task03-mine-popular-dishes

September 30, 2018

loganjtravis@gmail.com (Logan Travis)

```
In [1]: %%capture --no-stdout

        # Imports; captures errors to supress warnings about changing
        # import syntax
        from lxml import html
        import matplotlib.pyplot as plot
        import nltk
        import numpy as np
        import pandas as pd
        import random
        import re
        import wikipediaapi
```

```
In [2]: # Set random seed for repeatability
        random.seed(42)
```

```
In [3]: # Set matplotlib to inline to preserve images in PDF
        %matplotlib inline
```

1 Summary

From course page Week 3 > Task 3 Information > Task 3 Overview:

> The goal of this task is to mine the data set to discover the common/popular dishes of a particular cuisine. Typically when you go to try a new cuisine, you don't know beforehand the types of dishes that are available for that cuisine. For this task, we would like to identify the dishes that are available for a cuisine by building a dish recognizer.
>
> **Instructions**
>
> Before you begin, make sure you have downloaded the data set and any additional tools you wish to use, as described on the Data Set and Toolkit Acquisition page.
>
> Some questions to consider when building the dish recognizer are the following:
>
> 1. What types of dishes are present in the reviews for a cuisine?
> 2. Are there any surprising dishes in the list you annotated?
> 3. What types of dishes were you able to find?

2 Clean List of Dishes for Mexican Cuisine

I chose to explore dishes for Mexican cuisine. I both enjoy Mexican food and thought the use of Spanish words might yield interesting results.

```
In [4]: # Set paths to data source, work in process ("WIP"), and output
        PATH_SOURCE = "source/"
        PATH_WIP = "wip/"
        PATH_OUTPUT = "output/"

        # Set file paths
        PATH_SOURCE_MEXICAN_LABELS = PATH_SOURCE + "labels/Mexican.label"
        PATH_SOURCE_MEXICAN_TO_DEL = PATH_SOURCE + "labels/Mexican_TO_DEL.label"
        PATH_SOURCE_MEXICAN_TO_FLIP = PATH_SOURCE + "labels/Mexican_TO_FLIP.label"
        PATH_SOURCE_YELP_REVIEWS = PATH_SOURCE + \
                "yelp_academic_dataset_review.pkl.gzip"
        PATH_SOURCE_YELP_REST_TO_CUISINES = PATH_SOURCE + \
                "yelp_academic_dataset_restaurant_to_cuisine.pkl.gzip"
        PATH_WIP_YELP_REVIEWS_MEXICAN = PATH_WIP + \
                "yelp_academic_dataset_review_mexican_corpus.txt"
        PATH_WIP_MEXICAN_FINAL = PATH_WIP + "labels/Mexican_FINAL.label"

        # Set paths to AutoPhrase output
        AUTOPHRASE_LOG = "AutoPhrase/models/yelp_mexican_dishes/log.txt"
        AUTOPHRASE_RESULTS = "AutoPhrase/models/yelp_mexican_dishes/AutoPhrase.txt"
        AUTOPHRASE_RESULTS_BLIND = "AutoPhrase/models/yelp_mexican_dishes_blind/AutoPhrase.txt"
```

2.1 Inspect Provided List

The assignment included an initial list a list of frequent phrases for Mexican cuisine tagged with either a 1 to indicate a *potential* dish name or 0 to indicate a non-dish phrase. I emphasize “potential” because a cursory glance reveals non-dish phrases like “in n out” and “service stars”.

```
In [5]: # Read initial dish list for Mexican cuisine
        dfMexDishes = pd.read_csv(PATH_SOURCE_MEXICAN_LABELS, sep="\t", names=["dish", "include"

In [6]: # Make `dish` column the index
        dfMexDishes.set_index("dish", inplace=True)

In [7]: # Set `include` column to boolean data type
        dfMexDishes.include = dfMexDishes.include.astype(np.bool_,)

In [8]: # Print dish list shape and head
        print("---INITIAL---")
        print("Dish list has shape {} with {:,} dishes to include and {:,} common phrases to exc
                dfMexDishes.shape, \
                sum(dfMexDishes.include), \
                sum(~dfMexDishes.include)))
        dfMexDishes.head(5)
```

```
---INITIAL---
Dish list has shape (597, 1) with 200 dishes to include and 397 common phrases to exclude as dis
```

```
Out[8]:                include
        dish
        fried egg         True
        in n out          True
        triple sec        True
        mexican food      True
        service stars     True
```

2.2 Remove False-Positives

I reviewed the list of potential dishes - those frequent phrases tagged with a one - to compile a list of fals-positive phrases to remove. That list included both non-dish phrases and dish names not relevant to Mexican cuisine. I removed nearly all of the potential dishes raising a question that I will explore in detail later, “With such poor performance from frequent phrases, how else can I use to create a list of Mexican dishes?”

```
In [9]: # Read dishes to drop for Mexican cuisine
        wip = pd.read_csv(PATH_SOURCE_MEXICAN_TO_DEL, sep="\t", \
                          names=["dish", "include"], index_col=0)
```

```
In [10]: # Removed dishes to drop from dish list
         dfMexDishes.drop(wip.index, inplace=True)
```

```
In [11]: # Print dish list shape
         print("---AFTER REMOVING FALSE POSITIVES---")
         print("Dish list has shape {} with {:,} dishes to include and {:,} common phrases to ex
                 dfMexDishes.shape, \
                 sum(dfMexDishes.include), \
                 sum(~dfMexDishes.include))
```

```
---AFTER REMOVING FALSE POSITIVES---
Dish list has shape (412, 1) with 15 dishes to include and 397 common phrases to exclude as dish
```

2.3 Flip Indicator for False-Negatives

I identified less than ten false-negatives, dish names tagged with a zero, that I correct below.

```
In [12]: # Read dishes flip drop for Mexican cuisine
         wip = pd.read_csv(PATH_SOURCE_MEXICAN_TO_FLIP, sep="\t", \
                           names=["dish", "include"], index_col=0)
```

```
In [13]: # Removed dishes to drop from dish list
         dfMexDishes.loc[wip.index, "include"] = ~dfMexDishes.loc[wip.index, "include"]
```

```
In [14]: # Print dish list shape
         print("---AFTER CORRECTIN FALSE NEGATIVES---")
         print("Dish list has shape {} with {:,} dishes to include and {:,} common phrases to ex
                 dfMexDishes.shape, \
                 sum(dfMexDishes.include), \
                 sum(~dfMexDishes.include))

---AFTER CORRECTIN FALSE NEGATIVES---
Dish list has shape (412, 1) with 21 dishes to include and 391 common phrases to exclude as dish
```

2.4 Add Dishes from Wikipedia

Very few of the frequent phrases qualify as Mexican dishes. I therefore sought other sources finding Wikipedia page for "List of Mexican dishes". The AutoPhrase package - an improved version of SegPhrase - I use later *should* benefit from an expert list of labels. The cleaned list of frequent phrases includes fewer than 25 dish names. I therefore decided to add the list of dishes on Wikipedia.

```
In [15]: # Get Wikipedia page "List of Mexican dishes" and parse as HTML
         wp = wikipediaapi.Wikipedia('en', extract_format=wikipediaapi.ExtractFormat.HTML)
         wpMexDishesPage = wp.page("List_of_Mexican_dishes")

In [16]: # Define helper function to pretty-print secitons
         def printSections(sections, level=0):
             """Pretty-print sections from `wikipediaapi` page."""
             for i, s in enumerate(sections):
                 print("{}{:d}. {}".format(" " * 4 * level, i, s.title))
                 printSections(s.sections, level + 1)

In [17]: # Examine sections
         printSections(wpMexDishesPage.sections)

0. Antojitos
1. Cheese  dishes
2. Egg dishes
3. Meat dishes
    0. Beef dishes
    1. Goat dishes
    2. Pork dishes
    3. Poultry dishes
    4. Other meat and protein dishes
4. Moles, sauces, dips and spreads
5. Rice dishes
6. Seafood dishes
7. Soups and stews
8. Vegetable dishes
9. Desserts and sweets
10. Beverages
```

```
    0. Non-alcoholic
    1. Alcoholic
11. See also
12. References
13. External links
```

```
In [18]: # Get text from an example section
         wpMexDishesPage.sections[0].text
```

```
Out[18]: '<p>Street food in Mexico, called <i>antojitos</i> is prepared by street vendors and at
```

Each section includes one or more unorder list of dish names. Those dish names includes some unwanted text, usually explanatory, that I remove with the helper function below.

```
In [19]: # Define helper function to get list of dishes from section text
         def getDishesFromText(sectionText, removeTextAfter="[,-]", wordLimit=3):
             """Return a list of dish names from section text."""
             tree = None

             # Create an `lxml` element tree from HTML.
             tree = html.fromstring(sectionText)

             # Get dishes from <li> element text
             dishes = tree.xpath("//li/text()")

             # Remove parentheticals
             dishes = [re.sub(r"\(.*?\)", "", t) for t in dishes]

             # Remove text after passed characters
             dishes = [re.sub("(?<={}).*$".format(removeTextAfter), "", t) for t in dishes]

             # Trim to word limit
             dishes = [" ".join(re.split(r"\W+", t)[:wordLimit]).strip() for t in dishes]

             # Return list of dishes
             return set(dishes)
```

```
In [20]: # Define helper function to recursively get dishes from all
         # sections
         def getDishesFromSection(section):
             """Recusively print list of sections from `wikipediaapi` page."""
         #     print(section.title)
             if(len(section.sections) == 0):
                 return getDishesFromText(section.text)
             else:
                 dishes = set()
                 for s in section.sections:
                     dishes.update(getDishesFromSection(s))
                 return dishes
```

```
In [21]: # Get Mexican dishes from Wikipedia page "List of Mexican dishes"
         wpMexDishes = set()
         for section in wpMexDishesPage.sections[:11]:
             wpMexDishes.update(getDishesFromSection(section))
```

```
In [22]: # Remove empty items and known bad elements then format for
         # inclusion in common phrases
         wpMexDishes = [(d.lower(), 1) for d in wpMexDishes \
                        if d not in ["is of", "or", "", "as a", "where these"]]
```

```
In [23]: # Convert to dataframe
         dfMexDishesFromWP = pd.DataFrame(wpMexDishes, columns=["dish", "include"])
```

```
In [24]: # Make `dish` column the index
         dfMexDishesFromWP.set_index("dish", inplace=True)
```

```
In [25]: # Set `include` column to boolean data type
         dfMexDishesFromWP.include = dfMexDishesFromWP.include.astype(np.bool_,)
```

```
In [26]: # Merge original dish list and Wikipedia dish list then fill
         # missing values as False
         dfMexDishes = dfMexDishes.merge(dfMexDishesFromWP, how="outer", \
                                         left_index=True, right_index=True, \
                                         suffixes=["_initial", "_from_wp"])
         dfMexDishes.fillna(False, inplace=True)
```

```
In [27]: # Determine final inclusion from initial list or from Wikipedia
         dfMexDishes["include_combined"] = dfMexDishes.include_initial | dfMexDishes.include_fro
```

```
In [28]: # List dishes to include
         print("---FINAL---")
         print("Found {:,} dishes to incldue and {:,} common phrases to exclude as dishes.".form
                 sum(dfMexDishes.include_combined), \
                 sum(~dfMexDishes.include_combined)))
```

```
---FINAL---
Found 242 dishes to incldue and 391 common phrases to exclude as dishes.
```

```
In [29]: # Save dish list to working file for read into AutoPhrase
         includeDishes = dfMexDishes[dfMexDishes.include_combined].index.to_series()
         includeDishes.to_csv(PATH_WIP_MEXICAN_FINAL, header=False, index=False)
```

3 Evaluate Dish List Against Frequent Phrases

The "distant expert" - a term used by AutoPhrase's authors [1][2] - list of Mexican dishes clearly out performed the initial frequent phrase list. However, AutoPhrase like SegPhrase before it can improve its frequent phrase mining by introducing a list of known good labels. I do not expect it to beat the distant experts but want to evaluate how much it improves.