step-2. Cleaning up eq2015 Data.

Upload the data file eq2015.csv. After you checked that the data looks ok, create the project.

Google refine A power tool for working with messy data.

Create Project	« Start Over Configure Parsing Options												Project name eq2015 csv		Create Project »
Open Project	#	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	net	id	updated	place	type
Import Project	1.	2015-07-02T23:16:03.000Z	56.7152	-155.4884	5.4	3.6	ml				1.08	ak	ak11640129	2015-07-03T07:18:40.420Z	99km N of Chirikof Island, Alaska	earthquake
	2.	2015-07-02T22:40:35.240Z	36.8015	-97.7167	5	3	mb_lg		46	0.185	0.29	us	us10002n4d	2015-07-02T23:00:27.055Z	1km ESE of Medford, Oklahoma	earthquake
	3.	2015-07-02T22:31:28.190Z	-23.0587	-14.0431	10	4.8	mb		99	30.883	0.62	us	us10002n4f	2015-07-03T06:34:01.780Z	Southern Mid-Atlantic Ridge	earthquake
	4.	2015-07-02T19:38:39.760Z	32.981	-115.5813333	11.718	3.57	ml	67	63	0.0571	0.23	ci	ci37196663	2015-07-02T20:42:21.720Z	5km W of Brawley, California	earthquake
	5.	2015-07-02T19:22:44.570Z	-32.2014	-177.9748	35	5	mb		69	2.947	0.91	us	us10002n2x	2015-07-03T03:25:11.833Z	122km SE of L'Esperance Rock, New Zealand	earthquake
	6.	2015-07-02T19:06:28.220Z	-32.4952	-176.4412	37.72	4.9	mb		234	3.483	0.97	us	us10002n2l	2015-07-03T03:09:00.277Z	260km ESE of L'Esperance Rock, New Zealand	earthquake
	7.	2015-07-02T18:24:55.000Z	51.548	-175.7676	40.6	4.1	ml				0.75	ak	ak11639972	2015-07-03T02:27:38.059Z	71km ESE of Adak, Alaska	earthquake
	8.	2015-07-02T15:06:46.000Z	55.9723	-156.1441	41.2	3.3	ml				0.86	ak	ak11639884	2015-07-02T23:09:14.453Z	36km WNW of Chirikof Island, Alaska	earthquake
	9.	2015-07-02T15:01:10.650Z	-5.9841	147.335	82.79	5.3	mb		69	3.403	0.7	us	us10002n0i	2015-07-02T21:12:34.405Z	90km NNE of Lae, Papua New Guinea	earthquake
	10.	2015-07-02T13:36:20.770Z	11.8668	142.4645	45	4.6	mb		137	22.475	0.62	us	us10002n0a	2015-07-02T21:38:58.880Z	285km WSW of Merizo Village, Guam	earthquake
	11.	2015-07-02T07:26:49.470Z	-34.6926	-16.1809	10	5.7	mb		61	20.877	1	us	us10002mzh	2015-07-02T15:29:19.673Z	Southern Mid-Atlantic Ridge	earthquake
	12.	2015-07-02T07:18:24.270Z	34.4411	73.7036	10	5.4	mb		62	0.867	1.09	us	us10002mzg	2015-07-02T16:19:42.789Z	22km ENE of Muzaffarabad, Pakistan	earthquake
	13.	2015-07-02T07:04:32.000Z	-7.4327	120.222	413.11	4.9	mb		57	2.326	0.97	us	us10002mzb	2015-07-02T15:07:02.868Z	95km NNW of Congkar, Indonesia	earthquake
	14.	2015-07-02T07:03:23.790Z	53.1083	173.0287	23.49	4.8	mb		130	0.752	0.91	us	us10002mzc	2015-07-02T22:27:31.356Z	31km NNW of Attu Station, Alaska	earthquake
	15.	2015-07-02T06:05:46.840Z	38.5198	141.8613	61.57	4.6	mb		133	1.065	1.01	us	us10002mym	2015-07-02T12:55:55.168Z	50km ENE of Ishinomaki, Japan	earthquake
	16.	2015-07-02T04:03:45.700Z	19.0249	-66.365	40	3.2	Md	14	230.4	0.56593863	0.31	pr	pr15183001	2015-07-02T12:06:24.867Z	61km N of Brenas, Puerto Rico	earthquake
	17.	2015-07-02T03:37:41.200Z	18.5076	-68.657	135	3.4	Md	9	342	1.12379242	0.1	pr	pr15183002	2015-07-02T11:40:16.262Z	8km NNE of San Rafael del Yuma, Dominican Republic	earthquake
	18.	2015-07-02T02:18:27.000Z	60.9657	-147.4064	23.8	3.7	ml				0.55	ak	ak11639524	2015-07-02T03:01:09.571Z	60km WSW of Valdez, Alaska	earthquake
	19.	2015-07-02T01:56:11.220Z	27.749	85.2992	10	4.2	mb		120	0.039	0.96	us	us10002mxq	2015-07-02T09:42:31.008Z	5km NNW of Kathmandu, Nepal	earthquake
	20.	2015-07-02T01:35:49.100Z	19.6	-67.8811	44	3.4	Md	7	327.6	1.34118472	0.18	pr	pr15183000	2015-07-02T09:38:31.557Z	125km NNE of Punta Cana, Dominican Republic	earthquake
	21.	2015-07-01T23:20:13.000Z	56.1872	-153.5558	12	4.3	ml				0.71	ak	ak11639423	2015-07-02T21:15:08.040Z	118km S of Larsen Bay, Alaska	earthquake
	22.	2015-07-01T21:59:53.860Z	36.0803	-97.4545	2.23	3	mb_lg		65	0.283	0.29	us	us10002mx3	2015-07-02T21:52.713Z	18km NE of Crescent, Oklahoma	earthquake

Parse data as   Character encoding [   ]   Update Preview

CSV / TSV / separator-based files   Columns are separated by   ☐ Ignore   0 line(s) at beginning
Line-based text files   ● commas (CSV)   first   of file
Fixed-width field text files   ○ tabs (TSV)   ☑ Parse   1 line(s) as column
                                             ○ custom ,   next   headers
PC-Axis text files   Escape special characters with \   ☐ Discard 0 row(s) of data
JSON files   initial
RDF/N3 files   ☐ Load at 0 row(s) of data
                                                                                           most

As you can see the column "nst" is missing quite a few values. Look up the "nst" attribute in the glossary. What would happen if we just ignored row with missing values? Is there an obvious strategy for filling in the missing values? What would you suggest we do with the column?

Next we like to extract an approximate area from the "place" column. We would like to have a State or Country, and we like to store that information in a separate column we like to call "location".

As we review the "place" column we notice that the cell seems to be consisting two commas separated components. The components are one direction and distance; and a general location.

Select the command:

```
Edit Column -> Add Column based on this column
```

You should see a pop-up dialogue box looking as follows. We typed in the column name of the new column "location"
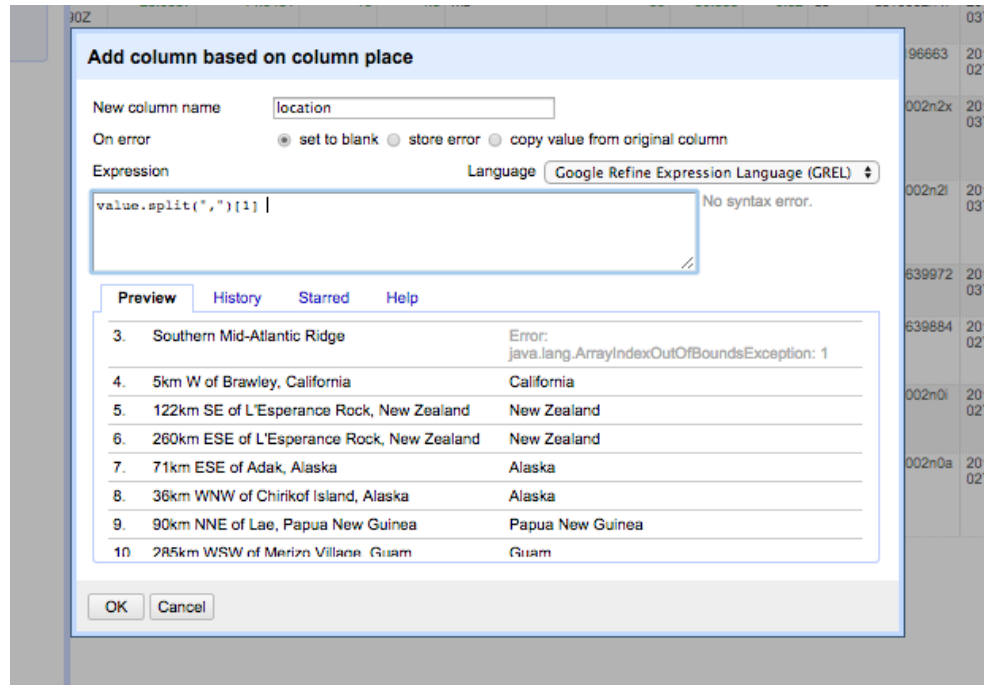
Since we noticed the cells have two comma-separated components, and the second is a location we defined the following expression.

```
value.split(",")[1]
```

But as you probably notice that did not work so well. In fact, not all cells have the two components. If you look at the data more closely it seems that is the place was in offshore the location component was missing. So we modify the expression as follows.

```
if(value.split(",").length() < 2 ,"Offshore",
value.split(",")[1])
```

If only has one component we assume it was Offshore and put that value in the location column.

Check the resulting data. Does it seem reasonable, or are more adjustments needed.

Check the value by using a text facet on the column. You may notice that there appears to be multiple strings that look like Alaska, but appears to be misspelled.

Clustering may help us detect more of these kinds of situations. Run clustering by pushing the Cluster button on the facet or use the column pop-up menu and select:

```
Edit Cell->Cluster and edit
```

Try key collision what do you see? Try nearest neighbor, Levenshtein what do you see? You can change the parameters such as radius and Block Chars. Radius provides a threshold for how close the strings should be to be considered representing the same entity. The Block Char parameter may behave a little counter intuitive. Blocking defines blocks within which the string distance method is applied. It helps with scalability, as we will not compare strings cross the whole data set. OpenRefine blocking parameter defines the size of a substring S, such that all strings that share S will be in a common block. So a smaller S will likely result in bigger blocks and more computation required.
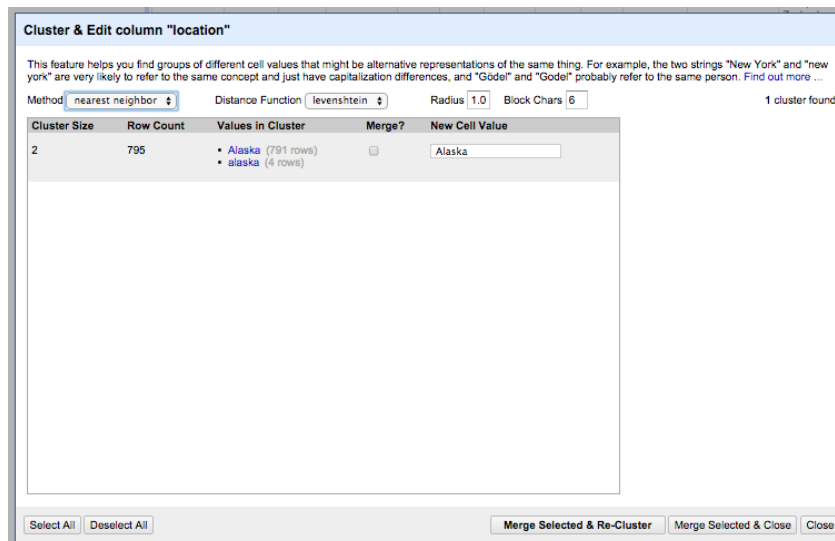
Change radius to 2.0, what happens?

Cluster & Edit column "location"

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 [nearest neighbor ⬍]   Distance Function [levenshtein ⬍]   Radius [2.0]   Block Chars [6]   2 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	795	• Alaska (791 rows) • alaska (4 rows)	☐	Alaska
2	85	• California (84 rows) • California (1 rows)	☐	California

 # Rows in Cluster

80 — 800

Average Length of Choices

7 — 11

[Select All] [Deselect All]   [Merge Selected & Re-Cluster] [Merge Selected & Close] [Close]

> **SUBMISSION 5: Change radius to 3.0, what happens? Do you want to merge any of the resulting matches?**

Change block size to 2 and run the clustering.

> **SUBMISSION 6: Change block size to 2. Give 2 examples of new clusters that may be worthwhile merging.**

You can try different parameters to see if you can catch the issues you see. If not you can also note that there are a few misspellings of Alaska that only occurs once. Hence, it is doable to go in and edit them by hand.

Cluster & Edit column "location"

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 [nearest neighbor ⬍]   Distance Function [levenshtein ⬍]   Radius [3.0] Block Chars [6]   2 clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	805	• Indonesia (797 rows) • Micronesia (8 rows)	☐	Indonesia
2	61	• Tajikistan (36 rows) • Pakistan (25 rows)	☐	Tajikistan

 # Rows in Cluster

60 — 810

Average Length of Choices

10 — 10.5

Length Variance of Choices

0.5 — 1

If you review the facet you will see that there still are values that seems wrong and that were not caught. There are single values, so the easiest if probably just to go in an edit those cells and fix the values. You can access the values by clicking on the facet widget.

Consider if you had a very large data set and you wanted to automate the cleaning. Expecting manual editing would not be feasible.

The place column strings are significantly longer that the strings for location. Try to do nearest neighbor clustering on "place' column.

SUBMISSION 7: Explain in words what happends when you cluster place, and why you think that happened? What additional functionality could OpenRefine provide to possibly deal with the situation.

Hint…it takes a long time; you may in fact want to cancel the run.

Step-3 Levenshtein Distance

Introduction
In this lab we will go over a simple example of Levenshtein distance calculation. We will then ask you to calculate the distance for two strings gumbarrel" and "gunbarell". We will point you to a simple implementation of the Levenshtein distance that you can use to check your result.

Installing Levenshtein python module
The following steps will just clone and build a Python Levenshtein module in a directory. It does not fully install the module. But you can use it to run a distance function from your shell to check your results.

```
$ git clone https://github.com/ztane/python-Levenshtein/
$ cd python-Levenshtein/
$ python setup.py build
$ cd Levenshtein/
$ python
>>> from  Levenshtein import *
>>> distance("hej","hei")
1
>>> distance("monthgomery st","montgomery street")
5
```

Example: Levenshtein Calculation
Lets step through the calculation of distance between the words LOYOLA and LAJOLLA. We will denote a cell with the d[i,j], where i is the row and j is the column. The first column and row indicates the indices we will be using.

As a reminder the algorithms is as follows:

Denote the column by c and row by r. We have n rows and m columns.
d[i,j] denotes the value on row i and columns j.
cost $[i,j] = 1$ if $c[i]\ !=r[j]$
cost $[i,j] = 0$ if $c[i] == r[j]$
d[i,j] is to be set to the minimum of: d[i-1,j]+1 or d[i,j-1]+1 or d[i-1, j-1]+cost[I,j]
Distance is found in the resulting value d[n,m]

We first set up the matrix. The blue row and column just contains the i and j values. We then insert values `0-m` in first row `i==1` and `0-n` in the column `j==1`.

		1	2	3	4	5	6	7
		L	O	Y	O	L	A
1		0	1	2	3	4	5	6
2	L	1						
3	A	2						
4	J	3						
5	O	4						
6	L	5						
7	L	6						
8	A	7						

Lets calculate the d[i,2]. Meaning the value for each row in the column 2.

```
d[2,2], cost is 0, minimum is d[1,1]+0=>0
d[3,2], cost is 1, minimum is d[2,2]+1=>1
d[4,2], cost is 1, minimum is d[3,2]+1=>2
d[5,2], cost is 1, minimum is d[4,2]+1=>3
d[6,2], cost is 0, minimum is d[5,1]+0=>4, or d[5,2]+1
d[7,2], cost is 0, minimum is d[6,2]+0=>5, or d[6,2]+1
d[8,2], cost is 1, minimum is d[7,2]+1=>6
```

		1	2	3	4	5	6	7
		L	O	Y	O	L	A
1		0	1	2	3	4	5	6
2	L	1	0					
3	A	2	1					
4	J	3	2					
5	O	4	3					
6	L	5	4					
7	L	6	5					
8	A	7	6					

Lets calculate the d[i,3]. Meaning the value for each row in the column 3.

```
d[2,3], cost is 1, minimum is d[2,2]+1=>1
d[3,3], cost is 1, minimum is d[2,2]+1=>1
d[4,3], cost is 1, minimum is d[3,2]+1=>2, or d[3,3]+1
d[5,3], cost is 0, minimum is d[4,2]+0=>2
d[6,3], cost is 1, minimum is d[5,3]+1=>3
d[7,3], cost is 1, minimum is d[6,2]+1=>4, or d[6,3]+1
d[8,3], cost is 1, minimum is d[7,2]+1=>4
```

		1	2	3	4	5	6	7
		L	O	Y	O	L	A	
1		0	1	2	3	4	5	6
2	L	1	0	1				
3	A	2	1	1				
4	J	3	2	2				
5	O	4	3	2				
6	L	5	3	3				
7	L	6	3	4				
8	A	7	4	4				

If you do the same thing for the remaining columns we will get the following matrix. You see the calculated edit distance in the cell `d[8,7]`.

		1	2	3	4	5	6	7
		L	O	Y	O	L	A	
1		0	1	2	3	4	5	6
2	L	1	0	1	2	3	4	5
3	A	2	1	1	2	3	4	4
4	J	3	2	2	2	3	4	5
5	O	4	3	2	3	2	3	4
6	L	5	3	3	3	3	2	3
7	L	6	3	4	4	4	3	3
8	A	7	4	4	5	5	4	**3**

If you use the Levenshtein function to check the result you will see the following result.

```
>>> distance("loyola","lajolla")
3
```

So we are assuming we got the calculation right.

Calculation: gumbarrel v.s gunbarell

Now calculate the edit distance between the words: "gumbarrel" and "gunbarell". After you done that use the python Levenshtein function to check you result.

		1	2	3	4	5	6	7	8	9	10
		G	U	M	B	A	R	R	E	L
1		0	1	2	3	4	5	6	7	8	9
2	G	1									
3	U	2									
4	N	3									
5	B	4									
6	A	5									
7	R	6									
8	E	7									
9	L	8									
10	L	9									

SUBMISSION 8: submit a representation of the resulting matrix from the leveshtein edit distance calculation. The resulting value should be correct.