Classification-Based Recommendation System for Recreational and Otherwise Visit-Worthy Venues

Prashant Gupta, Richard Hopper, Divya Jain, Nikhil Nayak, Sandhya Nayak

Abstract

We have developed a recommendation system that helps users discover new places of interest in different categories based on similar users' habits. We have utilized a dataset of Foursquare check-ins, which contains check-ins from New York City over ten months, to train this system. In generating recommendations, we first use k-means to cluster the users into six broad clusters based on their frequency of visitation across Foursquare venue categories. Then we use a decision tree-based classification model to find the appropriate cluster for our test user. Then we use a cosine distance metric to find three users most similar to our test user and compile a list of venues our test user hasn't visited yet. We give the test user a choice to select a particular category like food places, outdoor & sports activities, etc. for their recommendations. Based on the category selected by the test user, we recommend three venues from our venue list for the user to check out.

Introduction

A major problem that groups of friends or individuals in a new city encounter is not knowing how to utilize the venues around them. Most major cities offer a plethora of different dining establishments, recreational opportunities, and tourist attractions, but it's difficult to decide which of these opportunities are most appealing to any unique individual. Decision-making based on aggregate opinion (general reviews) aid in this process, but these results are often not enough to make an informed decision catered to any specific predilections an individual may have.

Our goal was to build a solution to this problem by building a robust recommendation system. As a general class of programs, recommendation systems are a subclass of information filtering systems that seek to predict the 'rating' or 'preference' that user would give to an item.[1] Recommendation systems have become an integral part of social computing, as it is one of the best ways to connect people with resources or products they are likely to enjoy. Because of this efficiency, these systems also have a large use case in targeted advertising. By aggregating and analyzing the potential user ratings of many products, these systems can choose the products that best fit the users in question and make the user aware of their presence.

We have gained access to a large database of Foursquare check-ins in New York City[2], and designed a system that uses this data to recommend establishments to visit based on previous check-ins and trends gathered from other users' habits. We have preprocessed this data to aggregate user's visits to each category of venue, then lumped these categories together into 9 high-level groups to associate users based on very general venue preferences (e.g. food and dining establishments, recreational facilities, etc).

To build a more efficient recommendation system, we have built in a clustering component that groups users based on similar visitation habits. When recommending new venues, we identify similar users based on these clusters and use their visitation habits to identify possible recommendations. This strategy makes our recommendation system much more efficient, as we do not have to analyze our entire pool of users with every query but can instead work with a much smaller dataset. In addition, this means that our recommendation system can identify large groups of users that would be ideal for a targeted advertising campaign, rather than inefficiently singling out and aggregating individual preferences.

System Design

We developed our system in the R statistical programming language, taking advantage of several of the analytic libraries provided in the CRAN repository.[3][4] Our system reads in a comma separated value (CSV) file containing preprocessed Foursquare check-in data, runs algorithms to produce clustering results, and ultimately generates recommendations based on this analysis. We developed our system through a large amount of research and comparison between clustering algorithms and various high-level methods for establishing the best recommendation data. This research will be explained in the following sections.

Data Preprocessing

Our initial Foursquare dataset contained 227,428 check-ins from 1,083 users in the New York City area. Each of these check-in rows contained information about the user's ID, the venue ID, the venue category ID, the venue category name, the location of the venue (latitude and longitude), the timezone offset, and the UTC time of the check-in. Most of this data was not relevant to our system, and we needed a way to reduce the dimensionality of this data for efficient clustering of users.

We first eliminated columns that were not relevant to our clustering criteria, leaving only the user ID and the venue category. In establishing similar visitation patterns between users, we did not require knowledge of which specific venue the user visited, the location of that venue, or the time at which they checked in. Once we were working with a two-column (user ID, venue category name) dataset, we ran a script on this data to generate a list of users paired with each category name, and recorded the amount of times each user had checked into a venue with a given category.

The next level of our preprocessing occurred when we grouped our 251 specific categories (e.g. *Aquarium*, *Subway*, and *Salad Place*) into the 10 higher-level categories provided by the Foursquare API (e.g. *Arts & Entertainment*, *Transportation*, and *Food*, respectively). We ran a Javascript algorithm to parse through the initial data and generate a new CSV file with each user's consolidated visit count to these broader categories. There was an *Event* category provided by Foursquare, but none of the check-ins we had in our dataset fit into this category, so we removed it from our category list and ended up with a 9-dimensional dataset to perform our analysis on.

We then noticed a large amount of noise in our data, so we decided to store relative frequency of visitation instead of total visitation counts. We tested different strategies for consolidating this data, including testing different measures of central tendency to establish a proper threshold for "frequent" visitation and evaluating binary frequency (frequent vs.

infrequent) against varying levels of frequency (1 .. n). We ran this evaluation within each column to ensure a true representation of user's relative visitation habits per category. Our research showed that a binary frequent/infrequent metric, generated using a trimmed mean measure generated the best results. The top 10% and bottom 10% of values were not used to compute the mean value, and hence the mean was not biased by outliers in our dataset.

Clustering

The first question we had to answer when forming user clusters was how many clusters to divide users into. We originally thought the best strategy would be to form the same number of clusters as we had groups, but decided after doing this that we needed to objectively determine the proper number of clusters by optimizing the sum squared error (SSE) in relation to the number of clusters. After modeling this relationship, we discovered an optimal relationship between these variables with 6 clusters, with the help of an elbow plot.

We then attempted to cluster the users using different strategies, including k-means, hierarchical clustering, and partitioning around medoids[5]. We found that **K-Means** was the most effective choice, and decided to form our clusters using that strategy for the final recommendation system. K means provided us with the simplest algorithm suitable to our needs and gave a number of advantages over other algorithms like fast running time, well defined non overlapping globular clusters.

Classification

Once the training data was clustered, that is each of the users was assigned to a cluster, the cluster number was added as an additional attribute to the user data. A decision tree model was built using the cluster number as the class labels for classification. This model will now enable us to assign a class label to a test user which is the same as the class label for users that it is most similar to. This will be helpful in reducing the search space when searching for similar users.

Different strategies such as KNN, decision trees and random forest were tried for classification. It was observed that decision trees and random forests gave much better results as compared to KNN. **Decision trees** were chosen for classification in our system as they are easier to understand. More details on the classification algorithms can be found in the Classification manual.

Recommendations

The last part of our system was to implement the recommendation system itself. This system takes as input two things: the user number and the broad category of venues in which the user wants a recommendation.

First, a list of all the venues in the dataset is compiled and a frequency value is associated with each venue. This frequency value is calculated as the count of visits to that venue (by any user) in the original check in data set.

Using the decision tree created from the training data set, the test user is first classified to identify the user group that the test user belongs to. Three users that are most similar to the test user are then identified from this user group using cosine similarity. The list of venues in the category that the test user requested is then compiled for recommending the user. This list contains the venues visited by the similar users but not the test user. The top three venues from this list, based on the frequencies of visits, are then extracted for recommendation.

It is possible that the list of recommendations has less than 3 venues (if the test user has been to all the venues in that category that the similar users have been to or if none of the similar users have been to any venue in that category). In this case, the search falls back to the original list of venues in that category and the top venues, according to the frequencies of visits that are not in the list of recommendations and have not been visited by the test user are added to the list of recommendations.

Thus at the end of this process, the list of recommendations has 3 venue IDs which represent venues in the category chosen by the user. Three calls to the Foursquare API are now made using each of the 3 venue IDs. The API resolves the venue IDs and returns the venue details which are then displayed to the user.

Evaluation

We evaluated the design of our system by excluding a portion of the original dataset from the training set, and passing in other users as the test set. For efficacy, we decided to keep an even 1070 records as the training set, and set aside the remaining 13 user records for testing since we could not validate a true recommendation because of the nature of the problem.

We also created a simulated data of 18 users to evaluate our clustering technique and the recommender system itself. We chose 6 categories out of the 9 given by the foursquare data and gave high number of check-ins to these 18 users in sets of three randomly in these categories. This was done just to corroborate that our clustering algorithm does cluster similar users based on their check in data. The resulting clusters all had 3 users each with similar venues checkins.

After validating our clustering technique, we created another user to test our recommender system. We found that our system first found the appropriate cluster for our user based on his check ins using the classification model we created, and then suggested a new place based on check ins of the other users in the cluster.

Conclusion and Future Work

In this project, we have introduced a recommendation system that recommends new venues to a user, based on his previous check-in patterns and check-in patterns of similar users. Our preprocessing and clustering of the training data was designed to enhance the speed and reliability of our recommendation system, and our tests on simulated data show that we have produced an effective solution. In the future, the system can be enhanced to make use of the check-in times of the users to recommend places that are currently trending, and also to find new friends who are currently at the same venue and have similar interests.

Another use of the project would be targeted advertising. By identifying groups of individuals having an interest in product and venue categories similar to those being advertised, we can deliver a market segment to advertisers that is prone to wanting the product in question. Through use of this system, advertisers can enjoy a reduced-cost campaign while reaching users with higher potential to convert to paying customers.

Citations

[1] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, pp. 1-35, 2011

[2] Dataset source: https://sites.google.com/site/yangdingqi/home/foursquare-dataset

[3] R Project (n.d.). http://www.r-project.org/about.html. Retrieved 2015-04-11.

[4] CRAN Repository. http://cran.r-project.org/

[5] Partitioning Around Medoids, R Programming Manual. https://stat.ethz.ch/R-manual/R-patched/library/cluster/html/pam.html

[6] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, Zhiyong Yu. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. IEEE Trans. on Systems, Man, and Cybernetics: Systems, (TSMC), 45(1), 129-142, 2015