Kristi Bui, Moustafa Abdelaziz, Aasish Basani, Vidhi Gondalia
DS 4300 Large Scale Information, Storage & Retrieval
Professor John Rachlin
April 14, 2020

Foodies Final Report: Food Recommendation System

Project Goals

Our project centers around building a food recommendation system, ideally one that differs from traditional food recommendation applications. Our initial goal was to design a recommendation system meant for health-minded people, ideally by utilizing metrics such as macronutrients and caloric content of dishes at restaurants to provide better recommendations to users on what to order. However, due to the lack of readily available API resources to achieve these metrics, we revised our plan.

Our new goal was to build a food recommendation system centered around recommending dishes at establishments users already enjoy ordering from. We built out a few recommendations through this new approach, such as recommending dishes at the same restaurant based on those who already like your favorite dish, recommending dishes at the same restaurant based on similar category of food, and recommending dishes at the same restaurant based on what your friends also order at that restaurant. Essentially, this would be akin to providing recommendations of similar entrees that users may like, and/or appetizers/sides that may pair well with a favorite dish that the user already had at that restaurant.

We initially started using a small data sample of 50 users with about 10 restaurants, of Mexican and Italian cuisine. This was to test and ensure that our recommendation cyphers were working properly. Once that was validated, we moved to scaling up our data. Our goal was to simulate the Boston restaurant scene, so we utilized US Restaurants Menus API to query restaurant and menu data for around 200 restaurants of all different types of cuisines in the Boston area. We attempted to simulate the Northeastern population of a given semester, so we generated around 5,000 users programmatically.

Motivation & Significance

Our primary motivations were to create a different and personalized recommendation system for food orders, as well as explore graph databases/Neo4J and how to build recommendation systems through them. We found that food would make an interesting option for a recommendation system given historical recommendations for new dishes and restaurants to try. Trends we have often noticed include observing food, hospitality and travel applications that recommend based on similarities between entities (e.g.

recommending restaurants to try based on similar genre, price range, area, etc.), and real life situations such as users asking servers for food recommendations at restaurants, as well as users asking their friends for ideas of foods to try.

As such, we proceeded to develop a few recommendation models that focused on recommending within the same restaurant that users were already ordering from so that we could build upon the information of what users already knew they liked. We were able to build three recommendation cyphers revolving around this information:

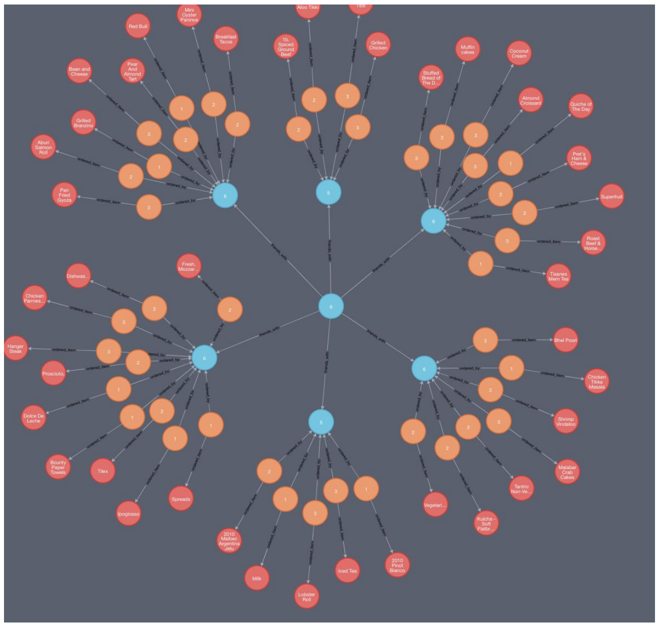
1. `Recommendation #1: Recommend a dish to a user from the same restaurant where they order the most from, by finding and recommending dishes that other users enjoy who have similar tastes to this user`
2. `Recommendation #2: Recommend a dish to a user from the same restaurant where they order the most from, by finding dishes of a similar category to this dish`
3. `Recommendation #3: Recommend a dish to a user from the same restaurant where they order from the most, by finding dishes that their friends like to order from this restaurant`

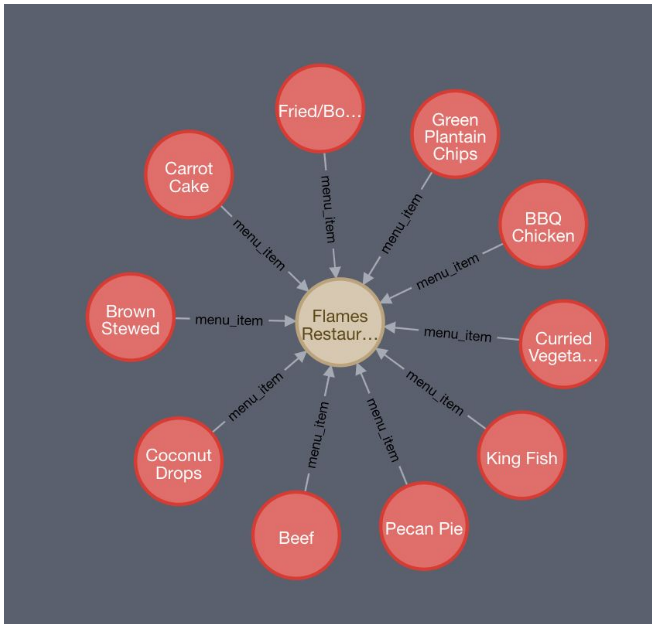
Our data model was flexible enough that we were also able to implement other recommendations, some more akin to traditional recommendation applications, although we did not include them in our full-stack application as of now:

4. `Recommendation #4: Recommend the top most liked dishes of your friends`
5. `Recommendation #5: Recommend the top-most ordered dish at the same restaurant you like (not including yours)`
6. `Recommendation #6: Recommend another top dish of a similar restaurant category that you like`
7. `Recommendation #7: Recommend a dish based on people who also like your favorite dish also order (not limited to your favorite restaurant)`

Note: The full cypher code for these 7 recommendations can be found in the recommendations.cql file.

System Architecture


User (blue), Times Ordered (orange), Menu Item (red)



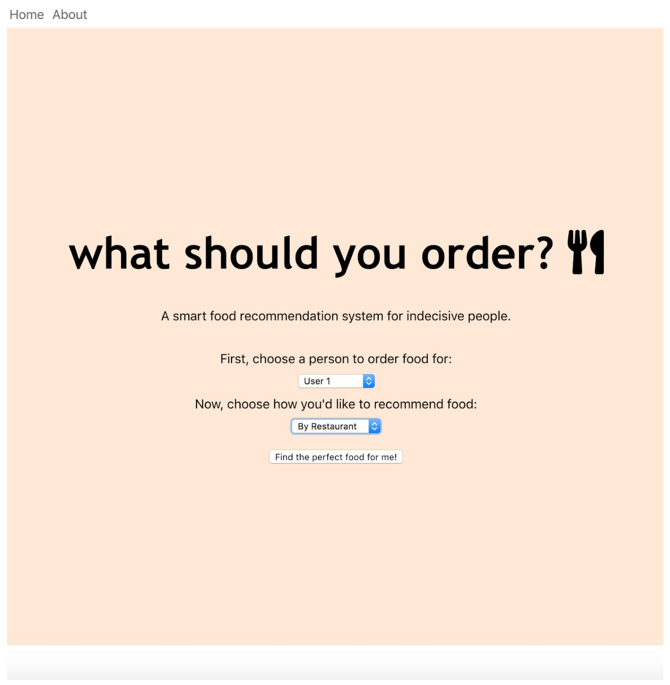
Restaurant (beige), Menu Item (red)

The raw data for restaurants and their corresponding menu items was chosen from the US Restaurant Menu API. We utilized Neo4J, a graph database, as our data store. Utilizing Neo4J allowed us to create our different recommendation cyphers that traversed our node structures to look for dishes to recommend. For the data we auto-generated user data (Initially 50 then 5000) and assigned friends where each user could have 1 or upto half the total population number of friends.

In order to present our recommendation service as an application, we wrote a simple backend using Python and the Flask micro web framework. Specifically, we created a driver in Python to run the necessary cyphers from our database, and then mapped the resulting data into a simple REST API server with Flask. Our frontend was written in React/ES6 and presents a simple user interface to show how picking different types of recommendations for different users might work. The frontend utilizes the API endpoints from our Flask server to pull the food and user data (now in JSON format) and render it on the client side.

Visualization

Selecting user & recommendation



Home About

what should you order? 🍴

A smart food recommendation system for indecisive people.

First, choose a person to order food for:

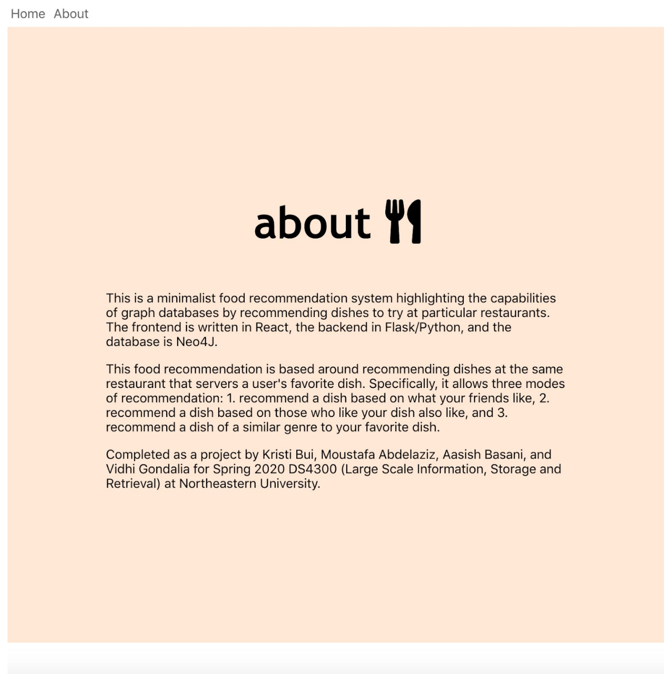
User 1

Now, choose how you'd like to recommend food:

By Restaurant

Find the perfect food for me!

About Page



Home About

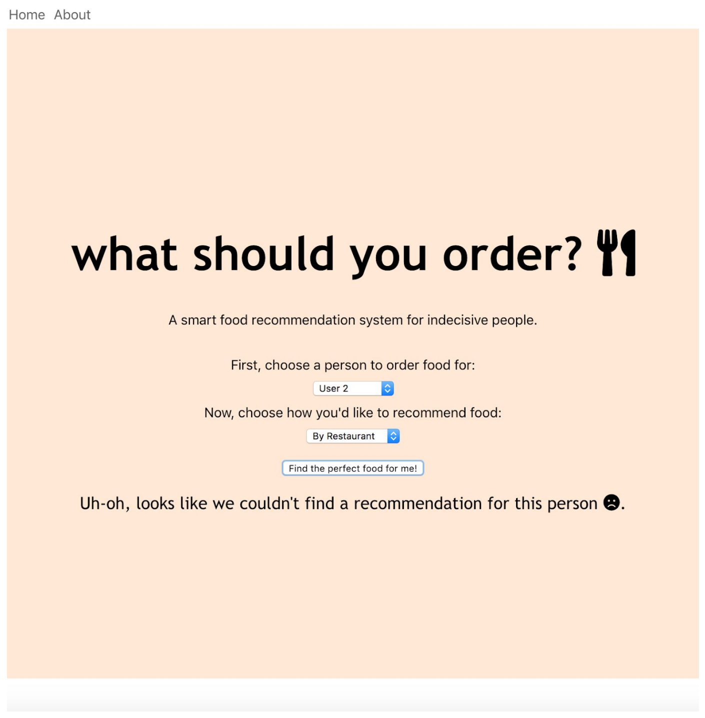
about 🍴

This is a minimalist food recommendation system highlighting the capabilities of graph databases by recommending dishes to try at particular restaurants. The frontend is written in React, the backend in Flask/Python, and the database is Neo4J.

This food recommendation is based around recommending dishes at the same restaurant that servers a user's favorite dish. Specifically, it allows three modes of recommendation: 1. recommend a dish based on what your friends like, 2. recommend a dish based on those who like your dish also like, and 3. recommend a dish of a similar genre to your favorite dish.

Completed as a project by Kristi Bui, Moustafa Abdelaziz, Aasish Basani, and Vidhi Gondalia for Spring 2020 DS4300 (Large Scale Information, Storage and Retrieval) at Northeastern University.

Successful & Unsuccessful Recommendations



Home About

what should you order? 🍴

A smart food recommendation system for indecisive people.

First, choose a person to order food for:

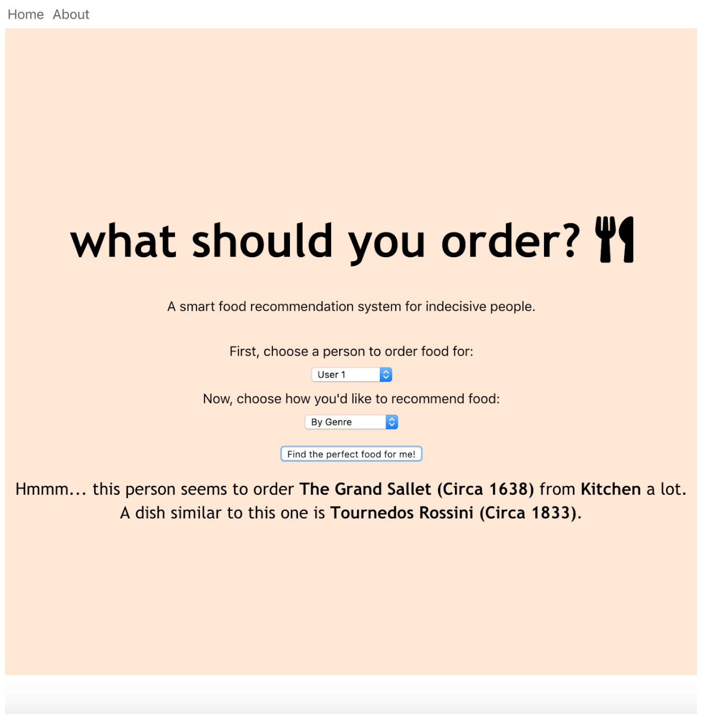
User 2

Now, choose how you'd like to recommend food:

By Restaurant

Find the perfect food for me!

Uh-oh, looks like we couldn't find a recommendation for this person ☹.



Home About

what should you order? 🍴

A smart food recommendation system for indecisive people.

First, choose a person to order food for:

User 1

Now, choose how you'd like to recommend food:

By Genre

Find the perfect food for me!

Hmmm... this person seems to order **The Grand Sallet (Circa 1638)** from **Kitchen** a lot. A dish similar to this one is **Tournedos Rossini (Circa 1833)**.

Results & Conclusions

We were able to implement a functional recommendation system that recommends dishes at the same restaurant that one already orders from using a variety of metrics to query on. Querying on patterns such as what other users order based on similarities between their ordering preferences and your own adds another layer of personalization to food recommendations. Alas, this would be a useful service integrated in part with a service such as an online meal delivery service, where when a user selects a dish to order, could also recommend a side dish or appetizer that may pair well with it given that other users who order the same dish also get that dish.

Much of the current limitations on the results of our project revolve around our data. As stated above, this recommendation algorithm would ideally be paired with an application that would retrieve and store user information about which restaurants they order at and at what quantity. As we were not building out that service, nor did we have the real life users to create that sample, we instead simulated our data by generating users and their orders to correspond to the food and restaurant data from the API. We took steps in our generation process to create as realistic relationships as possible (relationships between users, the quantities of food they order and which restaurants and dishes they order); however it would be much more precise if we had real user data that encapsulated real people's ordering and eating habits. A service such as GrubHub or Postmates, for instance, that could keep track of real-life orders made by a user, would work well for more precise results. Unfortunately, that data was not public at the time of this project creation.

One other limitation our project had was in scaling versus our local hardware capacity. As mentioned prior, we were looking to simulate ordering and the relationships between users who order food to the Boston (specifically Northeastern University population). Therefore, our initial plan was to simulate 10,000 users (approximately half the size of the undergraduate population) against some 200 restaurants in the Boston area. However, loading these users to Neo4J - specifically building the friendships between this many users - was not possible on our local machines. Consequently, we scaled back the number. A future exploration of this project may consider looking into finding better resources to support this so that the user base is more reflective of the actual demographics at Northeastern.