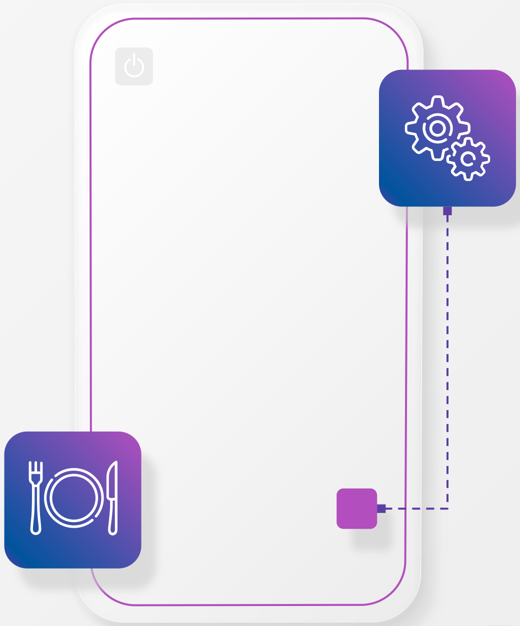
Restaurant Recommender

Daniil Abbruzzese



Agenda

01
Introduction
My motivation for selecting my idea

02
Process
How I created my data product

03
Live Demo
Generating restaurant recommendations

04
Next steps
Incorporating content based filtering

The Problem

Choosing which restaurant to eat at can be **overwhelming** due to the sheer amount of choices.

Googling "best restaurants in my city," yields too many **generic results** that are not personalized to the individual

As a result, I end up going to the same few restaurants that I discovered through word of mouth.

Solution: personalized recommendations

I have created a collaborative filtering model, trained on the **Yelp Open Dataset**, that delivers **personalized restaurant recommendations** to users.

My recommendation system learns what users like and dislike in a restaurant, whether or not they know it themselves. It thus helps users **expand their horizon**.

Process

Preparing that Data

Filtering my data and generating a user-item matrix

Model Building

Selecting a collaborative filtering model and tuning hyper parameters

Recommendations

Despite being red, Mars is actually a cold place. It’s full of iron oxide dust

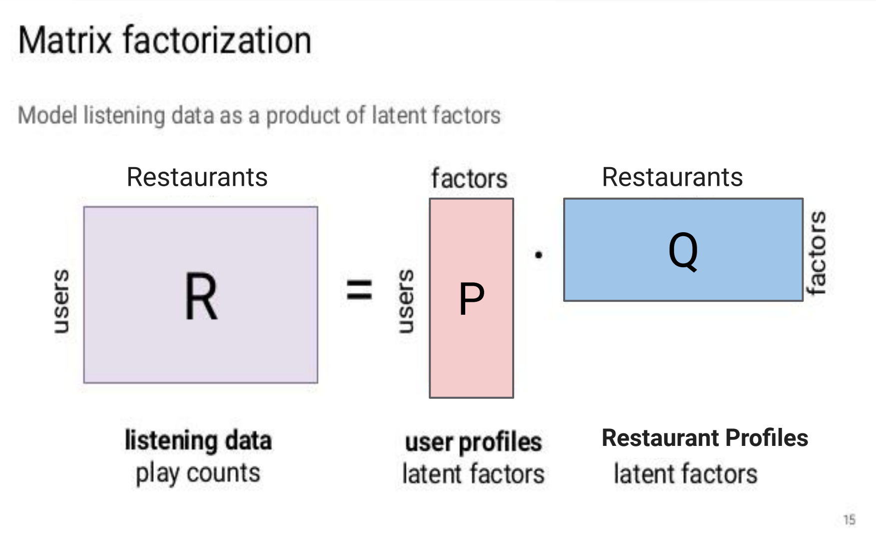
Preparing the Data

Preparing the Data

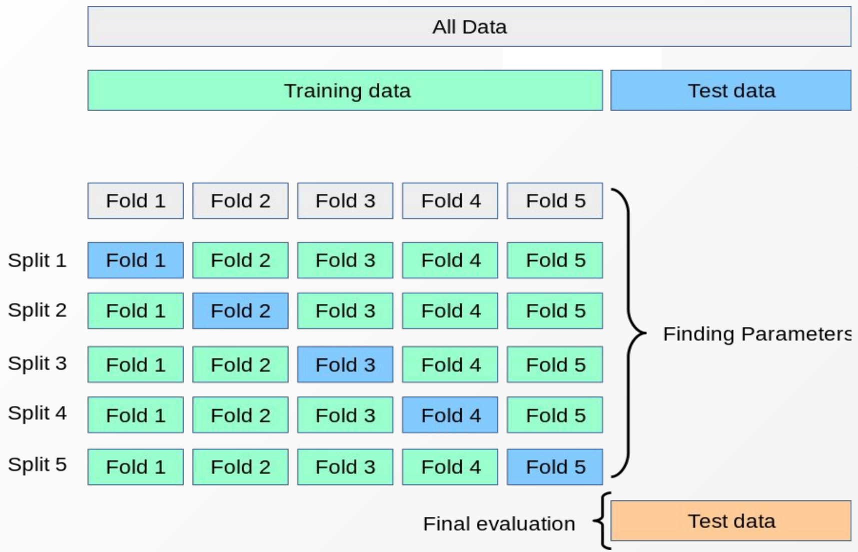
1. Filters: “Restaurants” in California only
2. Merging reviews dataset with business dataset
3. Users who left at least 5 reviews
4. Averaging ratings for users who left multiple reviews at same restaurant
5. Pivoting the df to get a UI matrix

Model Building

Matrix Factorization with Stochastic Gradient Descent

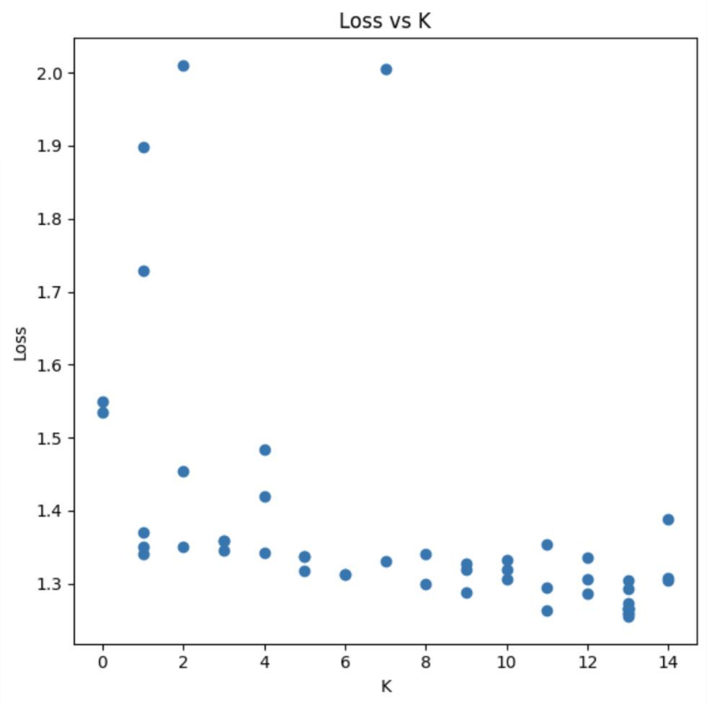


- Goal is to discover relationship to “Latent Factors” for users and restaurants
- Represented by Matrices P and Q
- Taking the dot product of P and Q yields an estimate of R which minimizes the loss
- Train / test split: for each user, I removed at least 1 rating for testing
- Compares predicted rating to actual to calculate the loss, adjusts values in the direction opposite to the loss function



Hyper Parameter Tuning

- Used cross validation to identify ideal hyper parameters
- Hyper parameters:
  - K: # Latent features
  - Learning Rate: size of the steps taken during gradient descent optimization
  - Regularization term: penalizes the magnitude of the parameters to avoid overfitting
  - Number of iterations: Number of times model updates values

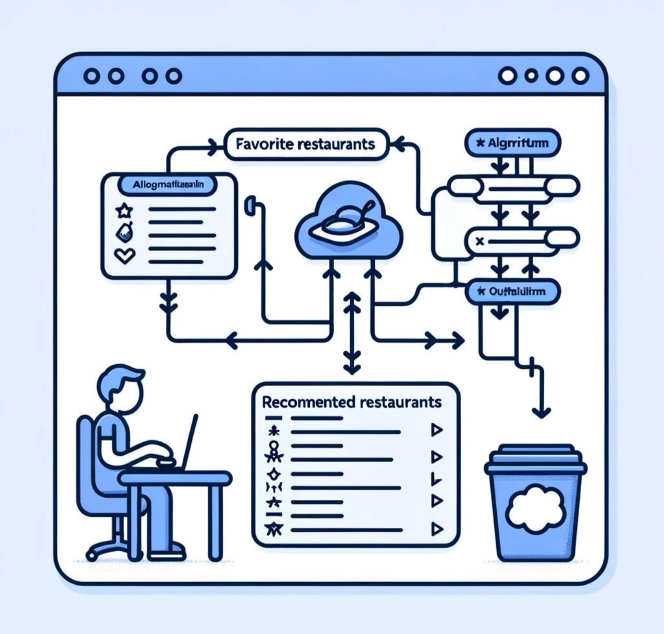


Bayesian Optimization

- I found that grid search was taking too much time
- Used Bayesian Optimization to guide the search for best hyper parameters
- Uses a probabilistic model to predict which set of hyper parameters will perform best on the validation set based on past results
- Reduces number of evaluations needed by focusing on areas in grid space with highest potential for improvement, making it much faster

Recommendations

Delivering Recommendations



1. Retraining model on all data
2. User inputs favorite restaurants in a list
3. List is converted to a user ratings vector (row in UI matrix)
4. A user latent feature vector is derived given our restaurant latent feature matrix (Q)
5. We predict ratings across all restaurants by taking dot product of user's latent feature vector and restaurant latent feature matrix
6. Sort by 10 highest ratings, giving recommendation

Live Demo

Next Steps: Hybrid Model

"I want a lively, high-energy restaurant that is perfect for a large group birthday dinner"

"My favorite restaurants are X, Y, and Z"

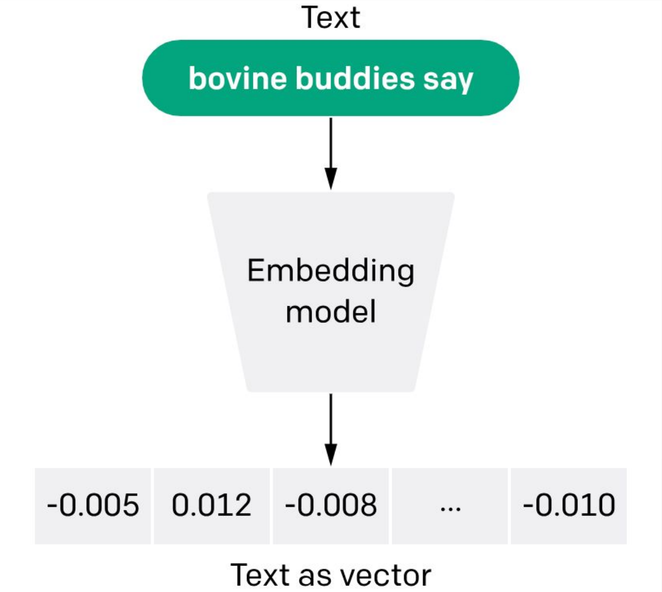
Hybrid Model

Incorporate **content-based** filtering into the recommendation system.

Users can filter for the **type of atmosphere** they want (Date night, birthday dinner, calm vs lively atmosphere, etc).

Open AI's embeddings can help with this. Embeddings turns text into a high dimensional vector representation that captures semantic meaning

High-level plan



1. New user types in their desired atmosphere
2. Convert this input into an embedding
3. Compute the cosine similarity between the user input embedding and the embeddings of various restaurants
4. Rank Restaurants: Rank the restaurants based on their similarity score with the user's input.
5. Hybrid model integration: of the top 30 restaurants that match the desired atmosphere, recommend top 5 for predicted score from collaborative filtering model

Obtaining one embedding per restaurant

Approach one:

- Filter for reviews that have keywords relating to ambiance
- Generate embeddings for each, take the average
- **Downside:** will miss out on reviews that don’t contain keywords but still convey information on ambiance

Approach two:

- Use GPT 3 or 4 to read all reviews and generate a single summary description that captures ambiance
- Convert summary caption into an embedding
- **Downside:** more timely and costly

Thank You