Cleaning text data with fuzzywuzzy

Fourth in a series on scikit-learn and GeoPandas

Posted by Michelle Fullwood on May 20, 2015

Previous articles in this series:

- 1. Motivations and Methods (/code-blog/2015/04/24/sgmap/)
- 2. Obtaining OpenStreetMap data (/code-blog/2015/04/27/osm-data/)
- 3. Manipulating geodata with GeoPandas (/code-blog/2015/04/29/geopandas-manipulation/)

In this fourth article, we'll look at how to clean text data with the `fuzzywuzzy` library (https://github.com/seatgeek/fuzzywuzzy) from SeatGeek.

Use case

The road data I downloaded from OpenStreetMap had some obvious errors among the street names, mostly misspellings. For example, there was "Aljuneid Avenue 1" when the correct spelling is "Aljunied". This was problematic since (1) misspellings make our ultimate goal of classification difficult, and (2) we can't unify roads that share a name, like "Aljunied Avenue 2", giving us more work to do. I could have gone through the list manually, but it would have been time-consuming.

My solution was to get a better list from outside OpenStreetMap, and match the less correct road names to it using a library called `fuzzywuzzy`, for fuzzy string matching. Here's how it works:

```
>>> from fuzzywuzzy import process

>>> correct_roadnames = ["Aljunied Avenue 1", "Aljunied Avenue 2", ...
>>> process.extractOne("Aljuneid Avenue 1", correct_roadnames)
('Aljunied Avenue 1', 94)
```

The first element of the return tuple indicates the closest match in the reference list, and the second number is a score showing how close it is. An exact match is 100.

Sometimes, when the correct road name wasn't in the reference set either, the score would be pretty low – which is as it should be!

```
>>> process.extractOne('Elgin Bridge', correct_roadnames)
('Jalan Woodbridge', 64)

>>> process.extractOne('Cantonment Close', correct_roadnames)
('Jago Close', 85)
```

I decided to set a boundary of 90, above which I would accept the solution `fuzzywuzzy` came up with automatically, and below which I would just manually review the road name to decide what it should be.

Using fuzzywuzzy in Pandas

So what we want is to apply `process.extractOne()` to the roadname column of our dataframe. This was my first attempt:

```python
def correct_road(roadname):
    new_name, score = process.extractOne(roadname, correct_roadnames)
    if score < 90:
        return roadname, score
    else:
        return new_name, score

df['corrected'], df['score'] = zip(*df['name'].apply(correct_road))
```

It took *forever*! The reason is that `extractOne` is doing a pairwise comparison of all the names in the dataframe with the correct names in the canonical list. But when the name is correct, which is the majority of the time, we don’t actually need to do all these pairwise comparisons. So I did a preliminary test to see if the roadname is in the list of correct names, and that cut down on time considerably.

```python
def correct_road(roadname):
    if roadname in correct_roadnames:  # might want to make this a dict
        return roadname, 100

    new_name, score = process.extractOne(roadname, correct_roadnames)
    if score < 90:
        return roadname, score
    else:
        return new_name, score

df['corrected'], df['score'] = zip(*df['name'].apply(correct_road))
```

You can put in other checks, for example I would only accept a >90 match if the number of words was the same. Whatever makes sense for your particular use case.

Conclusion

After getting the corrected dataframe, I went into OpenStreetMap and edited most of the incorrect road names, so hopefully Singapore street names are mostly correctly spelled now. The `fuzzywuzzy` library was a big help in cutting down the number of roads I needed to manually review, so I recommend adding it to your data cleaning arsenal.

← **PREVIOUS POST (/CODE-BLOG//2015/04/29/GEOPANDAS-MANIPULATION/)**

NEXT POST → (/CODE-BLOG//2015/06/18/CLASSIFYING-ROADS/)

● (/code-blog/feed.xml) ● (https://twitter.com/michelleful)

● (https://github.com/michelleful)