1. 概要

この課題では、自分のいる場所の近くの駅や食事場所の検索を実現するために、施設名を入力として、最寄りの駅やレストランを出力とするプログラムを作成し、それをWebブラウザ上で動作するようにした、

なお、サービスのURLはhttps://desolate-crag-93252.herokuapp.com/ である。

1. 利用したWebサービスの情報

この課題で使用したWebサービスは、HeartRails Express と、Google Maps PlatformのGeocoding API、Places APIである。当初はリクルートWebサービスのグルメサーチをレストラン検索に用いることを検討していたが、ホットペッパーグルメに登録されている店舗が少ない地域では正確な検索結果が得られないことを考慮してPlaces APIによる検索に変更した。

1. ソースコード

シェル上の処理とWebサービスの両方に用いたgooglemap\_test.pyというプログラムのソースコードを以下に示す。

import googlemaps

from dotenv import load\_dotenv

import sys

import os

import pprint

import heartrails\_json

load\_dotenv()

def create\_shopinfo(index, result):

print("""------ No. {} ------

【店名】 {}

【営業情報】 {}

【住所】 {}""".format(index+1, # No.

result["results"][index]["name"], # 店名

isopen(result["results"][index]),

result["results"][index]["vicinity"]

))

def info\_output(index, result):

return """------ No. {} ------

【店名】 {}

【営業情報】 {}

【住所】 {}""".format(index+1, # No.

result["results"][index]["name"], # 店名

isopen(result["results"][index]),

result["results"][index]["vicinity"]

)

def isopen(search\_result):

if (search\_result.get('opening\_hours', None) == None):

return "営業情報が存在しません"

elif(search\_result["opening\_hours"]["open\_now"] == True):

return "営業中"

else:

return "営業時間外"

def search(place, type, distance=200):

key = os.environ.get('GOOGLE\_API\_KEY') # 上記で作成したAPIキーを入れる

client = googlemaps.Client(key) #インスタンス生成

geocode\_result = client.geocode(place)

loc = geocode\_result[0]['geometry']['location'] # 軽度・緯度の情報のみ取り出す

s = "" #string

if type == "food":

place\_result = client.places\_nearby(location=loc, radius=distance, type='restaurant') #半径200m以内のレストランの情報を取得

search\_num = len(place\_result["results"])

for index in range(search\_num):

s += info\_output(index, place\_result)

s += "\n"

return s

else:

return heartrails\_json.info\_search(loc, s)

def main():

key = os.environ.get("GOOGLE\_API\_KEY") #上記で作成したAPIキーを入れる

client = googlemaps.Client(key) #インスタンス生成

num\_of\_val = len(sys.argv)

geocode\_result = client.geocode(sys.argv[1])

loc = geocode\_result[0]['geometry']['location'] # 軽度・緯度の情報のみ取り出す

if (num\_of\_val >= 3 and sys.argv[2] == "food"):

place\_result = client.places\_nearby(location=loc, radius=200, type='restaurant') #半径200m以内のレストランの情報を取得

search\_num = len(place\_result["results"])

for index in range(search\_num):

create\_shopinfo(index, place\_result)

else:

heartrails\_json.heartrailsAPI(loc)

if \_\_name\_\_ == "\_\_main\_\_":

main()

上記のソースコードでimportしたheartrails\_json.pyのソースコードを以下に示す。

import sys

import urllib.request

import urllib.parse

import json

def create\_station\_info(index, station):

return """------ No, {} ------

【駅名】 {}

【路線名】 {}

【現在地からの距離】 {}""".format(index,

station["name"], #駅名

station["line"], #路線名

station["distance"] #現在地からの距離

)

def main():

url = "http://express.heartrails.com/api/json?{}".format(

urllib.parse.urlencode(

{"method" : "getStations",

"y" : "".join(sys.argv[1]), # 入力からクエリを生成

"x" : "".join(sys.argv[2]), # 入力からクエリを生成

}))

#print("URL:",url)

f\_url = urllib.request.urlopen(url).read()

json\_result = json.loads(f\_url.decode("utf-8"))

#output

print("最寄り駅の一覧を表示します。")

for index, station in enumerate(json\_result["response"]["station"]):

print(create\_station\_info(index+1, station))

def heartrailsAPI(loc):

url = "http://express.heartrails.com/api/json?{}".format(

urllib.parse.urlencode(

{"method" : "getStations",

"x" : loc['lng'], # 入力からクエリを生成(経度)

"y" : loc['lat'], # 入力からクエリを生成(緯度)

}))

#print("URL:",url)

f\_url = urllib.request.urlopen(url).read()

json\_result = json.loads(f\_url.decode("utf-8"))

#output

print("最寄り駅の一覧を表示します。")

for index, station in enumerate(json\_result["response"]["station"]):

print(create\_station\_info(index+1, station))

def info\_search(loc, s):

url = "http://express.heartrails.com/api/json?{}".format(

urllib.parse.urlencode(

{"method" : "getStations",

"x" : loc['lng'], # 入力からクエリを生成

"y" : loc['lat'], # 入力からクエリを生成

}))

#print("URL:",url)

f\_url = urllib.request.urlopen(url).read()

json\_result = json.loads(f\_url.decode("utf-8"))

#output

for index, station in enumerate(json\_result["response"]["station"]):

s += create\_station\_info(index+1, station)

s += "\n"

return s

if \_\_name\_\_ == "\_\_main\_\_":

main()

また、シェル上で実行した際の例を3つ掲載する。

% python googlemap\_test.py 東北大学青葉山キャンパス food

------ No. 1 ------

【店名】 Kotaro

【営業情報】 営業時間外

【住所】 青葉山みどり厚生会館2F, 荒巻, 青葉区 仙台市

------ No. 2 ------

【店名】 TOHOKU UNIV.COOP LUNCH BOX & HALAL

【営業情報】 営業情報が存在しません

【住所】 青葉山みどり厚生会館2F, 荒巻, 青葉区 仙台市

------ No. 3 ------

【店名】 Midori Shokudo

【営業情報】 営業時間外

【住所】 Aoba Aramaki, Aoba Ward, Sendai

% python googlemap\_test.py 国会議事堂 food

------ No. 1 ------

【店名】 国会中央食堂

【営業情報】 営業時間外

【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

------ No. 2 ------

【店名】 The House of Representatives 2nd building Cafeteria

【営業情報】 営業時間外

【住所】 2-chōme-1-2 Nagatachō, Chiyoda City

------ No. 3 ------

【店名】 Issa Soba

【営業情報】 営業時間外

【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

------ No. 4 ------

【店名】 参議院 議員食堂

【営業情報】 営業時間外

【住所】 参議院本館 2階, 1-chōme-7−１ Nagatachō, Chiyoda City

------ No. 5 ------

【店名】 Mitou-an

【営業情報】 営業時間外

【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

------ No. 6 ------

【店名】 Yoshinoya

【営業情報】 営業時間外

【住所】 1-chōme-7-1 Nagatachō, Chiyoda City

% python googlemap\_test.py 東北大学病院

最寄り駅の一覧を表示します。

------ No, 1 ------

【駅名】 北四番丁

【路線名】 仙台市南北線

【現在地からの距離】 680m

------ No, 2 ------

【駅名】 勾当台公園

【路線名】 仙台市南北線

【現在地からの距離】 1090m

------ No, 3 ------

【駅名】 北仙台

【路線名】 JR仙山線

【現在地からの距離】 1300m

------ No, 4 ------

【駅名】 北仙台

【路線名】 仙台市南北線

【現在地からの距離】 1330m

次に、Webサービスを作るために用いたmoyori\_search.pyのコードを掲載する。

from flask import Flask,request, render\_template, Markup

import os

import googlemap\_test

app = Flask(\_\_name\_\_)

@app.template\_filter('cr')

def cr(arg):

return Markup(arg.replace('\n', '<br>'))

@app.route('/', methods=['GET'])

def get():

return render\_template('map\_search.html', \

title = 'Web Computing Assignment', \

subtitle = '最寄りの施設・駅の検索', \

message = "現在地から近い施設名を入力してください。\n下のラジオボタンで検索したい店を選ぶことができます。", \

results = "ここに結果が表示されます")

@app.route('/', methods=['POST'])

def post():

name = request.form['choice']

if name == "駅":

search\_type = "station"

else:

search\_type = "food"

return render\_template('map\_search.html', \

title = 'Web Computing Assignment', \

subtitle = '最寄りの施設・駅の検索', \

message = '現在地から近い施設名を入力してください。\n下のラジオボタンで検索したい店を選ぶことができます。', \

message2 = request.form["place"] + 'の近くの' + request.form["choice"] + 'の検索結果を表示します', \

results = googlemap\_test.search(request.form["place"], search\_type, distance=request.form["sel"]))

if \_\_name\_\_ == '\_\_main\_\_':

# Bind to PORT if defined, otherwise default to 5000.

port = int(os.environ.get('PORT', 5000))

app.run(host='0.0.0.0', port=port)

1. 考察など

このWebサービスを作る上で工夫した点は、APIキーを環境変数として定義することで、APIキーの悪用を防いだことである。最初にプログラムを作る際に直接APIキーを記載していたが、Google Maps PlatformのAPIはリクエスト数が増えると有料になってしまうことを考えて環境変数に定義することにした。

苦労した点は、初めてFlaskに触れたため、Heroku上で正しく動作するようにportを指定しなければいけないことに気がつかなかったことや、Places APIの検索結果のどこに必要な情報が入っているのかを判断することである。この点については、試行錯誤を重ねて把握することができた。

問題点としては、最初に実装した際に距離の選択肢で「選択してください」のまま最寄り駅を検索するとBad Requestになってしまうことがあった。これについては、選択肢の属性がdisabledになっていたことが原因で発生したため、距離の初期値である200をvalueに設定することで解決できた。

最後に、感想として、APIを使って何らかの便利なプログラムを作ってみることは初めてだったが、さまざまな資料を参考にすることでWeb上で動的にサイトを生成するところまで実現できたので、かなり達成感を感じた。