Content Based Recommendations

Content based model is best suitable when there is only limited amount of user data. As a user logs in to the application, we capture all the attributes that comprise this limited user data. We will be calculating Dice similarity coefficient between the user vector and the restaurant vectors. Given, previous ratings of restaurants by a particular user, we will predict the closest matches in similarity of restaurant attributes. In case, previous history is inaccessible (email login), we will rely on the onboarding data and yelp challenge dataset alone for user data.

Restaurant Attribute Mapping

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Restaurant/Attribute | Attribute 1 | Attribute 2 | Attribute 3 | Attribute 4 | Attribute 5 |
| Restaurant A | x | x | x | x |  |
| Restaurant B |  | x |  |  |  |
| Restaurant C | x | x | x |  | x |

Dice Coefficient

|  |  |  |  |
| --- | --- | --- | --- |
| Restaurant/Restaurant | Restaurant A | Restaurant B | Restaurant C |
| Restaurant A |  | 0.4 | 0.75 |
| Restaurant B |  |  | 0.4 |
| Restaurant C |  |  |  |

Considering a user liked A, C has highest similarity with A, and hence the recommendation.

Calculate the similarities between

* Restaurants and Cuisines
* Restaurants and Attributes
* Restaurants and Users (with yelp dataset considering we could identify the primary)
* Restaurants and Reviews

Following this, we need to use a hybrid model that is tweaked to produce the best results in combination of any or all of the above.

User to user recommendations

Users vs Restaurant Ratings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User/Rating | Restaurant A | Restaurant B | Restaurant C | Restaurant D | Restaurant E |
| User 1 |  | 2 | 1 |  | 1 |
| User 2 | 5 |  |  |  |  |
| User 3 | 5 |  | 1 | 2 |  |
| User 4 | 4 |  | 1 | 2 |  |
| User 5 | 3 |  | 5 | 1 |  |

Users that liked the similar restaurants are put into a cluster. We need to calculate the similarity (Pearson Correlation) between the users by similarity between their restaurant ratings.

Within each cluster, the algorithm needs to select the users that have already evaluated each of the unrated restaurant. Finally, predict ratings for each unrated restaurant for a given user.

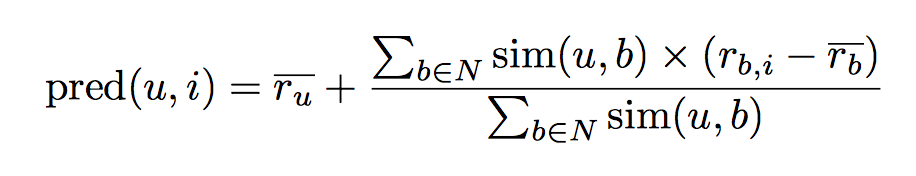
Target user mean rating = u

Unrated restaurant = i

Neighbour’s rating on the target restaurant = r b,i

Neighbour’s mean rating = r b

Neighbour and Target User similarity = sim(u, b)



Item to item recommendations

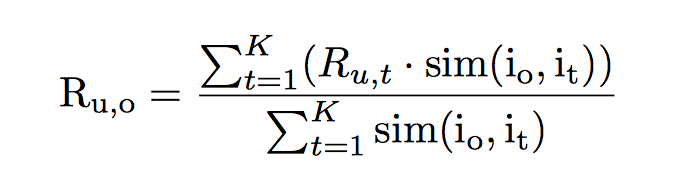
Predict the rating of the restaurant based on the other restaurants the user has rated before. Cosine similarity between the restaurants is calculated based on all the users that co-rated both restaurants.

Users have different rating profiles. Some users rate usually higher and others lower. This can be fixed with adjusted cosine similarity.

Adjusted Cosine Similarity Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Restaurant/Restaurant | Restaurant A | Restaurant C | Restaurant D |
| Restaurant A |  | - 0.86 | 0.86 |
| Restaurant C |  |  | - 0.5 |
| Restaurant D |  |  |  |

Predict the rating of a restaurant for a target user by using target user ratings on the similar items and applying the weighted sum.



K most similar restaurants to a given restaurant will be the recommendations.

Constraints/Association Rules

Define rules for each user cluster. Work out a restaurant vs attributes/cuisines/reviews/dishes matrix. Identify the restaurant variables that are apt for each cluster and attribute weights to the variables accordingly. Given a new user, based on the cluster he belongs to, suggest the restaurants that meet the set association rules.

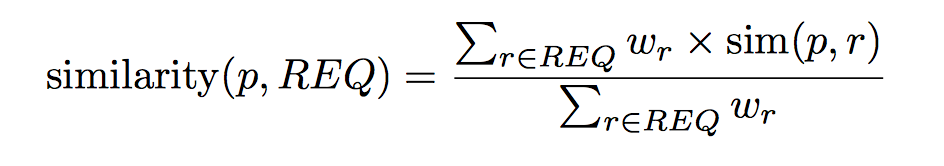
Case based

Calculate similarity based on the distance between user requirement vector and restaurant attribute vectors.

Attribute = p

User requirement = r

Weight of a specific requirement = wr



Recommend restaurants that are closest to the user requirements in terms of their attributes.

Evaluation Metrics

Topline metric is conversion, where conversion = # of restaurants visited/recommended

Ideally, the conversion metric needs to be measured against each individual user, which is likely biased as it depends on the understanding and the accuracy of the target audience. At the perfect fit between target audience and the model apt for them, the accuracy is 1. In reality, both the definition of the target audience and the model is interdependent and evolving. Hence, the measurement of any given model is only as good as the definition of target audience.

Given the scenario, there is no perfect model to be obtained until there is a saturation of the target audience. Target audience can never saturate as there will always tend to be pockets/clusters of people bound to a given definition of target audience. Forming clusters will bring the audience that is closer in definition together but never on perfect alignment. The only way to capture the nuances of every single user is by having a neural network dedicated to each user that will act as a replica of his knowledge/memory and behavior.

Considering the collaborative filtering algorithms still base their recommendations on clusters, the current evaluation is limited to the conversion of users within a given cluster, with each cluster having its own accuracy. An average accuracy for all clusters is biased as the recommendations are highly dependent on the set of users within a cluster.

1. A/B tests are the best way to evaluate different models.
2. Precision = # of restaurants relevant & recommended / # of items recommended [Measure of exactness]
3. Recall = # of restaurants relevant & recommended / # of relevant items [Measure of completeness]
4. For Top k recommendations, measure precision against recall. [Measure of relevance]

Others

* Mean Absolute Error
* Accuracy
* ROC Curve