YourFood: Machine Learning Recommendations for Restaurants

Developed and written by Pranav Wadhwa

Thomas Jefferson High School for Science and Technology

May 2020

Abstract

Today's advances in machine learning (ML) technology enable Artificial Intelligence (AI) developers to teach machines how to make predictions based on past behaviors. This inductive style of learning gives machines the ability to cater to the needs of each user. With this technology, the average user should not have to scroll through lists of dozens of restaurants to find one they like. This paper examines a machine learning approach to providing recommendations for restaurants based on users' tastes. This solution is provided through a mobile app which will learn the preferences of its user and provide a list of restaurants along a certain route where they would enjoy eating.

Purpose

The purpose of this project is to replace the unnecessary work of scrolling through lists of restaurants and guessing which one is the best fit. This mobile app makes it easy for users to find places they will like by matching their tastes with local restaurants. Especially when traveling and visiting new cities, the app will people overcome an unfamiliar list and discover the unique tastes the city has to offer.

Background

Other apps

While popular applications such as Yelp and Google Maps[1] help people find places to eat near them, they simply give a list of local restaurants. These apps do not personalize recommendations but rather present the user with a list typically organized by location.

[1] Google Maps later added a recommendations feature in their mobile app, but this was not available at the time the research began.

Machine Learning Text Vectorization

A crucial aspect of creating ML models is retrieving, compiling, and/or creating datasets that a computer can learn from. Often times, datasets consist of pieces of text, such as reviews of a restaurant. However, these text-based descriptions cannot be immediately parsed

by computers which prefer numbers as inputs. By representing pieces of text as vectors, ML algorithms can perform operations on these vectors. For instance, a common example of vector operations lies in the equation “King” – “Man” + “Woman” = “Queen”. Various parts of each word’s semantics are stored in multiple dimensions, as seen in this example where royalty and gender are stored in two separate dimensions. Thus, creating useful vectors to represent words has been a major challenge that ML scientists have been trying to solve for years.

A simple yet effective approach to this problem is through the bag-of-words (BOW) model. This model takes in a large piece of text and assigns each word in that text a numerical value based on its frequency in the text, essentially classifying the text as a “bag of words”. Handler et. al (2016) attempted to create an improved version of the BOW model by creating a Noun Phrase Finite-State Transducer (NPFST). The NPFST model effectively extracts more relevant phrases in the text by grouping phrases instead of just words. Overall, this strategy yields vectors with less likelihood of misinterpreting the text by taking words out of context.

Keyword extraction is an additional tool that can enhance text vectorization by removing unnecessary words (e.g. “the”, “a”) and prioritizing words that represent the text. For instance, Rapid Automatic Extraction Keyword (RAKE) is a Python library designed to extract the most important words and phrases out of a piece of text. By using these words and phrases instead of an unfiltered piece of text, the vector more accurately represents the text and not just common words like “the” and “a”.

Machine Learning Recommendations

J. Peña (2017) claimed that reviews about a place or product are more important than ratings when providing recommendations because they provide context and descriptions. Peña created Rich-Context, a recommender system that extracts context from reviews without defining any keywords in the text. The absence of definitions makes this strategy very convenient.

Huang (2017) describes another approach known as the cosine similarity analysis. A cosine similarity analysis is another strategy like Rich-Context that can create recommendations without definitions. Once a piece of text has been transformed into a vector of numbers, multiple vectors can be compared to each other using a cosine similarity test, which yields a numerical value between 0 to 1 of how close the numbers in those two vectors

are. Huang shows how this analysis can be used in Python to find the similarity between political text documents.

Development tools

Flask is a web framework built in Python that allows for the creation of web servers in Python. This foundational framework has helped build web servers for Netflix, Airbnb, Lyft, and many more sites. PythonAnywhere is a hosting platform used to deploy web apps built with Flask and Python.

Zomato is a database of restaurants across the globe, and they provide an Application Programming Interface (API) that allows developers to access information about these restaurants. Their API endpoints take in certain parameters about the restaurant search, such as location and name, and return a list of JavaScript Object Notation (JSON) objects that contain information about each restaurant, such as its name, address, cuisines, and reviews.

Firebase is a Google-backed platform that has many backend features for developers. In particular, the Firebase Realtime Database gives developers a large database in which they can store JSON objects. This is often useful for storing user data which can be accessed and modified from both Python web servers and iOS applications. Pandas is a Python library that converts 2D lists of data into an easy-to-use, tabular form, and this library can be used for modifying and converting data from Firebase.

Development

Data Retrieval

A crucial component of this project was restaurant data, especially reviews that can reveal a great deal of information about a restaurant. Several data sources were examined, including the Yelp, Google Maps, HERE, and Zomato. Due to the availability of restaurant reviews without cost, I chose to use the Zomato API and developed a Python script to download reviews and basic information from 1,000 restaurants from ten American cities and store them in a CSV file. For each city, the script would make a request to the Zomato API, retrieve a JSON object, and parse the object's data into a row. Here are a few rows from the data:

	Restaurant	ZomatoID	ZomatoURL	Cuisines	ReviewText
0	Spread Deli & Bottles	16862776	https://www.zomato.com/campbell-ca/spread-deli...	American, Sandwich	"This is a good choice for lunch. I had the to...
1	Urbanbelly	16739904	https://www.zomato.com/chicago/urbanbelly-west...	Vietnamese	"My wife and I stopped here for lunch while wa...
2	Rock Bottom Restaurant & Brewery	16858547	https://www.zomato.com/campbell-ca/rock-bottom...	Bar Food	"Amazing food and out waitress rocks! The ribs...
3	Booty's	17033306	https://www.zomato.com/surprise-az/bootys-1-su...	American, Burger	"Lousy food and service won't be going back tr...
4	La Madeleine	16944002	https://www.zomato.com/irving-tx/la-madeleine-...	French, Bakery	'Great place for for meal with good options of...

Machine Learning: Text Processing

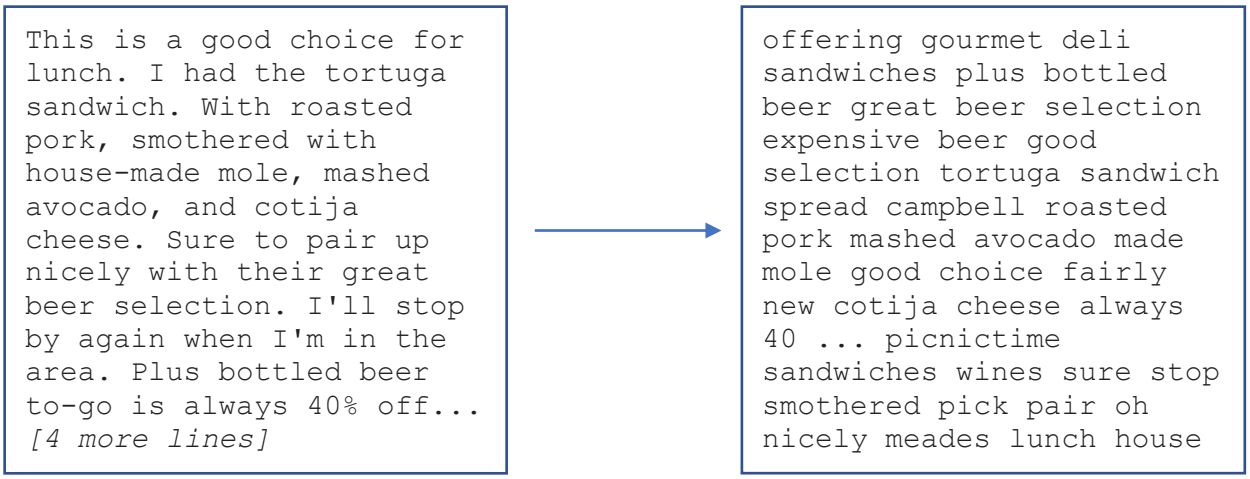
Now that I had data ready, I created a Google Collaboratory notebook to write Python code for the machine learning. My first step was to use RAKE, a Python library for text extraction. When comparing reviews from two restaurants to determine how similar they are, the important information is the food, not words like “this”, “the”, “with”, etc. If those extraneous words are left in, two reviews that have the word “the” several times may be seen as more similar than they should be. RAKE has two functions to solve this problem: `get_word_degrees` and `get_ranked_phrases`. The first method (word degrees) returns a dictionary of important words along with an assigned score:

```
{'oh': 1, 'tortuga': 2, 'sandwich': 2, 'gourmet': 4, 'deli': 4, 'sandwiches': 5, 'beer': 8, ....}
```

The second method (ranked phrases) returns an ordered list of the most important phrases found in the reviews:

```
['offering gourmet deli sandwiches', 'plus bottled beer', 'great beer selection', 'expensive beer', ...]
```

Although I have not systematically compared the effectiveness of these algorithms, I found that using the ranked phrases typically yields better recommendations. For instance, here is a complete review parsed by RAKE’s get_ranked_phrases function:



Initial review Extracted phrases

As shown, the most important phrases about the food at the restaurant compose the entirety of the text on the right.

Machine Learning: Recommendations Engine

With the extracted phrases, I began creating the recommendations. To do this, I used the `CountVectorizer` and the `cosine_similarity` from Python's scikit-learn library. The `CountVectorizer` performs the text vectorization using a bag-of-words model. Because the words in the input text have already been extracted through RAKE, the output is more accurate than a standard bag-of-words, mimicking the NPFST described in the background. The `cosine_similarity` function runs a cosine similarity analysis, that compares how similar two vectors are to each other. I run this function on the set of restaurants to get an output that shows how similar each restaurant is to another:

	Restaurant 0	**Restaurant 1**	**Restaurant 2**	**...**	**Restaurant 97**	**Restaurant 98**	**Restaurant 99**
Restaurant 0	1.0000000	0.0969762	0.0754761		0.1577495	0.0737574	0.1427110
Restaurant 1	0.0969762	1.0000000	0.2971523		0.1979723	0.0969762	0.1979723
Restaurant 2	0.0754761	0.2971523	1.0000000		0.0912358	0.2115432	0.2448217
...							
Restaurant 97	0.1577495	0.1979723	0.0912358		1.0000000	0.1093241	0.1001380

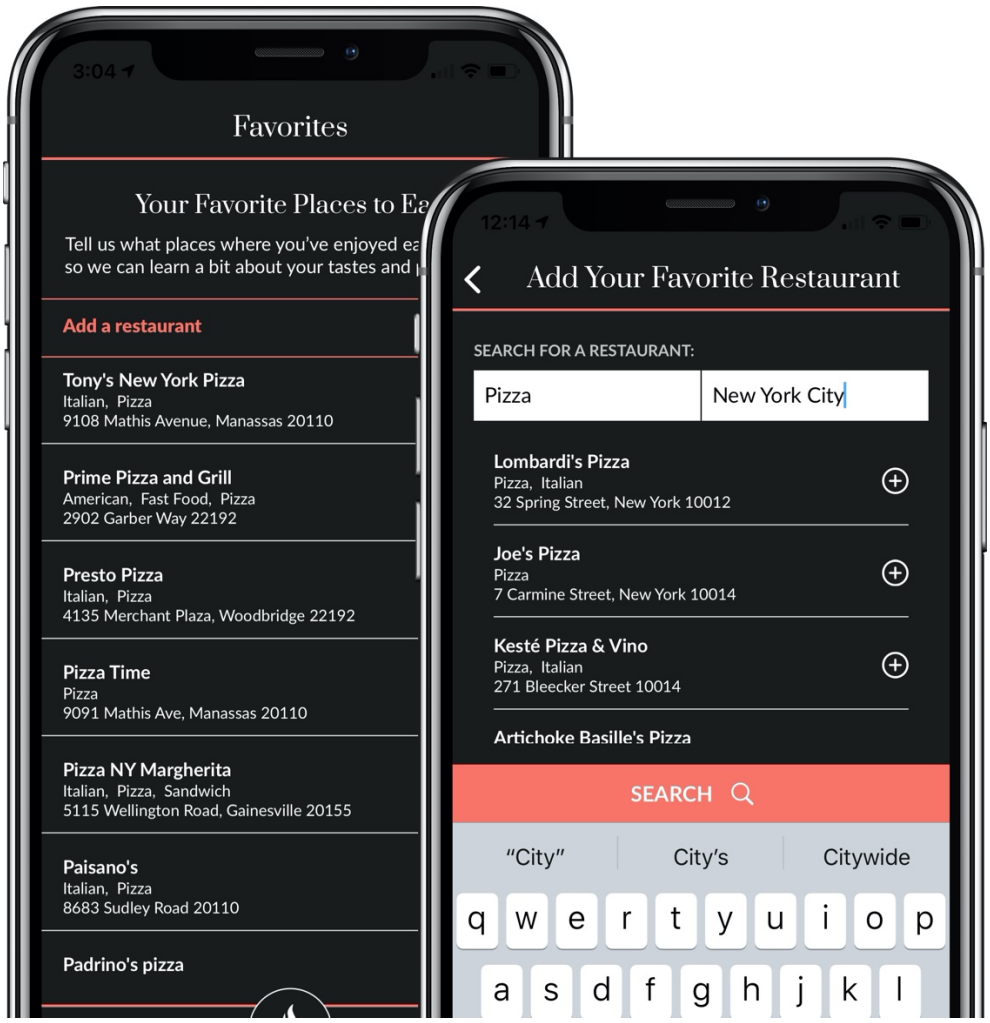
Restaurant 98	0.0737574	0.0969762	0.2115432		0.1093241	1.0000000	0.1968981
Restaurant 99	0.1427110	0.1979723	0.2448217		0.1001380	0.1968981	1.0000000

Backend Server

I added this machine learning code to a Flask server on Python Anywhere and set up a basic data upload system via Firebase. The mobile app would download reviews of restaurants the user has liked in the past along with reviews of local restaurants from Zomato and upload this data to Firebase's Realtime Database. Afterward, the server can take requests at http://pranavwadhwa.pythonanywhere.com/?dataid={uuid} where {uuid} is the unique ID of the data stored in Firebase. The web server downloads the reviews from Firebase and creates a table of reviews using Pandas. It then uses this table to compare the user's favorite restaurants with local restaurants and returns a score for each local restaurant based on the cosine similarity analysis.

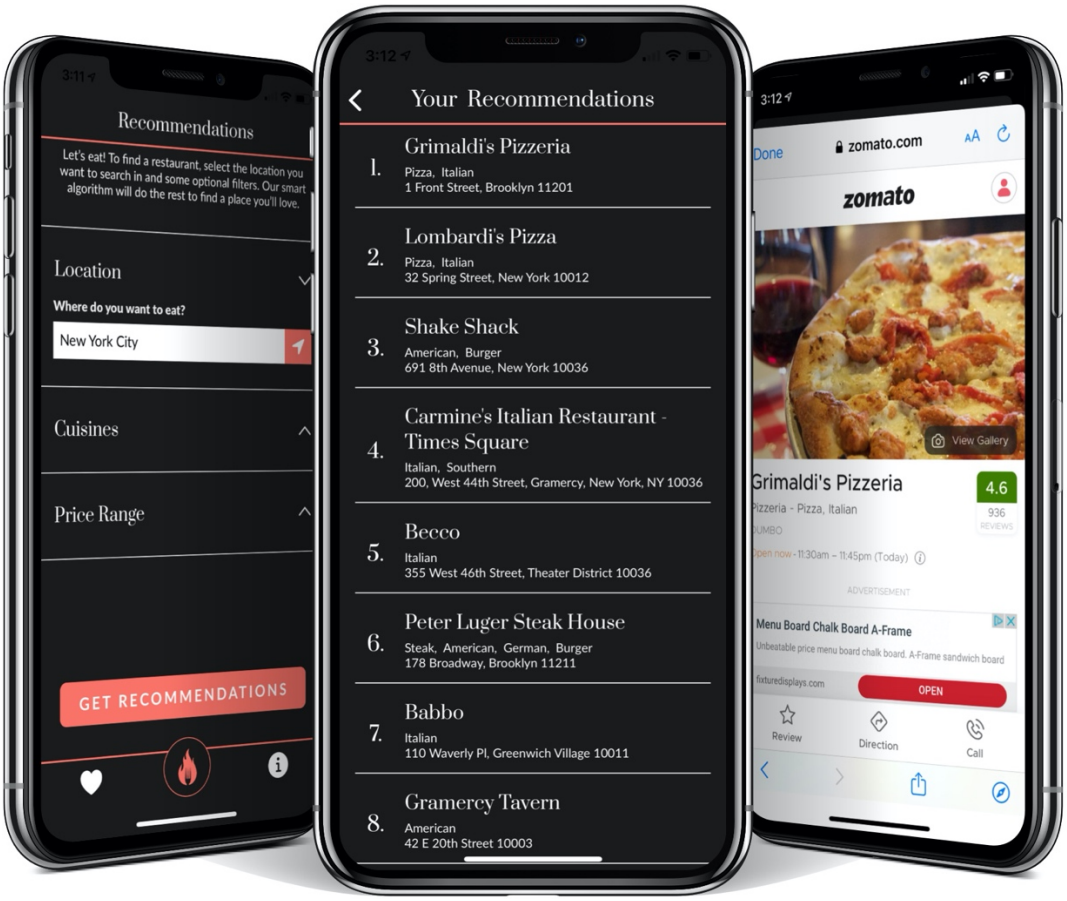
iOS App Development: Add Favorites

For the backend to make recommendations for the user, it must know what restaurants the user has liked in the past, so it can find similar ones. When searching for a restaurant, the user must provide both the name and the location of the restaurant. The app uses OpenCage geocoding to transform the location (e.g. "New York City") into a set of coordinates (40.7128° N, 74.0060° W) that can be sent to the Zomato request. The Zomato ID, name, reviews, and basic information of the restaurants that the user selects are stored on the iOS device using CoreData. The following screenshots demonstrate this process:



iOS App Development: Recommendations

Once they have added their favorite restaurants, users can then search for recommendations in a particular city. While the cuisines and price range features have not been added yet, they also can filter the results to help the user find exactly what they want. Here, the app downloads a list of twenty local restaurants along with their reviews and uploads this data along with the favorites data to Firebase. It then makes a request to the Python Anywhere server, which sends back a score for each of the local restaurants. The app then sorts this and displays them in a list, sending users to the restaurant's Zomato website if they click on it (see screenshots below for details).



As we selected several Italian restaurants and pizzerias as favorites earlier, the recommendations now include other Italian restaurants in the area.

Discussion

Overall, YourFood makes it easier for anyone to find what they want to eat quickly. Due to the limits of the dataset and server requests, the app will not be published on the App Store, but it stands as a proof-of-concept for how machine learning recommendations can enhance and personalize our everyday lives.

References

Handler, Abram & Denny, Matthew & Wallach, Hanna & O'Connor, Brendan. (2016). Bag of What? Simple Noun Phrase Extraction for Text Analysis. 114-124. 10.18653/v1/W16-5615.

Huang, L. (2017, March 30). Measuring Similarity Between Texts in Python. Retrieved May 25, 2020, from https://sites.temple.edu/tudsc/2017/03/30/measuring-similarity-between-texts-in-python/

Peña, F. J. (2017, August). Unsupervised Context-Driven Recommendations Based on User Reviews. Retrieved from ACM Digital Library database.