ITESO

Ingredientes Cocina

PAP Modelación Matemática para el desarrollo de planes y proyectos de negocio

Sara Eugenia Rodríguez Reyes

15/11/2015

Exploración y predicción de datos utilizando el lenguaje de programación Python

Introducción

Este trabajo tiene el objetivo de usar los ingredientes de recetas para categorizar el tipo de cocina; ya sea mexicana, hindú, italiana, etc.

Las zonas geográficas y culturales más fuertes se encuentran vinculadas por ingredientes culinarios en común. Lo que se busca es predecir la categoría de cocina de un platillo dado una lista de sus ingredientes.

Yummly es la página de donde se consiguió la base de datos; esta es una aplicación utilizada para la búsqueda de recetas por ingredientes, dietas, alergias, nutrición, sabor, calorías, grasas, precios, cocina, tiempo, etc.

En la base de datos, se incluye el id de la receta, el tipo de cocina y la lista de ingredientes de cada receta.

Código

A continuación se presenta el código con el que se hizo la exploración de datos; este mismo es comentado.

```
#Cargar librerías
import pandas as pd
import numpy as np  # procesa números, strings, objetos
from time import time
import matplotlib.pyplot as plt    #graficas
from nltk.stem import WordNetLemmatizer #para palabras
from collections import Counter  #contar objetos

# Cargar datos
train =
pd.read_json('C:\Users\Saruki\Documents\ITESO\PAP2\cooking\entr.json')
test =
pd.read_json('C:\Users\Saruki\Documents\ITESO\PAP2\cooking\prueba.json')
```

En el siguiente extracto de código se van a quitar los duplicados de tipos de cocina y a cada tipo se le asignarán todos los ingredientes que se usan.

```
train.head()
train['solo_ingredientes'] = train.ingredients.apply(lambda x: ", ".join(x))
train.solo_ingredientes.head()
cuisine_ingredient = train.groupby('cuisine')['ingredients'].sum() #se suman
los tipos de cocina e ingredientes
cuisine_ingredient
```

```
cuisine
brazilian       [ice cubes, club soda, white rum, lime, turbin...
british         [greek yogurt, lemon curd, confectioners sugar...
cajun_creole    [herbs, lemon juice, fresh tomatoes, paprika, ...
chinese         [low sodium soy sauce, fresh ginger, dry musta...
filipino        [eggs, pepper, salt, mayonaise, cooking oil, g...
french          [sugar, salt, fennel bulb, water, lemon olive ...
greek           [romaine lettuce, black olives, grape tomatoes...
indian          [water, vegetable oil, wheat, salt, black pepp...
irish           [cooking spray, salt, black pepper, yukon gold...
italian         [sugar, pistachio nuts, white almond bark, flo...
jamaican        [plain flour, sugar, butter, eggs, fresh ginge...
japanese        [sirloin, mirin, yellow onion, low sodium soy ...
korean          [jasmine rice, garlic, scallions, sugar, shiit...
mexican         [olive oil, purple onion, fresh pineapple, por...
moroccan        [ground cloves, whole nutmegs, ground ginger, ...
russian         [water, grits, mozzarella cheese, salt, water,...
southern_us     [plain flour, ground pepper, salt, tomatoes, g...
spanish         [olive oil, salt, medium shrimp, pepper, garli...
thai            [sugar, hot chili, asian fish sauce, lime juic...
vietnamese      [soy sauce, vegetable oil, red bell pepper, ch...
```

De la cocina Brasileña, se va a separar por ingredientes utilizados, cuántas veces son usados y cuánto representa este ingrediente del total.

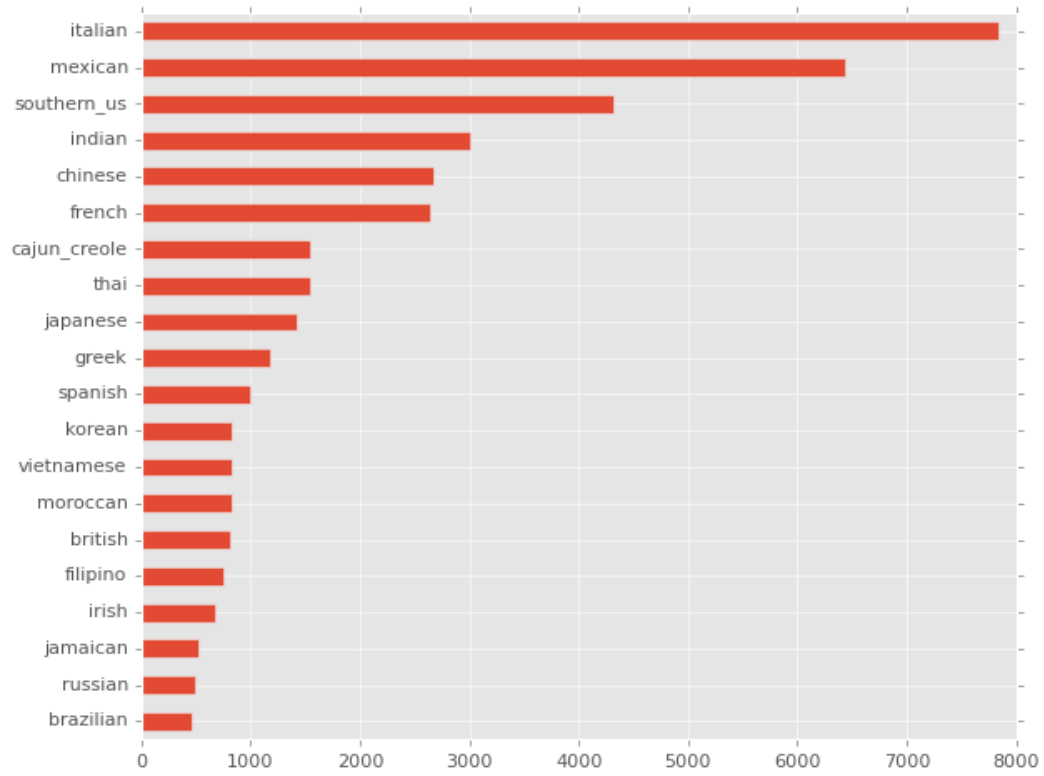
```
full_corpus = train['ingredients'].sum()
from collections import Counter
brazil_corpus = cuisine_ingredient[0]
brazil_c = Counter(brazil_corpus)
brazil_tf = [(x, y, float(y) / np.sum(brazil_c.values())) for x, y in
brazil_c.most_common()]
brazil_tf[0:10] #que muestre solo los 10 ingredientes más utilizados
```

Este tipo de cocina usa mucha sal para preparar sus platillos; de igual forma se utiliza mucho la cebolla y el aceite de oliva. Es un tipo de cocina muy condimentada; ya que entre los principales 10 ingredientes utilizados se encuentran condimentos, más que frutas, verduras y carnes. La primer columna muestra el ingrediente, la segunda muestra cuántas veces se utiliza y la tercer columna muestra el porcentaje.

```
[(u'salt', 194, 0.04363472784525416),
 (u'onions', 133, 0.029914529914529916),
 (u'olive oil', 118, 0.026540710751237068),
 (u'lime', 89, 0.020017993702204 23),
 (u'water', 87, 0.019568151147098516),
 (u'garlic cloves', 83, 0.0186684660 3688709),
 (u'garlic', 82, 0.018443544759334234),
 (u'cachaca', 70, 0.015744489428699954),
 (u'sugar', 69, 0.0155195681511471),
 (u'tomatoes', 63, 0.01417004048582996)]
```

Se hará un análisis de la distribución de los tipos de cocina, así como de los ingredientes que son utilizados.

```
# Estilo del gráfico (como usar ggplot en R)
plt.style.use(u'ggplot')
# Encontrar la distribución de los tipos de cocina
cuisine_distribution = Counter(train.cuisine)#cuenta cuantas veces es usada
esa cocina
#Gráfico: ordena de mayor a menor las frecuencias de las cocinas y que se
grafiquen en un gráfico de barras
cuisine_fig = pd.DataFrame(cuisine_distribution,
index=[0]).transpose()[0].sort(ascending=False,
inplace=False).plot(kind='barh')
#invierte los ejes para que las frecuencias estén en el eje de las X
cuisine_fig.invert_yaxis()
cuisine_fig = cuisine_fig.get_figure()
cuisine_fig.tight_layout()
plt.show()
```

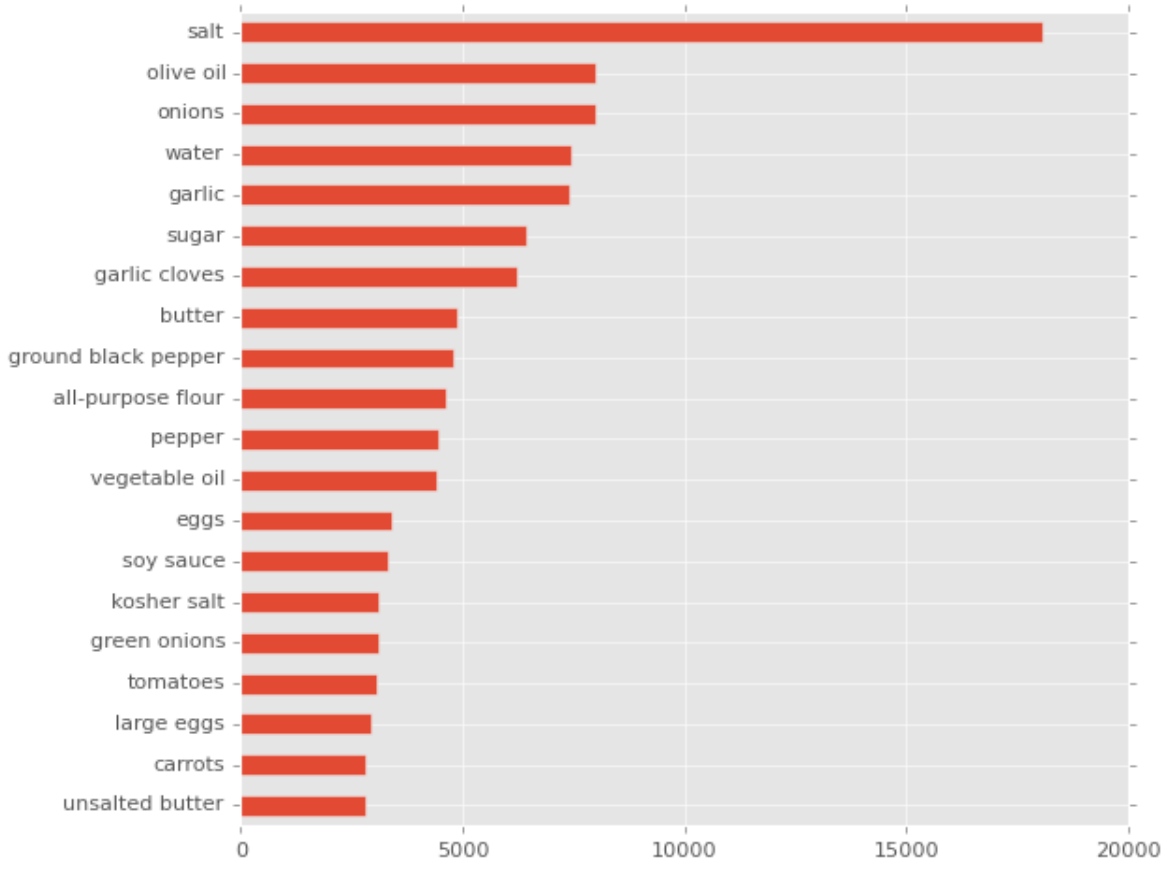


La cocina mexicana se encuentra en segundo lugar; después de la italiana. Las recetas menos utilizadas son las brasileñas.

Cocina	Cantidad
Italian	7838
Mexican	6438
Southern_us	4320
Indian	3003
Chinese	2673
French	2646
Cajun_creole	1546
Thai	1539
Japanese	1423
Greek	1175
Spanish	989
Korean	830
Vietnamese	825
Moroccan	821
British	804
Filipino	755
Irish	667
Jamaica	526
Russian	489
Brazilian	467

```
# Encontrar la distribución de los ingredientes
recipe_ingredient = [Counter(recipe) for recipe in train.ingredients]
ingredient_distribution = sum(recipe_ingredient, Counter())
ingredient_fig = pd.DataFrame(ingredient_distribution,
index=[0]).transpose()[0].sort(ascending=False,
inplace=False)[:20].plot(kind='barh')
ingredient_fig.invert_yaxis()
ingredient_fig = ingredient_fig.get_figure()
ingredient_fig.tight_layout()
plt.show()
```

La gráfica muestra los 20 ingredientes más utilizados en la preparación de alimentos.



A continuación se presenta cuáles cocinas son las que utilizan más sal en lugar de salsa de soya.

```
for cuisine in np.unique(train.cuisine):
    total = 0
    with_salt = 0
    with_soy = 0
    for i in range(len(train.cuisine)):
        if train.cuisine[i]==cuisine:
            total += 1
            if train.ingredients[i].count('salt')>0:
                with_salt += 1
            if train.ingredients[i].count('sauce')>0:
                with_soy += 1
    print(cuisine, with salt/total)
```

Las cocinas que utilizan más sal son la hindú y la jamaiquina; mientras que las cocinas que utilizan más salsa de soya son la tailandesa y la japonesa.

Cocina	Cantidad Sal/total
brazilian	0.4154
British	0.5062
Cajun_creole	0.4805
Chinese	0.3393
Filipino	0.5576
French	0.4546
Greek	0.4859
Indian	0.6440
Irish	0.5637
Italian	0.4406
Jamaican	0.6311
Japanese	0.2965
Korean	0.3048
Mexican	0.4224
Moroccan	0.5030
Russian	0.5889
Southern_us	0.5300
Spanish	0.4651
Thai	0.2573
Vietnamese	0.32

Predicción

Para la predicción del tipo de cocina, se utilizará el modelo de Random Forest utilizando como variables los ingredientes.

```
# Machine Learning con Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import cross_val_predict
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer

# Ingredientes únicos
palabras = [' '.join(item) for item in train.ingredients]
# Construir Word2Vec
#La siguiente línea convierte el texto a una matriz de vectores
vec = CountVectorizer(max_features=2000)
Word2Vec = vec.fit(palabras).transform(palabras).toarray()

## Usar Random Forest con Clasificación
random_forest = RandomForestClassifier(n_estimators=200)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
# Cuanto tiempo tarda en entrenar el modelo
start = time()
random_forest.fit(Word2Vec, train.cuisine)
print("RandomForest Training finished in %.2f" % (time() - start))
```

El modelo tarda en entrenar 353.92 segundos.

```
# Evaluar con el set de entrenamiento
start = time()
train_pred = cross_val_predict(random_forest, Word2Vec, train.cuisine, cv=2)
print("RandomForest Evaluation finished in %.2f" % (time() - start))
```

El modelo tarda 693.13 segundos en hacer la predicción.

```
# Exactitud
print("Accuracy: ", accuracy_score(train.cuisine, train_pred))
```

```
('Accuracy: ', 0.74133856288027355)
```

El modelo tiene una exactitud del 74.05%

```
# Prueba
test_words = [' '.join(item) for item in test.ingredients]
test_bag = vec.transform(test_words).toarray()

# Predicción
result = random_forest.predict(test_bag)
output = pd.DataFrame(data={"id":test.id, "cuisine":result})
```

Aquí un pequeño vistazo a las predicciones realizadas con el modelo.

	cuisine	id
0	italian	18009
1	southern_us	28583
2	italian	41580
3	cajun_creole	29752
4	italian	35687
5	southern_us	38527
6	southern_us	19666
7	chinese	41217
8	mexican	28753
9	southern_us	22659
10	italian	21749
11	greek	44967
12	indian	42969
13	italian	44883
14	british	20827
15	italian	23196
16	southern_us	35387
17	southern_us	33780
18	mexican	19001
19	southern_us	16526
20	japanese	42455
21	indian	47453
22	french	42478
23	indian	11885

Conclusión

La característica más llamativa de Python es que la forma de delimitar los bloques es con el sangrado. Ya no tienes que usar los {}. Otra ventaja es que puedes programar funciones en una sola línea; lo cual ahorra líneas de código.

Es gratis; pero el inconveniente es que es un poco complicado empezarlo a usar, ya que se tiene que descargar Anaconda y Ipython.

Es un lenguaje de programación del mundo real que es usado tanto con fines académicos como comerciales.