MODEL USER PREFERENCES FOR LOCATION BASED RECOMMENDATION

CS5100 – Final Project

http://rohitbegani.github.io/FAIproject-CS5100
https://github.com/rohitbegani/FAIproject-CS5100

Bhanu Pratap Jain
Rohit Begani
Ronny George Mathew

Contents

1 Abstract

In this project we will create an artificial agent that can recommend eating choices to a user. Our agent will run through an existing dataset, representing user's eating behavior (cuisines, category, etc.) and learn the user's preference model for food choices. Based on the learned user preferences, current location and time, the agent will recommend an eating choice for the user in terms of food type and the optimal place to eat. Additionally, through user's feedback, the agent will evolve its preference model and recommendation function.

2 Methodology

We have used concepts of supervised learning to come up with a prediction algorithm which is derived from KNN.

The KNN algorithm or the k-Nearest Neighbors Algorithm is a non-parametric lazy learning algorithm. [1]. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. [2] The model for KNN is the entire training dataset. When a prediction is required for an unseen data instance, the KNN algorithm will search through the training dataset for the k-most similar instances. The prediction attribute of the most similar instances is summarized and returned as the prediction for the unseen instance. The similarity measure is dependent on the type of data. For real-valued data, the Euclidean distance can be used. Other types of data such as categorical or binary data, Hamming distance can be used. [3]

For our custom algorithm we have leveraged from the nearest neighbor measure from the KNN to formulate the prediction score for each sample. Further is an explanation of our data modelling and algorithm:

2.1 Data Modelling

We have used the yelp dataset in our project. To work with the huge dataset, we uploaded the entire dataset into mongodb. From this dataset, we found out a user who has the maximum number of reviews and created a json file from the reviews of this user and the corresponding business.

The json file we are using in this project includes the reviews of one user and the details about the reviewed business. This generated json is inbound to our system which has the following data models:

a. User Model
b. Business Model

2.1.1 User Model

The user model represents the user interacting with the system. We have used passive learning over an existing data set to train the user model. The user model has the following attributes:

- User Id: Id of the user.
- Name : Name of the user

- Location Latitude (Filter[1]) : Current Latitude of the user
- Location Latitude (Filter) : Current Longitude of the user
- Music (KNN Class): Specifying the type of music user like. It is also used as a class in KNN
- Ambience (KNN Class): Type of ambience user prefers in a restaurant. It is also used as a class in KNN
- Good for (KNN Class): Good for features in a restaurant that user prefers. It is also used as a class in KNN
- Category (KNN Class): Type of food user prefers in a restaurant. It is also used as a class in KNN
- Dietary Restrictions (KNN Class): Any dietary restriction that user considers while going to a restaurant. It is also used as a class in KNN
- Alcohol (KNN Class): Alcohol preference of the user, in a restaurant. It is also used as a class in KNN
- Attire (KNN Class): Attire preference of the user, in a restaurant. It is also used as a class in KNN
- Price range (KNN Class): Price Range for the user for a restaurant. It is also used as a class in KNN
- Parking (KNN Class): Parking preference of the user in a restaurant. It is also used as a class in KNN
- Miscellaneous (KNN Class): Other attributes that user sees while going to a restaurant. These might be 'delivery', 'smoking' etc. It is also used as a class in KNN
- Wi-Fi (KNN Class): Wi-Fi preference of the user in a restaurant. It is also used as a class in KNN
- Stars: Average rating for a particular restaurant. Used in learning user model.
- User Rating: User rating for a particular restaurant. Used in learning user model.

More details on the User Model can be found in the Data Mapping Sheet.

2.1.2 Business Model

The restaurants in the system are represented by the Business Model. All the KNN classes mentioned in the user model are exactly replicated in the business model. The attributes in the business model are:

- Business Id: Id of the Business.
- Name : Name of the Business
- Location Latitude (Filter) : Current Latitude of the Business
- Location Latitude (Filter) : Current Longitude of the Business
- Open Now: Tells if the business is open
- Hours: Working hours during the week
- Music (KNN Class): Specifying the type of music in the Business. It is also used as a class in KNN
- Ambience (KNN Class): Type of ambience user prefers in a Business. It is also used as a class in KNN
- Good for (KNN Class): Good for features in a Business. It is also used as a class in KNN
- Category (KNN Class): Type of food in a Business. It is also used as a class in KNN

[1] Filter on data set

- Dietary Restrictions (KNN Class): Any dietary restrictions a Business can cater. It is also used as a class in KNN
- Alcohol (KNN Class): Alcohol type availability in a Business. It is also used as a class in KNN
- Attire (KNN Class): Attire preference for a Business. It is also used as a class in KNN
- Price range (KNN Class): Price Range for the Business. It is also used as a class in KNN
- Parking (KNN Class): Type of parking available in the Business. It is also used as a class in KNN
- Miscellaneous (KNN Class): Other attributes the Business has. These might be 'delivery', 'smoking' etc. It is also used as a class in KNN
- Wi-Fi (KNN Class): Wi-Fi type availability in a Business. It is also used as a class in KNN
- Stars: Average rating for a particular Business. Used in learning user model.
- User Rating: User rating for a particular Business. Used in learning user model.
- Prediction Score: The score predicted for a particular Business w.r.t to user model.

More details on the Business Model can be found in the Data Mapping Sheet.

2.1.3 KNN Classes

The KNN classes mentioned in the Data Models have a specific format.

- User Model KNN Class:
  It is a list of tuple (feature, weight). Weight is the cumulative weight of all the businesses that are used to train the user.
- Business Model KNN Class:
  It is either a number/string or a list of number/string representing the feature.

2.1.4 Data Sets

The incoming system specific json files are loaded into the system as Business Models, which are further separated into:

- **Training Data**: The training data are the Business Model Data Set which we train the User model from.
- **Test Data**: This is Business Model Data Set, which is used for prediction in KNN.

2.1.5 Data Filtering

Data filters are applied to test data. We have two specific data filters in our system:

a. Location Filter:
   Filters the test data on Business Location with user's longitude and latitude.
   We have used *Haversian formula* to compute this.

b. Time Filter:
   Filters the test data on Business operating hours with user's current time.
   We have taken *system time* in our tests.

2.1.6 User Learning

We train the user model over the training data set of the business models. For this training we use a learning factor that defines the user's bias over training sample. This factor is used to compute a cumulative weight for individual knn class feature for user model. The weight function is defined as:

$$w_i \; = w_i + (lf * r_u) + ((1 - lf) * r_b)$$

$$w_i - Weight\ of\ the\ Class\ Feature\ in\ User\ Model;$$
$$lf - Learning\ Factor\ ; 0 < lf \leq 1$$
$$r_u \; - User\ Rating\ for\ a\ Business$$
$$r_b - Genral\ Busniess\ Rating$$

For Example, let us consider the KNN Class Wi-Fi. Possible features in Wi-Fi can be 'free', 'paid' and 'no'.

For Business class Wi-Fi can be represented as ,

B1 → wifi (free)
B2 → wifi(paid).

For User without training,

U1→ wifi[(free,0),(paid,0),(no,0)].

After training User U1 over training samples B1 and B2,

U1 → wifi[(free,w_1),(paid,w_2),(no,0)]

With the linear weight vector mentioned above w_1 and w_2 can be obtained as:

$$w_1 = 0 + (.75 * 4) + (.25 * 4)\ [Assume\ lf=.75;\ r_{u,}\ r_{b,} = 4]$$

$$w_2 = 0 + (.75 * 4) + (.25 * 4)\ [Assume\ lf=.75;\ r_{u,}\ r_{b,} = 4]$$

Additionally after training the user over all the training data we normalize the feature weights to 1.

2.1.7 Data Modelling Workflow

The flowing workflow is used in the user and business modelling.

STEP 1. Import raw json files into Mongo DB
STEP 2. Convert raw data into system specific json format as inbound to the algorithm
STEP 3. Load Business Models from system specific json
STEP 4. Divide Business Models in to Test and Training Data
STEP 5. Filter Test Data on Location and Time
STEP 6. Train User Model on Training Data

2.2 Prediction

For predictions we have used KNN to find the nearest neighbors. As the KNN classes for both the user and the business model are the same, we can find the nearest neighbors (Businesses) for our trained user. Instead of taking the distance between two samples we are measuring the prediction score between the user and the business based on KNN classes.

2.2.1 Prediction Workflow

STEP 7. Iterate over the Test Data
STEP 8. Match User Class feature-values to Business Class (Test Data) feature-value.
STEP 9. If there is a match, accumulate the class weight from the user as prediction score.
STEP 10. Take 'k' Business Models with the highest prediction score as an output.

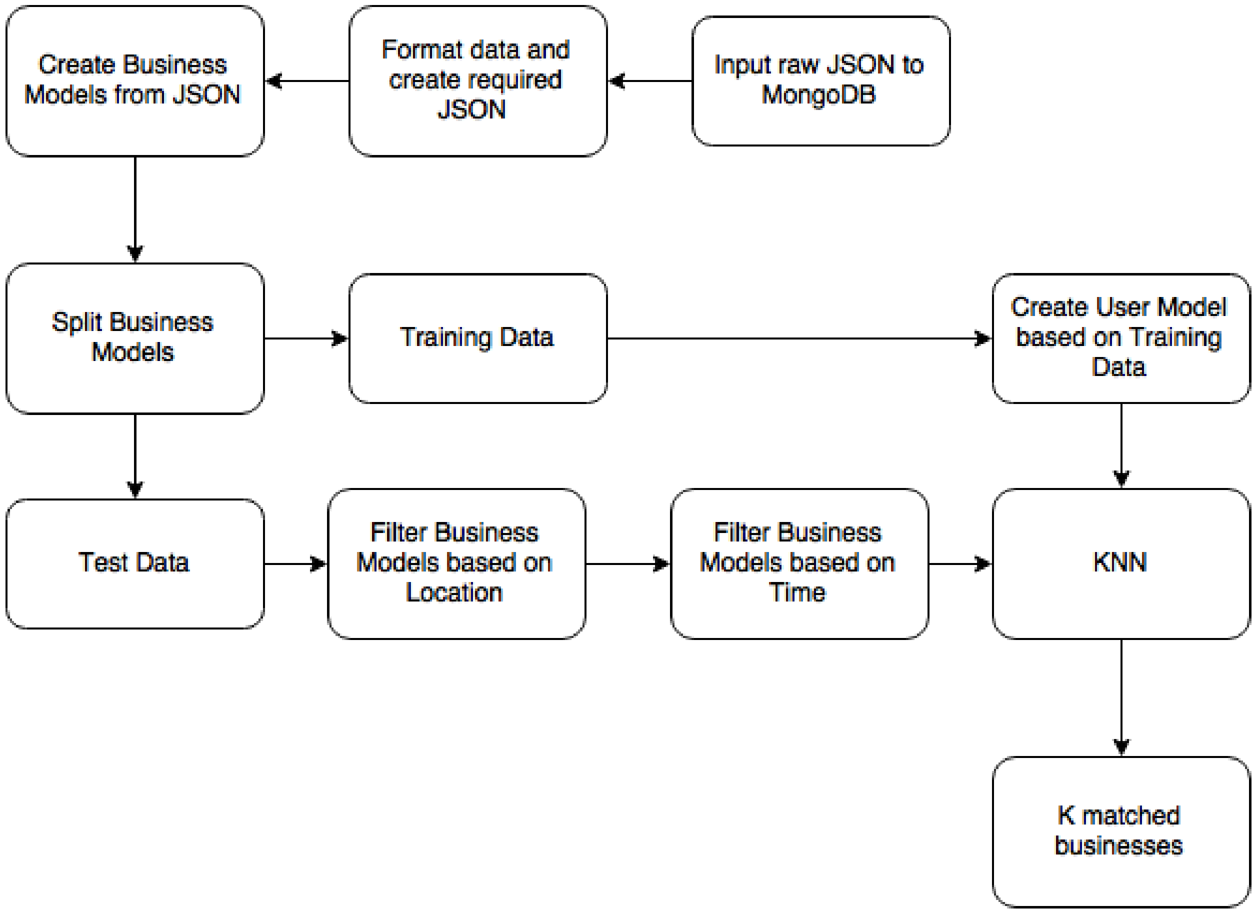

Create Business Models from JSON
Format data and create required JSON
Input raw JSON to MongoDB
Split Business Models
Training Data
Create User Model based on Training Data
Test Data
Filter Business Models based on Location
Filter Business Models based on Time
KNN
K matched businesses

Figure 1: Integrated System Workflow

2.3 Mean Absolute Error

We are using Mean Absolute Error (MAE) measure to compute the accuracy of our prediction. To compute the MAE we consider the true rating for the business given by the user and the predicted rating for the business by our model. To map the predicted score from KNN to a predicted rating, we consider the following scale:

Predicted Score Range	**Predicted Rating**
0	0
0 to 1	0.5
1 to 2	1
2 to 3	1.5
3 to 4	2
4 to 5	2.5
5 to 6	3
6 to 7	3.5
7 to 8	4
8 to 9	4.5
9 to 10	5

Using this predicted rating, we will compute the MAE using the below formula,

$$MAE = \frac{1}{n}\sum |\, r_p - r_u \,|$$

$n \text{ is the total number of predictions}$

$r_p \text{ is the predicted rating for the particular prediction}$

$r_u \text{ is the user rating for the particular prediction}$

3 Results

We ran the algorithm in two scenarios as following.

3.1 Without Data Filters

Here we are not using any location or time filter on the data set,

With *learning factor* as *.75, distance filter False, k as 5, time filter as False.*

- Input :

Item	Record Count
Raw Data	661
Processed Business Models	661
Test Data Set	159
Training Data Set	441

- Output : ***MEA = 0.7 (Computed only on the predictions)***

Business Name	User Rating	General Business Rating	Prediction Score	Prediction Rating	Prediction Rank
grimaldi's pizzeria	4	4	6.13016666667	3.5	1
carlsbad tavern	4	4	5.755	3.5	2
chicago brewing company	3.5	5	5.64075	3.5	3
ajo al's	3	4	5.63683333333	3.5	4
blue adobe grille	4	4	5.6155	3.5	5

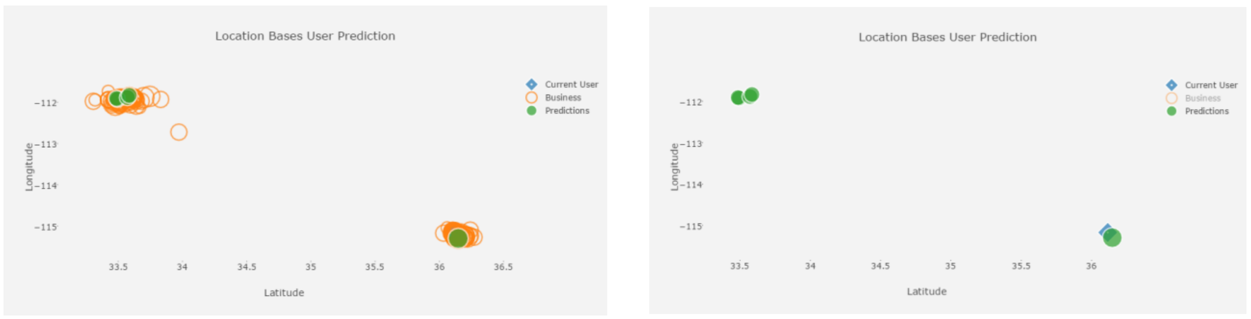


Figure 2: Prediction Visualization: Without Data Filters

3.2 With Data Filters

Here we are using both location and time filter on the data set,

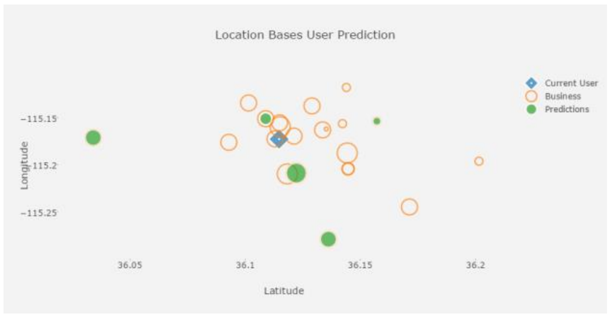
With *learning factor* as *.75, distance filter as 10, k as 5, time filter to system time.*

- Input :

Item	Record Count
Raw Data	661
Processed Business Models	661
Test Data Set	24
Training Data Set	441

- Output : ***MEA = 1 (Computed only on the predictions)***

Business Name	User Rating	General Business Rating	Prediction Score	Prediction Rating	Prediction Rank
carmine's	4	4	5.58466666667	3.5	1
bonito michoacan	4	5	5.57925	3.5	2
capriotti's sandwich shop	4	4	5.49033333333	3	3
geebee's bar & grill	4	4	5.36133333333	3	4
casa don juan	3.5	2	5.306	3	5



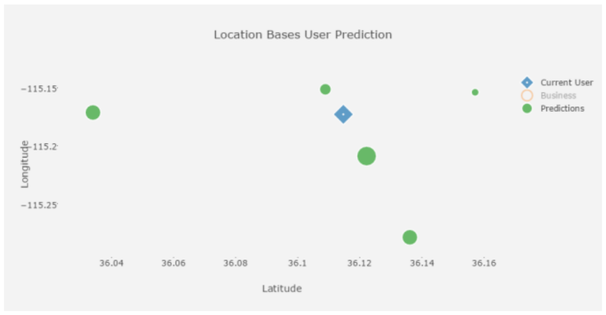


Figure 3: Prediction Visualization: With Data Filters

3.3 Performance

Our algorithm gives an MAE (Mean Absolute Error) of:

- 0.7, without any restrictions on the data set.
- 1, with time and location restrictions on the data set.

The fact that we already have a restricted the data set gives us a MAE of 0.7 but restricting it further changes our MAE value by a factor of 1. With a deeper dataset our algorithm would give an even better MAE value.

4 Future Scope

This application can be extended by adding Natural Language Processing. In future, we aim to apply Natural Language Processing to a user's review and try to understand that if they do or don't like a business then what is the real reasoning behind that? Is it because of some specific feature like a lack of parking? If it is, should we stop recommending businesses which doesn't have parking space? Is it because the business doesn't have free Wi-fi? All these factors we believe would help us in improving our recommendations and giving even better results.

5 Project Public Repository

https://github.com/rohitbegani/FAIproject-CS5100

6 Reference

[1] https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/
[2] http://www.saedsayad.com/k_nearest_neighbors.htm
[3] http://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
[4] http://www.ijera.com/papers/Vol3_issue5/DI35605610.pdf