1. Extracting City and Country Information from News Headlines

October 10, 2019

1 Extracting City and Country Information from News Headlines

1.1 Parsing the News Headlines

Objective: Find any city and/or country names mentioned in each of the news headlines. We will use these names to find geographic locations of the headlines in the next section.

Workflow: 1. Load in the headline data and examine it for any data quality issues. - Use any library/data structure to read in the headlines - Read through some of the headlines and identify potential problems 2. Using regular expressions and the cities and countries within the geonamescache library, match any cities/countries within each headline. - Make sure to normalize headlines and city/country names by removing accent marks. This can be done with the unidecode library. - Watch out for multiple cities in a headline and matches on short words! We want the match to be on the entire city – for example San Marino – and not a partial match – San. 3. Put the extracted data into a pandas DataFrame with three columns: headline, city, country. 4. Make sure there were no issues with the extraction by sampling some of the headlines and examining the city and country names. - One method for finding problems is to look for the most common names and see if there are any issues. 5. Once you are confident you've found all the cities/countries in each headline, save the DataFrame for the next part.

Deliverable:

The deliverable is a Jupyter Notebook documenting your workflow as you take the headlines.txt file, extract the city/country names, and put the results into a Pandas DataFrame. This DataFrame will allow us to quickly perform analysis on the headlines and geographic data that we will find in the next part.

1.1.1 Read in the data

We can read in all the headlines as a list and then strip out the newline marks.

[1]:
```
data = open("../data/headlines.txt", "r").readlines()
data[:4]
```

[1]:
```
['Zika Outbreak Hits Miami\n',
 'Could Zika Reach New York City?\n',
 'First Case of Zika in Miami Beach\n',
 'Mystery Virus Spreads in Recife, Brazil\n']
```

[2]:
```
# Strip out newline from each headline.
data = [headline.strip() for headline in data]
data[:4]
```

[2]:
```
['Zika Outbreak Hits Miami',
 'Could Zika Reach New York City?',
 'First Case of Zika in Miami Beach',
 'Mystery Virus Spreads in Recife, Brazil']
```

2 Cities and Countries from GeonamesCache

Now we'll create a list of city names and country names using the `geonamescache` Python library.

[3]:
```
import geonamescache

gc = geonamescache.GeonamesCache()
countries = [country["name"] for country in gc.get_countries().values()]
countries[:4]
```

[3]:
```
['Andorra', 'United Arab Emirates', 'Afghanistan', 'Antigua and Barbuda']
```

We can do the same with cities.

[4]:
```
cities = [city['name'] for city in gc.get_cities().values()]
cities[:4]
```

[4]:
```
['Andorra la Vella', 'Umm Al Quwain City', 'Ras Al Khaimah City', 'Zayed City']
```

2.1 Duplicate Cities

There are cities in `geonamescache` that are recorded more than once in different countries (or even multiple times in the same country). We'll have to figure out how to handle this later.

[5]:
```
from collections import Counter

city_counts = Counter(cities)
city_counts.most_common(10)
```

[5]:
```
[('Springfield', 8),
 ('San Pedro', 7),
 ('Richmond', 7),
 ('San Fernando', 7),
 ('Mercedes', 6),
 ('La Paz', 6),
 ('Victoria', 6),
 ('San Francisco', 6),
 ('Auburn', 6),
```

```
 ('Santa Cruz', 6)]
```

One city is even mentioned 8 times! This could cause an issue - we'll handle the duplicate cities in the second part of this project when we match cities to locations.

2.2 Removing Accent Marks

We need to remove the accent marks from the lists of countries and cities. For this we will use the `unidecode` library. (Method from this Stack Overflow answer.) For the cities and the countries from geonamescache, we will map the unaccented name to the accented name.

[6]:
```
import unidecode

country_accent_mapping = {
    unidecode.unidecode(country): country for country in countries
}
city_accent_mapping = {unidecode.unidecode(city): city for city in cities}
city_accent_mapping["Asmar"]
```

[6]:
```
'Āsmār'
```

We see that the accented city names have been mapped to their unaccented versions. Let's also remove the accent marks from the headlines so we can match on the unaccented version (in the `keys` of the mapping.)

[7]:
```
data = [unidecode.unidecode(headline) for headline in data]
data[-4:]
```

[7]:
```
['More Zika patients reported in Indang',
 'Suva authorities confirmed the spread of Rotavirus',
 'More Zika patients reported in Bella Vista',
 'Zika Outbreak in Wichita Falls']
```

Now, we can look for the unaccented city and country names in the headlines.

3 Searching for Cities and Countries

Next, we'll search each headline for any cities and/or countries. To do this, we use regular expressions created from the unaccented cities and countries.

[8]:
```
# Create list of cities and countries
unaccented_cities = list(city_accent_mapping.keys())
unaccented_countries = set(country_accent_mapping.keys())

print(f"There are {len(unaccented_cities)} cities to look through.")
print(f"There are {len(unaccented_countries)} countries to look through.")
```

```
There are 23022 cities to look through.
There are 252 countries to look through.
```

3.1 Regular Expressions

The two regular expressions for searching the headlines will consist of all the cities and all the countries. We want to be careful about two things:

1. Match entire words. The solution for this is use the `\b` tag as explained in this answer. Basically, we have to surround our regular expression words with `\b` like so `\bcity_name\b`.
2. Find the entire city. For some cities, like "San Jose", we will end up matching "San" because there is a city in the list with "San" and regular expressions are greedy, returning the first complete match. Therefore, we need to sort the lists of cities and countries by descending length before creating the regular expressions. This ensures we match the longest city and country instead of getting a partial match.

3.1.1 Partial Match Issue

[9]:
```
import re

problem_city = 'San Jose'
re.search('\\bSan\\b|\\bSan Jose\\b', problem_city)
```

[9]:
```
<re.Match object; span=(0, 3), match='San'>
```

Here we see the second problem. We've matched only `San` instead of the entire city name. To correct this, we change the ordering of the regular expression.

[10]:
```
re.search('\\bSan Jose\\b|\\bSan\\b', problem_city)
```

[10]:
```
<re.Match object; span=(0, 8), match='San Jose'>
```

Issue solved! If we first sort the cities from longest to shortest, we will be sure to match the entire city name.

3.1.2 Sort Cities and Countries by Length

We can sort the cities and countries first.

[11]:
```
unaccented_cities = sorted(unaccented_cities, key=lambda x: len(x),␣
↪reverse=True)
unaccented_cities[:2]
```

[11]:
```
['Chak Two Hundred Forty-nine Thal Development Authority',
 'Dolores Hidalgo Cuna de la Independencia Nacional']
```

[12]:
```
unaccented_countries = sorted(unaccented_countries, key=lambda x: len(x),␣
↪reverse=True)
unaccented_countries[:2]
```

[12]:
```
['South Georgia and the South Sandwich Islands',
 'United States Minor Outlying Islands']
```

These lists are now in order from longest to shortest names.

3.1.3 Constructing the Regular Expressions

We construct the regular expressions by joining together the list of strings. The words are separated with a `|` for the or symbol in a regular expression. We also use the `\b` tag to make sure to match on entire words (beginning and end.)

[13]:
```
city_regex = r'\b|\b'.join(unaccented_cities)
city_regex[1500:1800]
```

[13]:
```
'-Baume\\b|\\bTamuning-Tumon-Harmon Village\\b|\\bTultitlan de Mariano
Escobedo\\b|\\bSan Bernardino Tlaxcalancingo\\b|\\bSan Francisco
Tlalcilalcalpan\\b|\\bFraccionamiento Ciudad Olmeca\\b|\\bPresidencia Roque
Saenz Pena\\b|\\bZurich (Kreis 11) / Oerlikon\\b|\\bSan Fernando de Monte
Cristi\\b|\\bPuerto Francisco de '
```

Let's test out the city regex.

[14]:
```
import numpy as np

np.random.seed(50)

test_headlines = np.random.choice(data, 10)

for test_headline in test_headlines:
    print(test_headline)
    match = re.search(city_regex, test_headline)
    if match:
        print(match.group(0), "\n")
```

```
More Zika patients reported in Custodia
Custodia

Tokyo Encounters Severe Symptoms of Meningitis
Tokyo

Zika Troubles come to Kampong Cham
Kampong Cham

19 new Zika Cases in Sengkang
Sengkang

Mumbai's Health Minister warns of more Zika cases
Mumbai

Varicella re-emerges in Lagos
Lagos
```

```
Mumbai's Health Minister warns of more Zika cases
Mumbai

Milwaukee authorities confirmed the spread of Rhinovirus
Milwaukee

Zika cases concern Charlotte residents
Charlotte

Four cases of Zika in Hidalgo County
Hidalgo
```

That seems to work well. We do have an issue with the last headline, but there will always be some data quality problems.

Let's make a regular expression for the countries.

[15]:
```
country_regex = r"\b|\b".join(unaccented_countries)
country_regex[:100]
```

[15]:
```
'South Georgia and the South Sandwich Islands\\b|\\bUnited States Minor Outlying
Islands\\b|\\bBonaire, S'
```

[16]:
```
np.random.seed(100)
test_headlines = np.random.choice(data, 10)

for test_headline in test_headlines:
    print(test_headline)
    match = re.search(country_regex, test_headline)
    if match:
        print(match.group(0), "\n")
```

```
Longwood volunteers spreading Zika awareness
More Zika cases in Soyapango
Spike of Dengue Cases in Stockholm
Case of Measles Reported in Vancouver
Zika arrives in Belmopan
Outbreak of Zika in Colombo
Zika symptoms spotted in Arlington
Malaria re-emerges in Boise
Southampton Patient in Critical Condition after Contracting Tuberculosis
Manassas Encounters Severe Symptoms of Measles
```

No country matches! We might not have many countries to work with in this project. Let's test both the city and country regex on a headline with a city and a country.

[17]:
```
test_headline = data[3]
print(test_headline)
```

```
print(re.search(city_regex, test_headline).group(0))
print(re.search(country_regex, test_headline).group(0))
```

```
Mystery Virus Spreads in Recife, Brazil
Recife
Brazil
```

For any matches, we can look up the accented version in our mapping.

[18]:
```
print(city_accent_mapping["Recife"])
print(country_accent_mapping["Brazil"])
```

```
Recife
Brazil
```

Neither of these have accents.

3.1.4 City and Country Regular Expression Function

Let's encapsulate the logic to find city and country names into a function.

[19]:
```
def find_city_and_country_in_headline(headline):
    """
    Find the city(s) and/or country(s) in a text headline.

    :param headline: string for headline

    :return dict: a dictionary mapping the headline to city(s) and/or countries.
    """
    city_match = re.search(city_regex, headline)
    country_match = re.search(country_regex, headline)
    cities = None if not city_match else city_match.group(0)
    countries = None if not country_match else country_match.group(0)
    return dict(headline=headline, countries=countries, cities=cities)
```

Let's test this out.

[20]:
```
find_city_and_country_in_headline(data[3])
```

[20]:
```
{'headline': 'Mystery Virus Spreads in Recife, Brazil',
 'countries': 'Brazil',
 'cities': 'Recife'}
```

[21]:
```
find_city_and_country_in_headline(data[1])
```

[21]:
```
{'headline': 'Could Zika Reach New York City?',
 'countries': None,
 'cities': 'New York City'}
```

Things look pretty good at this point. Let's move on to using this for all headlines.

3.2 Apply Regular Expression to All Headlines

Now we apply this function to all headlines. We'll end up with a list containing all the headlines and and cities or countries in the headlines.

[22]:
```
headline_cities_and_countries = [
    find_city_and_country_in_headline(headline) for headline in data
]
headline_cities_and_countries[-10:]
```

[22]:
```
[{'headline': 'Authorities are Worried about the Spread of Varicella in Clovis',
  'countries': None,
  'cities': 'Clovis'},
 {'headline': 'More Zika patients reported in Fort Worth',
  'countries': None,
  'cities': 'Fort Worth'},
 {'headline': 'Zika symptoms spotted in Boynton Beach',
  'countries': None,
  'cities': 'Boynton Beach'},
 {'headline': 'Outbreak of Zika in Portoviejo',
  'countries': None,
  'cities': 'Portoviejo'},
 {'headline': 'Influenza Exposure in Muscat',
  'countries': None,
  'cities': 'Muscat'},
 {'headline': 'Rumors about Rabies spreading in Jerusalem have been refuted',
  'countries': None,
  'cities': 'Jerusalem'},
 {'headline': 'More Zika patients reported in Indang',
  'countries': None,
  'cities': 'Indang'},
 {'headline': 'Suva authorities confirmed the spread of Rotavirus',
  'countries': None,
  'cities': 'Suva'},
 {'headline': 'More Zika patients reported in Bella Vista',
  'countries': None,
  'cities': 'Bella Vista'},
 {'headline': 'Zika Outbreak in Wichita Falls',
  'countries': None,
  'cities': 'Wichita Falls'}]
```

That looks pretty good. It won't always be perfect, but it looks to capture most of the cities and countries. We can now write this as json for loading back in.

3.3 Saving Data

We can save our list of dictionaries as json. This format can be easily read in base Python and be a number of libraries.

[23]:
```
import json

save_file = "../data/headline_cities_and_countries.json"
with open(save_file, "w") as fout:
    fout.write(json.dumps(headline_cities_and_countries))
```

Let's quickly make sure we can load this back in.

[24]:
```
with open(save_file, "r") as fin:
    check_data = json.loads(fin.read())
```

[25]:
```
check_data[-10:]
```

[25]:
```
[{'headline': 'Authorities are Worried about the Spread of Varicella in Clovis',
  'countries': None,
  'cities': 'Clovis'},
 {'headline': 'More Zika patients reported in Fort Worth',
  'countries': None,
  'cities': 'Fort Worth'},
 {'headline': 'Zika symptoms spotted in Boynton Beach',
  'countries': None,
  'cities': 'Boynton Beach'},
 {'headline': 'Outbreak of Zika in Portoviejo',
  'countries': None,
  'cities': 'Portoviejo'},
 {'headline': 'Influenza Exposure in Muscat',
  'countries': None,
  'cities': 'Muscat'},
 {'headline': 'Rumors about Rabies spreading in Jerusalem have been refuted',
  'countries': None,
  'cities': 'Jerusalem'},
 {'headline': 'More Zika patients reported in Indang',
  'countries': None,
  'cities': 'Indang'},
 {'headline': 'Suva authorities confirmed the spread of Rotavirus',
  'countries': None,
  'cities': 'Suva'},
 {'headline': 'More Zika patients reported in Bella Vista',
  'countries': None,
  'cities': 'Bella Vista'},
 {'headline': 'Zika Outbreak in Wichita Falls',
  'countries': None,
  'cities': 'Wichita Falls'}]
```

It looks to be in shape!

[26]:
```
check_data[:5]
```

[26]:
```
[{'headline': 'Zika Outbreak Hits Miami',
  'countries': None,
  'cities': 'Miami'},
 {'headline': 'Could Zika Reach New York City?',
  'countries': None,
  'cities': 'New York City'},
 {'headline': 'First Case of Zika in Miami Beach',
  'countries': None,
  'cities': 'Miami Beach'},
 {'headline': 'Mystery Virus Spreads in Recife, Brazil',
  'countries': 'Brazil',
  'cities': 'Recife'},
 {'headline': 'Dallas man comes down with case of Zika',
  'countries': None,
  'cities': 'Dallas'}]
```

Let's also save out the mappings.

[27]:
```
with open("../data/city_accent_mapping.json", "w") as fout:
    fout.write(json.dumps(city_accent_mapping))
```

[28]:
```
with open("../data/country_accent_mapping.json", "w") as fout:
    fout.write(json.dumps(country_accent_mapping))
```

4 Producing a DataFrame

We can directly convert our results into a Pandas DataFrame by reading in the json. The Pandas DataFrame is like a spreadsheet in Python and allows us to quickly analyze and manipulate our data.

[29]:
```
import pandas as pd

data = pd.read_json("../data/headline_cities_and_countries.json")
data = data.replace({None: np.nan})

data.head(10)
```

[29]:

	headline	countries	cities
0	Zika Outbreak Hits Miami	NaN	Miami
1	Could Zika Reach New York City?	NaN	New York City
2	First Case of Zika in Miami Beach	NaN	Miami Beach
3	Mystery Virus Spreads in Recife, Brazil	Brazil	Recife
4	Dallas man comes down with case of Zika	NaN	Dallas
5	Trinidad confirms first Zika case	NaN	Trinidad
6	Zika Concerns are Spreading in Houston	NaN	Houston
7	Geneve Scientists Battle to Find Cure	NaN	Geneve
8	The CDC in Atlanta is Growing Worried	NaN	Atlanta

```
9       Zika Infested Monkeys in Sao Paulo       NaN      Sao Paulo
```

This dataframe is the final output of the first section. We will use the dataframe in the next part to find geographic coordinates of the headlines.

5 Summary

In this notebook we:

- Processed the headline data
- Found the cities and/or countries in the headlines

The end deliverable from this section is a Pandas DataFrame with each headline and the city and/or country mentioned in the headline. There may be some errors in the extraction, but we'll move to the next section. If we encounter errors along the way (as is inevitable), we can always correct them as needed.

[ ]: