CSE 6242 - Team 88 - Final Project Poster
Restaurant Finder

Introduction

Problem
Current restaurant recommendation methods uses content-based filtering that recommends restaurants based on user's previous restaurant ratings and restaurant features, thus the recommendations are limited to the KNOWN interests of the users

Solution

Our **Web Application** allows the users to select cuisine, type of restaurant and location to generate recommendation based on the preferences of similar users.

Back-end model uses a weighted combination of rating generated from **'Scikit-Surprise'** based recommendation prediction AND a polarity score generated from **'NLTK and SentiWordNet' library** based sentiment analysis to compute a **relevance score**.

Motivation(Why should we care?)
Yelp's impressive reach - 73 million active unique visitors and 6.3 million active businesses.
Consumer-facing web application - attempts to derive customer value based on their preferences and engagement with local restaurants.
The application also helps businesses reach more customers who love their goods and services

Approaches

What are they?
We use mix of two approaches to generate restaurants recommendations -

1: Collaborative filtering at scale using large dataset: generating rating for the selected user by identifying similar users and showing restaurants what those user have rated highly.

2: Sentiment Analysis to create a sentiment score for each restaurant that gauges the tone and emotion of reviews.

The sentiment score provides a more granular grading system without making to choose from a limited list as above **AND** helps tackle the skewness of numerical ratings data in 1.

How do they work?

1: Collaborative filtering: **Proof of Concept (POC)** model was created to select the best performing algorithm among KNNBasic, KNNWithMean, KNNWithZScore, KNNBaseline, SlopeOne and NormalPredictor algorithms.

POC model was trained using above algorithms on training data and tested on test data using Root Mean Square Error (RMSE) for model performance.

KNNBaseline was selected as the **best algorithm.**
KNNBasic creates a prediction by aggregating the predictions of all similar users and dividing by number of users. KNNBaseline add a baseline rating for each user which attempts to capture the effect that some users may systematically rate all restaurants higher or lower than others.

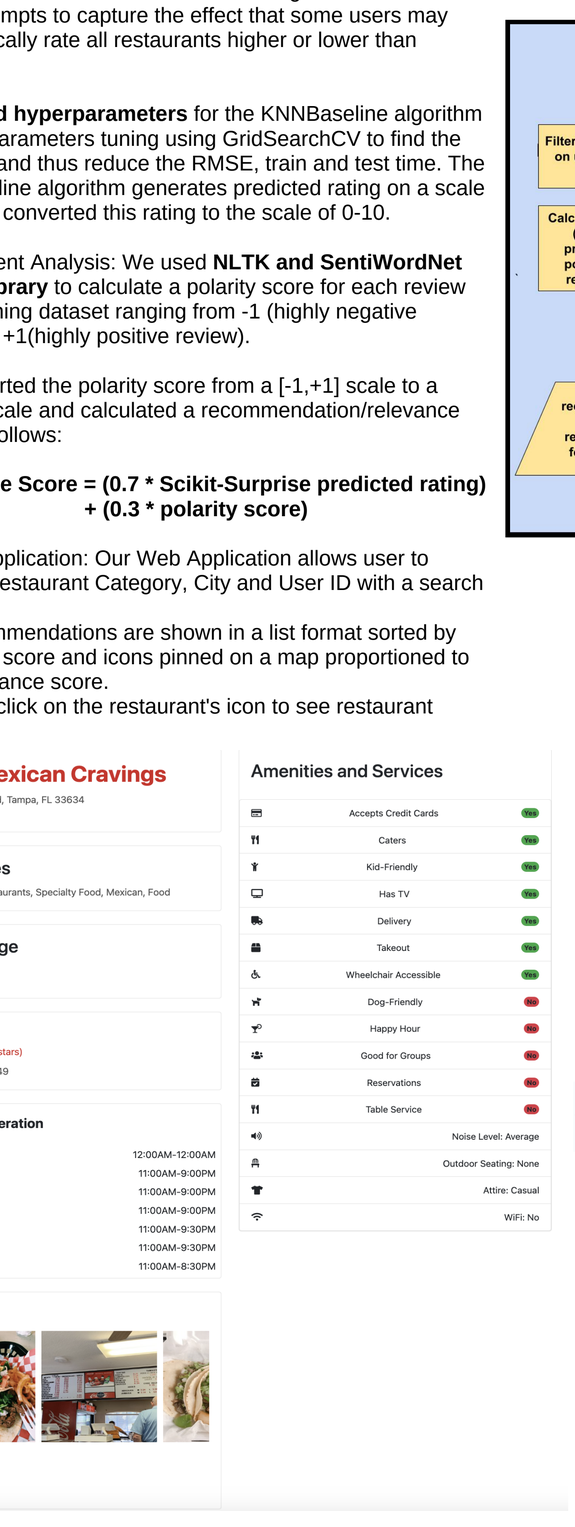
Optimized hyperparameters for the KNNBaseline algorithm by hyperparameters tuning using GridSearchCV to find the optimal k and thus reduce the RMSE, train and test time. The KNNBaseline algorithm generates predicted rating on a scale of 0-5.We converted this rating to the scale of 0-10.

2: Sentiment Analysis: We used **NLTK and SentiWordNet Python library** to calculate a polarity score for each review in the training dataset ranging from -1 (highly negative review) to +1(highly positive review).

We converted the polarity score from a [-1,+1] scale to a [-10,10] scale and calculated a recommendation/relevance score as follows:

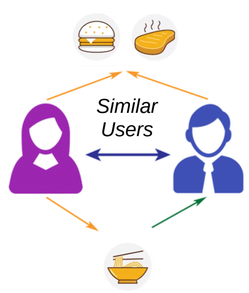
Relevance Score = (0.7 * Scikit-Surprise predicted rating) + (0.3 * polarity score)

3: Web Application: Our Web Application allows user to select a Restaurant Category, City and User ID with a search button.
The recommendations are shown in a list format sorted by relevance score and icons pinned on a map proportioned to their relevance score.
User can click on the restaurant's icon to see restaurant details.



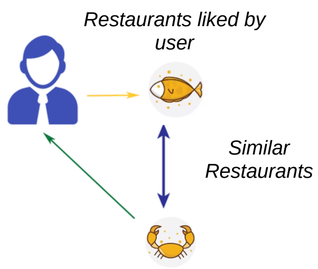
Collaborative Filtering

Restaurants liked by both users

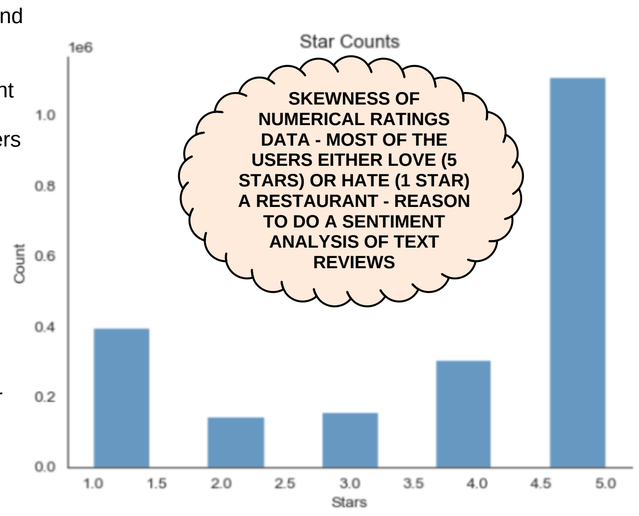


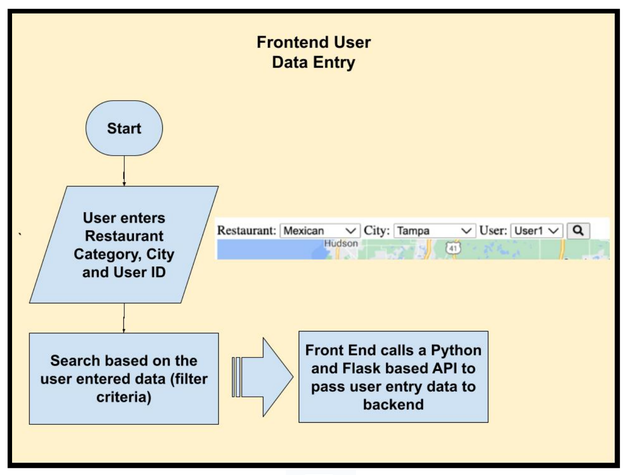
Visited by her, recommended to him!

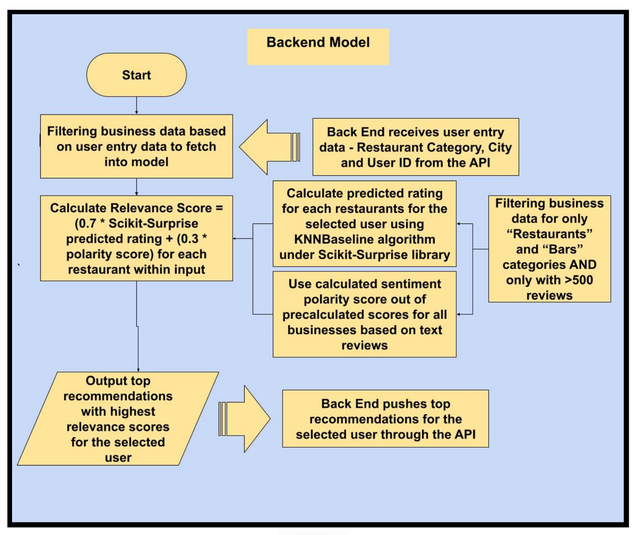
Content-Based Filtering

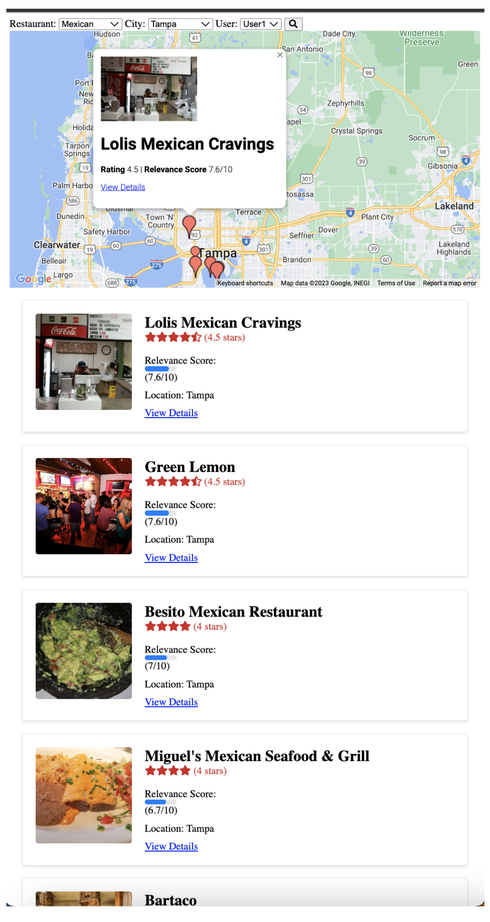


Source: https://www.researchgate.net/figure/Content-based-filtering-vs-Collaborative-filtering-Source_fig5_323726564









How do they solve the problem?
A test that indicates that the model works - We manually inspected the recommendation generated for a user. The user had visited 8 out of 10 recommended restaurants. The unvisited ones serve international cuisines – French and Italian. The user has rated restaurants serving international cuisine very highly. In the training dataset, the user has also written a comment about "loving" a café with a French name.

What is different?
The recommendations generated based on our mixed approach are not limited to the known interests of the users thus allows more variance on recommendations based on similar users and skewness correction of rating data with text reviews to certain extent.

Data

How did you get it?
We got the yelp business, reviews and user data from https://www.yelp.com/dataset/download. All the files were clean **JSON files with no need for preprocessing**. We used business and reviews data ONLY as the both of them had User_ID in order to tie to a user.

What are its characteristics?
The Yelp dataset contains 150,000 businesses out of which **- 1,200 are restaurants with more than 500 reviews**. We limited the training of our prediction and sentiment analysis model to restaurants with more than 500 reviews due to running memory restrictions on a **APPLE Silicon M1 Macbook Pro with 16GB RAM**

Also, fragmented the entire training dataset by cities to work around the memory restrictions.

Business Data

Business Id Name Address
City State Postal Code
Latitude Longitude Is open?
Hours
Attributes --> a few filtering attributes such as accept credit cards?
Categories - has a "Restaurants" category. But alsoe type of food such as Sandwiches, Burgers, Italian, Pizza, Steakhouses, and keywords such as good service, etc. Used for initial data filtering and to filter results based on 'Category' input on the front-end
Business Review Data -->
Stars
Review Count

Reviews Data

User Id
Business Id
Review Id
Review Text

Review Rating Data -->
Stars
of 'useful' votes
of 'funny' votes
of 'cool' votes
of elite' votes

Experiments and Results

How did you evaluate your approaches?

Scalability
1: The first experiment was performed to determine if there was an upper bound on the volume of data used to train our model using KNNBaseline algorithm.

Accuracy
2: The second experiment was performed to select a collaborative filtering algorithm. We trained and tested a POC model using different algorithms and 3-fold cross validation to calculate RMSE values.

Hyper-parameter Tuning
3: The POC model was tuned for hyper-parameters using Grid-SearchCV

What are the results?
1: Training/Testing Times for various size of reviews dataset is as follows.
Based on the results, our **model requires 4 seconds of additional training time for every 10,000 reviews** added to the training set.

The Challenge - Training algorithm is memory bound – larger training sets will require more RAM to run.

Solution - We scaled our model training by splitting up the dataset by cities.
Also, it allows to train on more reviews but does not user similarity across cities.

We decided to go with 60,000 reviews for the following experiments.

Table 1: Training/Testing Times (business dataset fixed at 2800 businesses)

Size of review dataset	Training time (seconds)	Testing time (seconds)	Memory (RAM) errors
60,000	6.954	0.248	None
70,000	9.695	0.353	None
80,000	13.276	0.448	None
90,000	17.174	0.578	None
100,000	21,971	0.705	None
110,000	27.552	0.843	None
120,000	-	-	Exceeded amount of RAM on the machine – error returned

2: Accuracy Metrics of different Collaborative Filtering Algorithms for the POC model is as follows.
We concluded that KNNBaseline is the best performing algorithm and was used for building the model for the application.

Table 2: Accuracy Metrics of different Collaborative Filtering Algorithms

Algorithm	RMSE	Train time (seconds)	Test time (seconds)
KNNBaseline	1.180	13.227	0.259
KNNBasic	1.228	12.933	0.251
KNNWithZScore	1.260	13.484	0.260
KNNWithMeans	1.261	13.158	0.252
SlopeOne	1.261	0.123	0.052
NormalPredictor	1.591	0.021	0.071

3: RMSE, Training time and test time results for Hyper-parameters tuning are as follows.
Hyper-parameter tuning has slightly improved the RMSE.

Table 3: Optimized KNNBaseline vs Un-optimized Using the Same Dataset

Algorithm	RMSE	Train time (seconds)	Test time (seconds)
Optimized KNNBaseline	1.158	19.086	0.824
Unoptimized KNNBaseline	1.173	14.215	0.930

How do your method compare to other methods?

Model - As mentioned in the Approaches section, we chose a user in test dataset and inspected the recommendations for the user. We generated 2 new recommendations which user has not visited and were generated based off user's similarity with other users and overall sentiment for those restaurants.

Future Research -
Experimenting with using highly scalable cloud services to optimize model training using full dataset.
Experiment to determine threshold value better than 500 reviews for business filtering that optimizes model performance.
Experiment to optimize the weighting between Scikit-Surprise predicted rating and sentiment analysis polarity score.

Web Application -
Our Web Application is customer focused thus we ran a usability survey within family and friends. Based on the usability analysis, we received medium to lower range score under user-friendliness category. We concluded that our web application needs work to make it more easy to use and broadly scoped for other cities.