W205 Lab 10

Yiran Sheng
UCB
yiran@ischool.berkeley.edu

1. Submission 1: Missing Values in States

5377 rows have missing values in State column, indicated by the following choice counts:

```
(blank) 5377
```

2. Submission 2: Missing Values in Zipcode

There are 4362 (blank) / missing values rows for column Zipcode.

3. Submission 3: Valid Zipcodes

Using the following GREL for custom text facet:

```
if(value>0, if(value < 99999, "Valid", "Invalid"), "blank")
```

We found out:

```
blank 4362
Invalid 34961
Valid 345175
```

There are 345175 valid zipcodes and 39323 invalid ones, out of which 4362 are missing / blank.

4. Submission 5: Change the radius to 3.0

We observe the following clusters:

```
2    795
Alaska(791 rows)
alaska(4 rows)

2    805
Indonesia(797 rows)
```

```
Micronesia(8 rows)

2    61
Tajikistan(36 rows)
Pakistan(25 rows)

2    85
California(84 rows)
Cailfornia(1 rows)
```

And merge the following two clusters:

```
2    795
Alaska(791 rows)
alaska(4 rows)

2    85
California(84 rows)
Cailfornia(1 rows)
```

5. Submission 6: Change the block size to 2

The following two clusters might be good candidates for merging:

```
Alaska(795 rows)
Alaka(1 rows)
Alska(1 rows)
Alaa(1 rows)
Alaksa(1 rows)

Canada(33 rows)
Candaa(2 rows)
Cnaada(1 rows)
```

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

Clustering on place column takes significantly longer. This is because levenshtein distance algorithm has a $O(m \times n)$ time complexity, where m and n are length of the strings compared for each pair.

One possible way of dealing with it is to split longer strings into tokens(words) and run the clustering based on levenshtein distance of tokens (i.e. treat each

token as a single alphabet).

7. Submission 8: Submit a representation of the resulting matrix from the Leveshtein edit distance calculation. The resulting value should be correct.

		g	**u**	**n**	**b**	**a**	**r**	**e**	**l**	**l**
	0	1	2	3	4	5	6	7	8	9
g	1	0	1	2	3	4	5	6	7	8
u	2	1	0	1	2	3	4	5	6	7
m	3	2	1	1	2	3	4	5	6	7
b	4	3	2	2	1	2	3	4	5	6
a	5	4	3	3	2	1	2	3	4	5
r	6	5	4	4	3	2	1	2	3	4
r	7	6	5	5	4	3	2	2	3	4
e	8	7	6	6	5	4	3	2	3	4
l	9	8	7	7	6	5	4	3	2	3