AN INTELLIGENT RECOMMENDATION PLATFORM THAT UTILIZES ARTIFICIAL INTELLIGENCE TO DRIVE PEOPLE TO MAKE BETTER FOOD DECISIONS

Julian Sun[1], Ang Li[2]

[1]Dos Pueblos Senior High school, 7266 Alameda Ave Goleta, CA 93117
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

People are often given options on restaurants to eat at and are also given the ratings of those restaurants. However, the ratings can sometimes be rather similar and hard to choose from, and it can also be hard to find a restaurant that suits a person's special needs, and people often eat at a singular place once they find a good restaurant; we want to change that by trying to encourage people to try new restaurants. While our idea isn't original, we still decided to add it to the list of probably hundreds of sites out there that do the same thing. We made a restaurant recommendation with one purpose in mind, to gather data from restaurants and share that data with everyone. We used many methods to get that data, create a user interface, and add that data to the site so everyone can use it. This project had originated from another idea for a Roblox sniping site which I was told was a bit too advanced and was suggested something similar in design. A restaurant recommender and a Roblox sniping site are similar in the way they both use web scraping. Web scraping is the ability of a website to get the code from other sites. Our restaurant recommender gets data from Yelp to add to our database in the way that a Roblox sniping site can get information from the Roblox catalog to display to people to see when there's a good bargain. The restaurant recommender uses the data it gets to give recommendations for a better restaurant and it gives the 3 worst reviews on the restaurant, which are to highlight some of the potential flaws of the input restaurant. The main pieces of data the recommender gets are the restaurant genre for people to see what kind of restaurant it is, the restaurant region to see what type of food the restaurant serves, the restaurant type to see if its a bar or restaurant or what type it is, the restaurant's overall rating to see how good the restaurant is, and the Yelp page of the restaurant, for a deeper look into the restaurant itself. We then use the data to get a better restaurant. We use the restaurant type, region, and genre to find a similar restaurant, and we use the rating to find a restaurant with a better rating. We used many libraries and coding languages to build our site. We used HTML and CSS to build the user interface, we used Python to run the server we were using and build the web scraper, and then we used csv for the database containing all the data. We used beautiful soup to organize the data, and we used requests to get user input. We used pandas for the data analysis and we used sklearn to build the predictor for a better restaurant.

KEYWORDS

Web scraping, Interface, Input, Python

David C. Wyld et al. (Eds): SIPM, ITCA, CMIT, FCST, CoNeCo, SAIM, ICITE, ACSIT, SNLP -2023
pp. 103-110, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130911

1. INTRODUCTION

Most people come to enjoy the great experience of eating outside; however, many people don't know how to choose where to eat at. This situation is reminiscent of how many people can't find an extremely rare limited item at one of the lowest prices possible before it's gone, which pushed the development of RblxTrade, which is a website whose catalog constantly refreshes in which you could track the prices of limited items. However, this is not a very effective website that would rarely fulfill its purpose, so we decided to build something that would both be functional and something everyone could use. So we created a Restaurant Recommender, a site that can be used to gather data on restaurants for companies to use, or for people to use to look for better restaurants to eat at but maintain the same style of food they tend to enjoy. The RblxTrade item catalog is similar to our restaurant recommender in the way that it uses web scraping, a way for code to access and use source code from other websites. And like RblxTrade, our restaurant recommender can take user input to add more to our database in the way RblxTrade takes in inputs to add more items to the catalog [4]. Our restaurant recommender takes in user input of a restaurant name and a location and will then find a restaurant by that name around that location. It will then output the 3 worst reviews on that restaurant to present some potential flaws of the input restaurant as well as a link to the Yelp page of another restaurant of similar types and genres but of a higher rating. While it currently takes the predicted restaurant from our database of random restaurants, a prediction based on location will soon be implemented. We used many libraries to build this program, but the most central part of our program is machine learning. Machine learning is the ability of a program to take in data, which can range from an object and its attributes to an entire database, and analyze that data to predict a specific value using it, which is what we used to design the predictor for finding a better restaurant. In this case, the data that we are collecting is the attributes of the input restaurant and the prediction that is being made is the better restaurant, and the way we predict the restaurant is by using linearization, where we graph points. The restaurants are stored in our database on a graph, then we plot another point that represents our input restaurant and find the point closest to it, which represents the restaurant that's the closest in comparison.

Yelp categorizes restaurants based on the type of food they serve, the regional origin of the food, and the type of restaurant. The type of food is what category of food they serve at the restaurant. The regional origin is where the food was from. And the type of restaurant is what kind of establishment the restaurant is. Yelp uses these categories to sort restaurants and we use these categories to find a recommendation for a better restaurant with similar categories. Our recommender also uses these categories to classify our restaurants in the database.

2. CHALLENGES

To build this data-driven model/application, we had a couple of data-related obstacles that we had to overcome throughout the process.

2.1. Selecting the Programming Languages to use for the Application

For the application, both a front end and a back end would be required. To make the front end, we considered that the programming language would need to be both relatively easy to start with and highly customizable, hicwh would greatly improve the development process in the early stages as well as the late stages of the application. HTML and CSS seemed to be suitable choices for this [14]. The same philosophy was used to select Python as the designated language for the backend; while Python has a very simple syntax, it also features countless packages that can be used for almost any purpose or application [13].

2.2. Identifying the Homogeneity of Certain Restaurants and using that Data to Build a Model

To classify the restaurants our recommender got, we needed to find a way to organize the categories. So we classified them by numbers representing different categories of that group. However, classifying by numbers alone would not work, since we also had to get the names of the restaurants. So we used web scraping to get the names of the restaurants. We then built a database for storing the names of the restaurants, what categories they fall into, and a link to their Yelp page. And then we made a data scraper, where we could enter a Yelp search link and we could add all the restaurants into the database.

2.3. Understanding the Complexity of how Restaurants are Grouped and how that Plays a Factor in How People Make Decisions

While many people can assume that restaurants are grouped by just fast food places or ordinary restaurants, they would be wrong. Yelp categorizes restaurants based on a huge variety of categories all falling into different groups of categories. Food categories can include fast food, seafood, Mexican, Italian, and American. There are bars, restaurants, delis, and even food trucks for types of restaurants. People may find choosing restaurants complicated already, but those who know how many categories there really are would know how complicated it really is [3].

3. SOLUTION

While Yelp restaurant categories don't narrow down people's choices of where to eat, sites like our restaurant recommender can collect data from these categories for analysis and, in some cases, provide this data to people to help them decide where to eat at.

Figure 1. Screenshot of restaurant recommender 1

Our restaurant recommender is to ensure that people can check if their restaurant is a good choice or not. Our application runs on 4 different parts: the data scraper and database, the HTML and CSS for User Interface, the server, and the predictor. The data scraper and database is a manual way of collecting and storing data on restaurants. We use HTML and CSS to create the User Interface. We use the server to be able to run both the predictor and the User Interface. The predictor is for predicting a better restaurant than the one input. Unfortunately, this model is unavailable on mobile and is only available on PCs at this time.

This application was built with mostly Python with a bit of HTML, CSS, and csv to build a database. Many Python libraries were used including BeautifulSoup, sklearn, requests, csv, pandas, NumPy, and matplotlib. This application was fully designed in Visual Studio Code. BeautifulSoup is a library that specializes in extracting data from HTML files. For instance,

BeautifulSoup is capable of retrieving all text from a page and retrieving URLs within specified tags [8]. The requests library from Python allows the code to make requests to specific URLs to retrieve data from websites [12]. The sklearn library, which is short for scikit-learn, provided the machine learning capabilities necessary to build the restaurant recommendation predictor model for the application [5]. Pandas helped with the data analysis section of the application [10]. Lastly, matplotlib creates visualizations alongside NumPy to create plots for the application [6] [7].

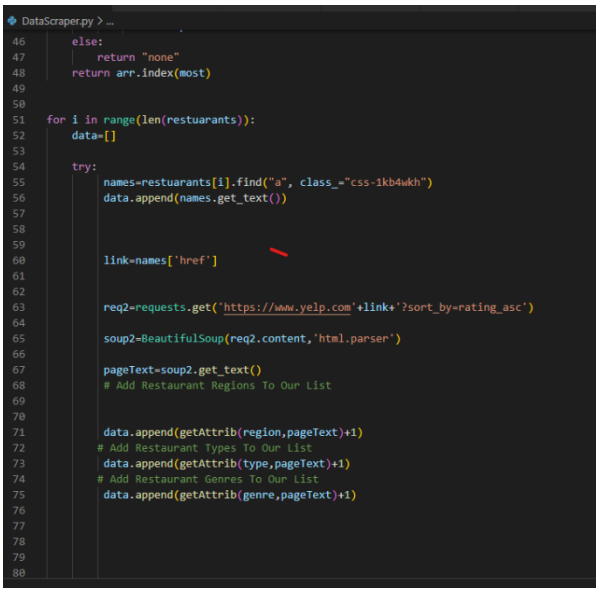


Figure 2. Screenshot of code 1

We used a lot of files to build our site. Our data scraper file is a supervisor access-only file that can collect data on a restaurant's name, categories, and Yelp page to add to our database. The picture above is a snippet from the data scraper file.

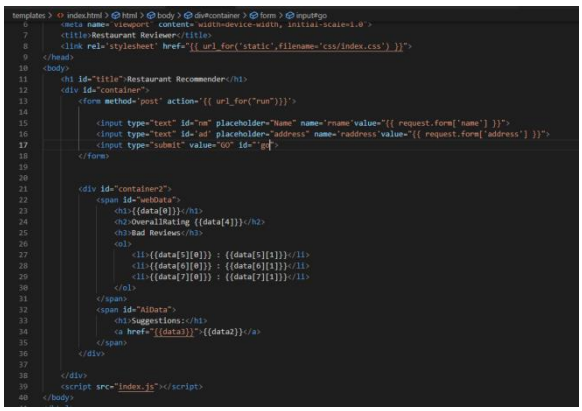


Figure 3. Screenshot of code 2

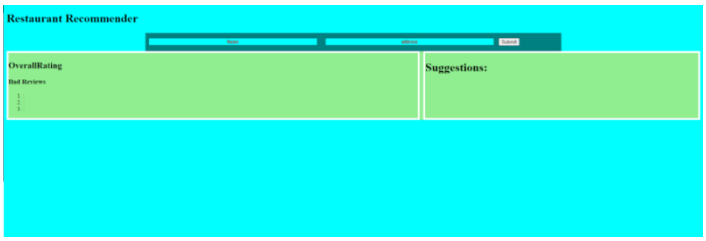
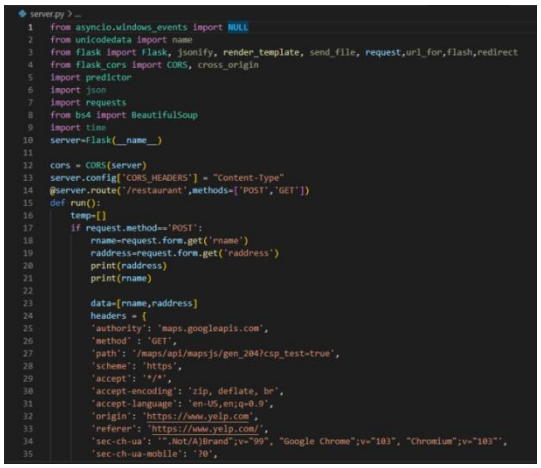


Figure 4. Screenshot of restaurant recommender 2

The HTML and CSS file were to create the user interface. The HTML was to shape the interface and put everything where it's supposed to be and the CSS was to make the interface more colorful. The HTML file is then connected to the predictor by another file.



```python
from asyncio.windows_events import NULL
from unicodedata import name
from flask import Flask, jsonify, render_template, send_file, request,url_for,flash,redirect
from flask_cors import CORS, cross_origin
import predictor
import json
import requests
from bs4 import BeautifulSoup
import time
server=Flask(__name__)

cors = CORS(server)
server.config['CORS_HEADERS'] = "Content-Type"
@server.route('/restaurant',methods=['POST','GET'])
def run():
    temp=[]
    if request.method=='POST':
        rname=request.form.get('rname')
        raddress=request.form.get('raddress')
        print(raddress)
        print(rname)

        data=[rname,raddress]
        headers = {
        'authority': 'maps.googleapis.com',
        'method' : 'GET',
        'path': '/maps/api/mapsjs/gen_204?csp_test=true',
        'scheme': 'https',
        'accept': '*/*',
        'accept-encoding': 'zip, deflate, br',
        'accept-language': 'en-US,en;q=0.9',
        'origin': 'https://www.yelp.com',
        'referer': 'https://www.yelp.com/',
        'sec-ch-ua': '".Not/A)Brand";v="99", "Google Chrome";v="103", "Chromium";v="103"',
        'sec-ch-ua-mobile': '?0',
```

Figure 5. Screenshot of code 3

The Server file helps to connect to run the application on the web, which is performed using Flask. Flask is a micro web framework from Python to link Python to the interface made with HTML and CSS. The HTML builds the interface, the javascript makes the input and the button.

```python
from re import X
import numpy as np
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import *
import csv

#get data from our database in csv format
Data=[]
with open('database.csv','r') as d:

    reader=csv.reader(d)
    for r in reader:
        Data.append(r)
del Data[0]

testData=[]
Names=[]
links=[]
for d in Data:
    Names.append(d[0])
    links.append(d[len(d)-1])
    tempData=[]
    for i in range(1,4):
        tempData.append(int(d[i]))
    testData.append(tempData)
testNames=[]
for i in range(len(Names)+1):
    testNames.append(i)
del testNames [0]

model=svm.SVC()
model.fit(testData,testNames)
```

Figure 6. Screenshot of code 4

The predictor file as shown above helps to do the prediction of a better but similar restaurant. It checks for a restaurant's rating and categories to see if they fit the input restaurant and also gets the Yelp page of the restaurant and outputs it as the restaurant's name as the link to its Yelp page.

4. EXPERIMENT

Sklearn was the primary tool in Python that was used to create the model for the restaurant recommender. In particular, svm.SVC was the model responsible for testing the data of the database. SVC stands for Support Vector Classification, which allows the differentiation of different categories [11]. First, a database that was stored in a CSV (comma-separated values) file would be read and appended to a list of data in Python [9]. Then, the names and links for each piece of data would be stored in their respective lists, which are extracted by retrieving the first and last entries in the list that represents the data item, respectively. The svm.SVC model would be initialized and fitted with the test data and test names. After letting the model process the test data, the results are provided.

4.1. Experiment 1

To test the accuracy of the model we created for the application, the model was experimented on using different test cases. The test cases were created using three numbers, in which the first number represents the region, the second number represents the type of restaurant, and the third number represents the genre of restaurant.

Test Case 1: [4, 4, 4](region, type, genre)
Result: [32](Tip Top Meats)(3, 13, 5)

This test case consisted of #4 for all 3 categories. The predictor returned a most similar case of 3, 13, and 5. The 3 and 5 seem to be very close to 4 being only one off for each; however, the type was 13, which is 9 off of the original test case. For this test, it produced a restaurant that had 2 near perfectly matching categories; however, the 3rd one seems to be off.

Test Case 2: [1, 2, 1](region, type, genre)
Result: [61](Flip n Shake)(3, 10, 5)

This result seems different, the test input consisted of region #1, type #2, and genre #2. However, the results seem to be a bit scattered. The region returned was 3, which seems to be very close to 1, the type is 10, which is once again very far from 2, and 5 seems to be mid from 1. This could potentially be a coincidence, but the results of test cases #1 and #2 seem similar in some ways.

Test Case 3: [7, 8, 6](region, type, genre)
Result: [74](Noodle Boulevard)(4, 10, 5)

This result once again seems peculiar. The test input consisted of consecutive numbers of 7, 8, and 6. The return restaurant was of categories 4, 10, and 5. The 4 seems somewhat close to 7, the 5 is in near-perfect alignment with 6, and the 10 seems to be very close to 8. These tests have shown an expected pattern and an unexpected pattern or possible coincidence. The tests seem to return restaurants that have at least 2 categories that are quite similar to the input. However, all the returned restaurants have quite similar categories, this could be an unexpected issue that can be fixed, or it could be a coincidence with choosing our test cases. Overall, the analysis concludes that while the application seems to be somewhat effective at choosing a suitable restaurant for the user, there is still possible room for improvement regarding the model itself.

5. RELATED WORK

Graves Allen analyzed the relationship between a restaurant's ratings and reviews and the restaurant's profits. Their research showed that profits could rise by a considerable amount if the rating of a restaurant grew by even 1 star, and similarly profits could drop a considerable amount if the rating were to fall. The suspected reason for this is because review sites attract more people to a restaurant with good ratings and steer people away from restaurants with bad ratings [1]. Allen's work is similar to this work in that the main topic revolves around restaurants and their data. However, Allen chooses to place more focus on the correlation between ratings and profits, while we choose to create an application with a model that helps its users choose the best restaurant for them.

The EHL Faculty decided to analyze why people choose to look at review sites for restaurants. There appeared to be 4 main reasons. One is because then they can rest easy knowing that they won't regret their purchase at a certain restaurant. Another is that they can also look through the restaurant's menu and decide what they want and what they should order beforehand. Another is that people can see what other people's impressions of the restaurant were and then they know if they should eat at that restaurant [2]. What the EHL Faculty's work and our work have in common is that restaurants and their reviews are analyzed. The EHL Faculty uses the data to select what food item should be selected in a menu; on the other hand, we use the data to choose which restaurant to go to.

6. CONCLUSIONS

To provide a convenient method for people to find the restaurant that would suit them the most, an application was created to help them search for new places to eat at. This application features a simple interface created in HTML and CSS, and the back end was created using Python [15]. To test the application's accuracy in identifying which restaurant would be best for a particular user, an experiment was run on the model. Multiple test cases were done on the model using numeric values to represent the region, type, and genre of the restaurant. The results of the experiment seemed to indicate that while the recommendations that were provided based on the inputted numerical values seemed to be somewhat accurate in one or two components, at least one component seemed to have a noticeably large discrepancy in the majority of test cases.

One major limitation that can be seen with the application is its lack of features. While the application can recommend new restaurants to people and open them up to new places, it does not offer anything else to its users. This is fine if the application is specifically only for one purpose, but a lack of features makes the application less versatile and less able to satisfy all of its users' needs regarding restaurant recommendations. In the long run, this may mean that users will either have to use multiple applications for viewing other aspects such as recommended menu items or switch to another application entirely.

If I were to keep working on it, however, I would add more features. Some features that could be added in the future are extracting positive reviews from other websites and linking the prediction to the recommender itself. By seeing what exactly is being said in the positive reviews, the users may have a better idea of what exactly they should order at the restaurant. Overall, I believe the restaurant recommender turned out well.

REFERENCES

[1] Cotter, Michael, and Wayne Snyder. "How guide books affect restaurant behavior." Journal of Restaurant & Foodservice Marketing 3.1 (1998): 69-75.

[2] Zhang, Ziqiong, et al. "The impact of e-word-of-mouth on the online popularity of restaurants: A comparison of consumer reviews and editor reviews." International Journal of Hospitality Management 29.4 (2010): 694-700.

[3] Kim, Woo Gon, Jun Justin Li, and Robert A. Brymer. "The impact of social media reviews on restaurant performance: The moderating role of excellence certificate." International Journal of Hospitality Management 55 (2016): 41-51.

[4] Zeng, Jun, et al. "A restaurant recommender system based on user preference and location in mobile environment." 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI). IEEE, 2016.

[5] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

[6] Bisong, Ekaba, and EkabaBisong. "Introduction to Scikit-learn." Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners (2019): 215-229.

[7] Van Der Walt, Stefan, S. Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." Computing in science & engineering 13.2 (2011): 22-30.

[8] Zheng, Chunmei, Guomei He, and Zuojie Peng. "A Study of Web Information Extraction Technology Based on Beautiful Soup." J. Comput. 10.6 (2015): 381-387.

[9] Hunt, John, and John Hunt. "Working with CSV Files." Advanced Guide to Python 3 Programming (2019): 241-247.

[10] Snider, L. A., and S. E. Swedo. "PANDAS: current status and directions for research." Molecular psychiatry 9.10 (2004): 900-907.

[11] Komer, Brent, James Bergstra, and Chris Eliasmith. "Hyperopt-sklearn." Automated Machine Learning: Methods, Systems, Challenges (2019): 97-111.

[12] De Smedt, Tom, and Walter Daelemans. "Pattern for python." The Journal of Machine Learning Research 13.1 (2012): 2063-2067.

[13] Van Rossum, Guido, and Fred L. Drake Jr. Python tutorial. Vol. 620. Amsterdam, The Netherlands: Centrum voorWiskundeen Informatica, 1995.

[14] Yadav, Pallavi, and Paras Nath Barwal. "Designing responsive websites using HTML and CSS." international journal of scientific & technology research 3.11 (2014): 152-155.

[15] Park, Thomas H., Brian Dorn, and Andrea Forte. "An analysis of HTML and CSS syntax errors in a web development course." ACM Transactions on Computing Education (TOCE) 15.1 (2015): 1-21.