Restaurant Recommender

A Yelp Open Data Project in R

Data Studies Summer Series | September 9, 2015

Students:
Graham Sarasy
Shubhangi Srivastava
Andrew Tom
Iris Xie
Adam Zhu

Mentors:
Prof. Duncan Temple Lang & Prof. Joe Dumit

Approaching the Dataset

- Previously attempted data problem:
  - Identify “struggling” businesses
- Strategy
  - Text mining of user reviews to find trends
  - Sentiment analysis to gauge the accuracy of star ratings
- Limitations of this data problem:
  - Businesses close for reasons unrelated to star ratings
  - On the periphery of Yelp’s capabilities
- Back to the drawing board
  - Further exploratory analysis
  - Problem clarification
  - Research on previous Yelp dataset projects

Newly Refined Data Problem

- Recommendation System
  - Suggesting new restaurants to Yelp users, based off the star ratings of other Yelp users who have reviewed similar or the same restaurants as the user
  - This allows for a more personalized user experience
  - Focus: consumers near college neighborhoods
  - There are a number of filtering techniques that can be applied to populations

Goals

- Cluster Yelp users based on similar preferences
  - Preferences determined by nearest neighbors (i.e. users that positively review the same businesses)
  - Star rating from 1 to 5
  - Similarity based on review scores alone
- Predict user ratings based on preference clusters
  - Generate index of similar users
  - Connect user preferences to business_id
  - Map the unique business IDs to actual business names

Data Collection

- Yelp Academic dataset variables:
  - Businesses
  - Reviews
  - Users
- Information known about users:
  - User_id
  - Review_id
- Information known about businesses:
  - Business_id
  - College neighborhood
  - Location (City, State, latitude, longitude)
  - Aggregate star ratings for business

Data Cleaning

- Cleaned “categories” using Google Refine:
    - Cleaned Typos
    - Merged and tagged neighborhoods associated with one campus.
        - e.g. - UCBerkeley-UC/Campus Area/Telegraph Avenue/Downtown

13490 rows Extensions: Fre

Show as: **rows** records Show: 5 **10** 25 50 rows Sort « first ‹ previous **1 - 10** ne

city	review_coun	name	neighborhoo	url	longitude	state	stars
Providence	5	roba!dolce	Brown;College Hill	http://www.yelp.com/biz/roba-dolce-providence	-71.4006111	RI	2
Providence	20	Grad Center Bar	Brown;College Hill	http://www.yelp.com/biz/grad-center-bar-providence	-71.4007221	RI	4.5

Sampling the Population

1) Take out a portion of the data to be used as a “holdout” set for cross-validating our results.
   a) For each run of the algorithm, 4 businesses were sampled out.
2) Create a comparison matrix (M).
3) Create a distance matrix:
   a) users (rows) and businesses (columns)
   b) For the purpose of determining the distance between different users that reviewed the same businesses to find “nearest neighbors.”
4) Make predictions based on neighbors
   a) Calculate which users are closest to one another
   b) Smallest distance => closest neighbor => best recommendation

User / Business Matrix

Sparse matrix of dimensions length(business_id), length(user_id)

M	Business ID 1	Business ID 2...
User ID 1	Review Score	Review Score
User ID 2...	Review Score	Review Score

1) Generate a matrix
2) Compare User reviews to Businesses
3) Map overlap between user reviews for similar businesses

Distance Matrix

Calculate distance between reviewers from difference between review scores

D	User_id
User_id	Score 1 minus Score 2

1) Generate a matrix of distance between user reviews for the same business
2) Populate this matrix based on the most prolific user

Refining the Algorithm

- Variables (matrix considers only ratings)
- Cut the population down to only include users > 5 reviews
- Similarity constant (Euclidean distance)
  - “How far away a neighbor can be”
  - Varied depending on the matrix, city, # of neighbors, etc
- # of k-nearest neighbors (20 neighbors)

Results

- Number of businesses recommended varied based on our recommendation criteria (>3 or >4 stars).

```
> recover3
DfrvJL-6H5i1nsyL6H6VGg ubjbE_LZIYS_i6yg8S21-Q hzSyQBWeoX94WOpUtxuWVg aysngmx-Tqb7Lpyx9f6yPw yoem-KW_J1n0LqsdLwLeUQ
                   5.0                    5.0                    4.0                    4.0                    4.0
q2EXggq1IIrw2OYPSfvJeQ uKB1eXRNNBq2w4-wUkusZA _9ZZv5V-uM5BXx3P-HslIw qnRO5swUiQ2dNHuHRbAIzg RThn3_Y6qN8MsqZKwa6WyQ
                   4.0                    4.0                    4.0                    4.0                    4.0
BU6FTbHe38g3Enbi077fVA 4y49CSzDlkZB7nkyVizuzg
                   4.0                    3.5
```

Results

- Once we subset the recommended business IDs, we could match those to business names for an output of businesses.

```
> bids$name[place]
 [1] "Hungry Mother"                 "Bertucci's Italian Restaurant - Kendall Square" "Mr. Bartley's Burger Cottage"
 [4] "Z Square"                      "Grafton Street"                                 "Phoenix Landing"
 [7] "Charlie's Kitchen"             "Russell House Tavern"                           "B-Side Lounge"
[10] "The Blue Room"                 "Qdoba"                                          "Pinocchio's Pizza & Subs"
>
```

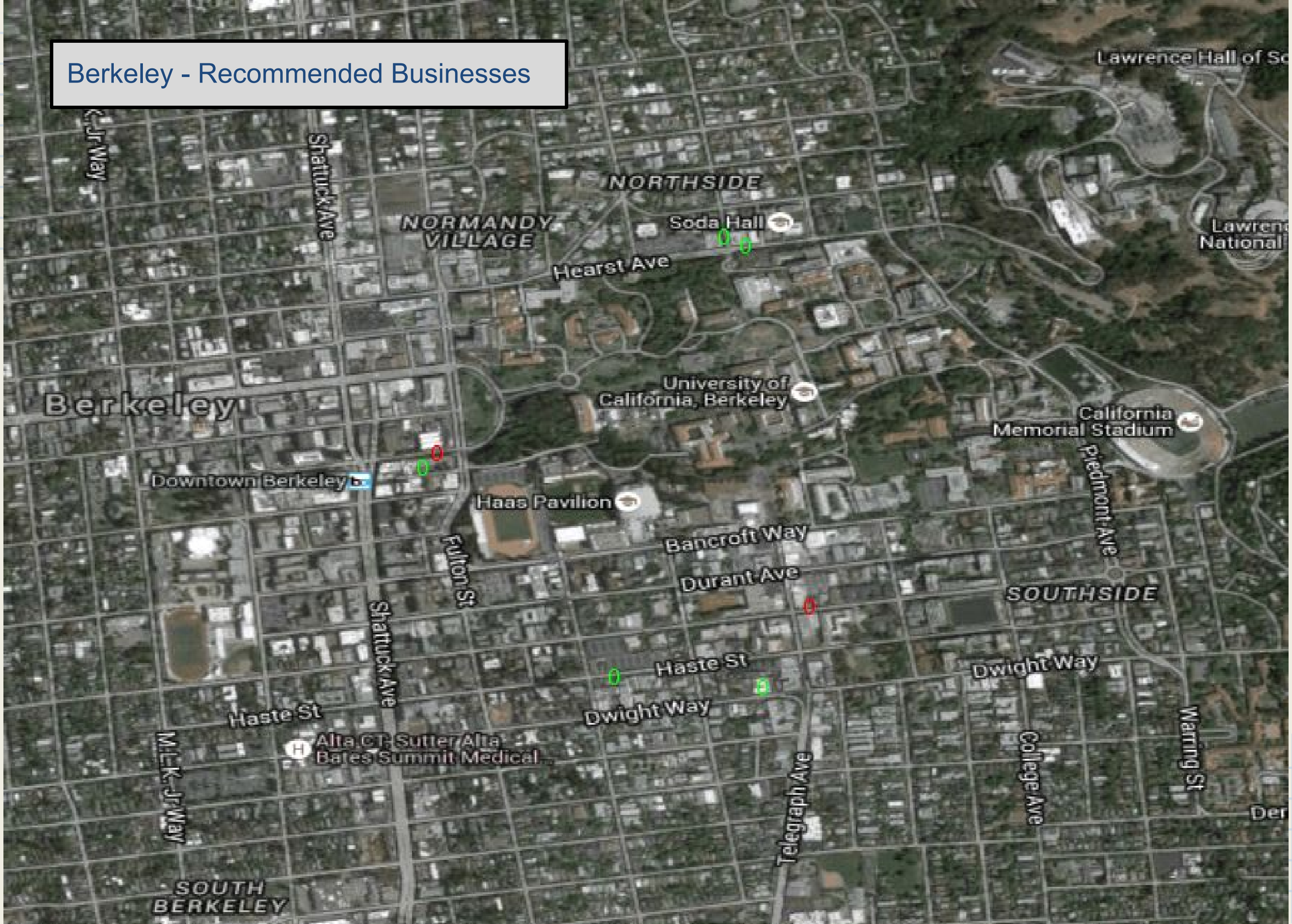
Replicated commendations based on prediction matrix, subset by city

- Berkeley, CA — 20890 Reviews, 185 Unique Businesses
- Los Angeles, CA — 17925 Reviews, 308 Unique Businesses
- Cambridge, MA — 30767 Reviews, 259 Unique Businesses

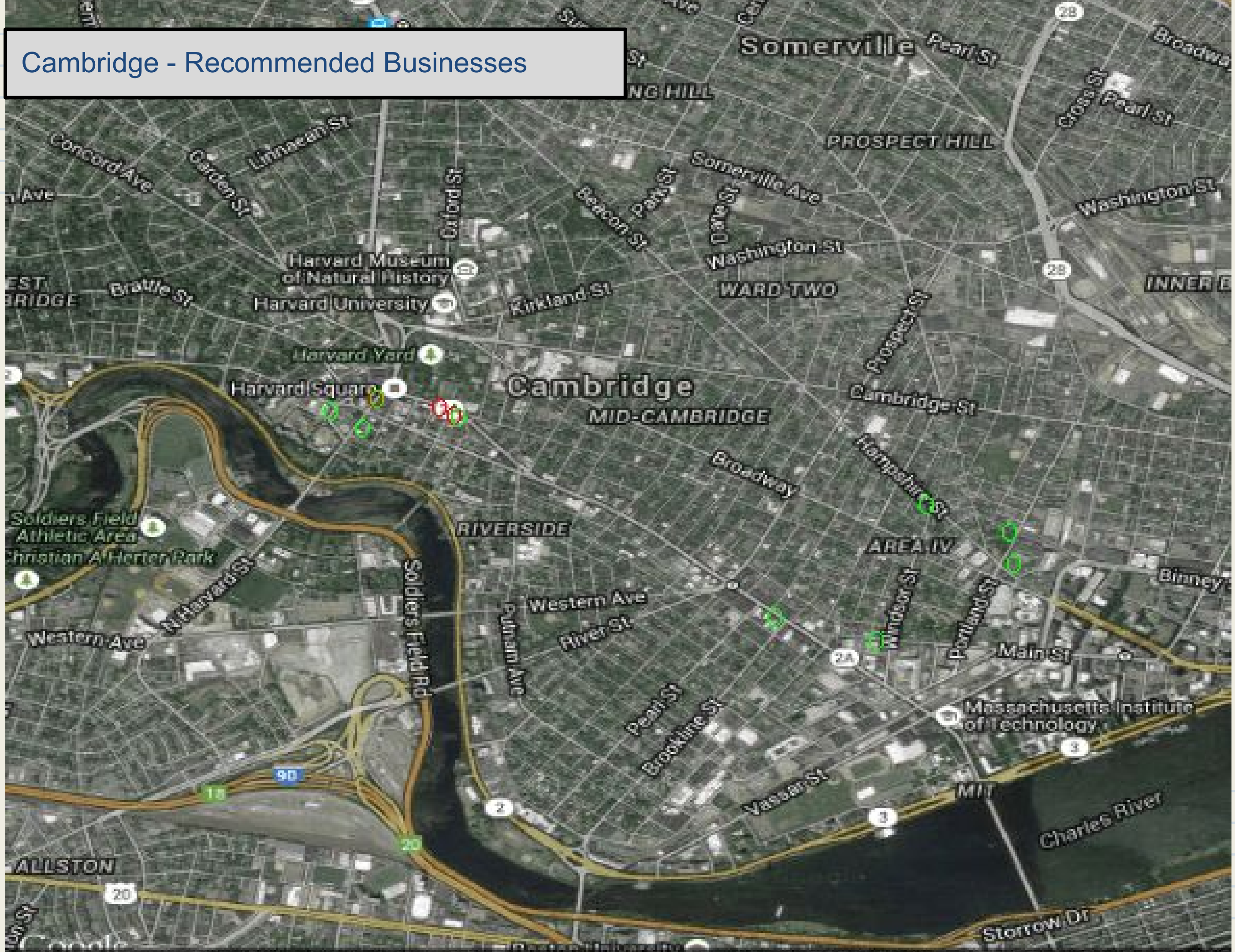
Top Recommended Restaurants

In the slides that follow, we mapped out functional output to maps.

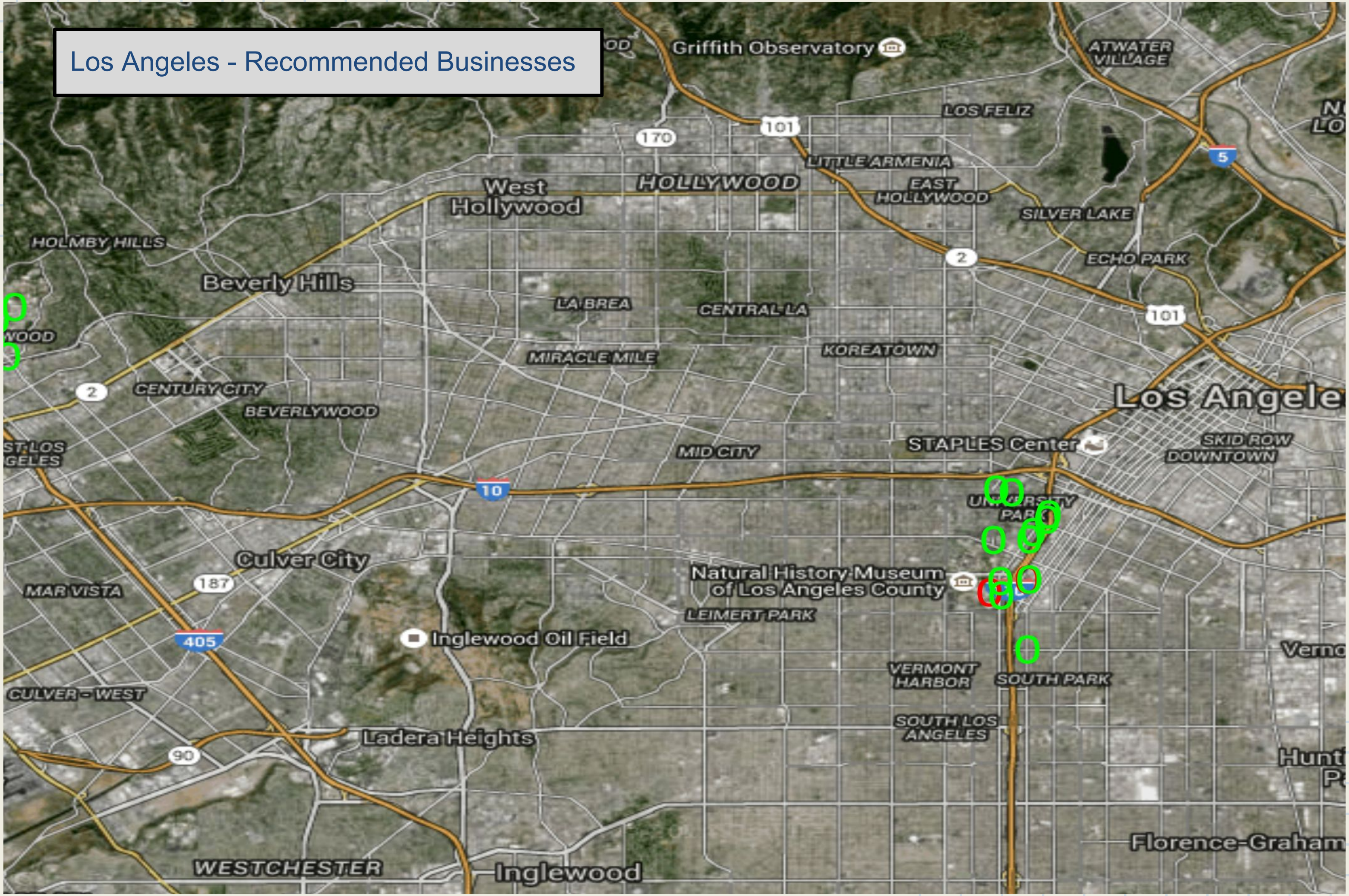
Berkeley - Recommended Businesses



Cambridge - Recommended Businesses



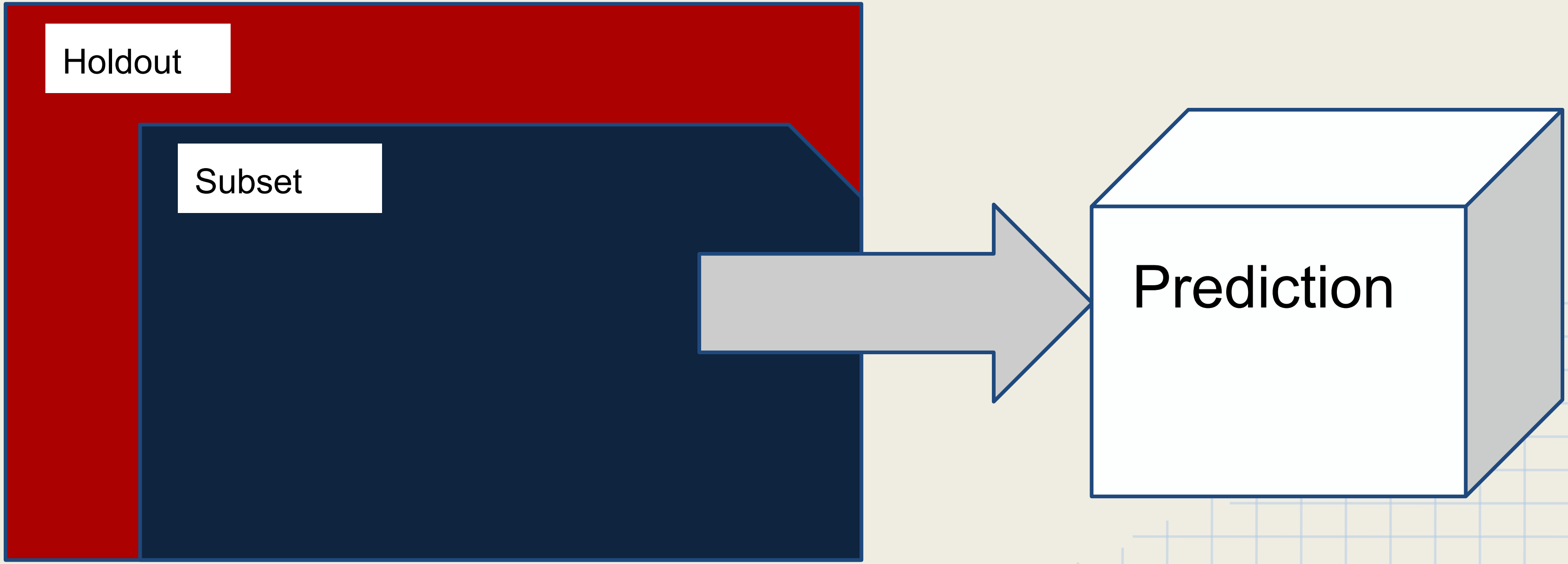
Los Angeles - Recommended Businesses



Prediction / Error Checking

Cross validation between whole dataset and prediction subset

- Error checking (actual - prediction & holdout - prediction)



Error Checking

- Cross validation between whole dataset and prediction subset
- Errors within 0.5 stars would result in stronger recommendations

```
> neigh = sort(D[id,][D[id, ] >= 0])[1:20]
> bus = "qHmamQPCAKkia9X0uryA8g"
>
> # Get the user and its neighbor, and then discard all the columns
> # when that business is not reviewed by  any of these.
> tmp = M[names(neigh),holdout]
> #tmp = tmp[ , apply(tmp, 2, function(x) !all(is.na(x)))]
>
> preds = colMeans(tmp, na.rm = TRUE)#apply(tmp, 2, mean, na.rm = TRUE)
>
> actual = M[id, holdout]
> #actual = actual[!is.na(actual)]
>
> abs(actual - preds[names(actual)])
9cFY3JC7Wx7juBqx7P7Y9w jzmByiypWzevH0GkWv73YQ Yk3KzKmdNRxSdhsc2Hsmpg ICoLhQP3bryEmiYeUNu51g
                    NA                    0.0                    0.5                     NA
```

Limitations

- Used reviews from Oct. 2004 - Oct. 2012
  - Populated matrix using 1 users' reviews - their old reviews are less descriptive of the business at present.
- We are assuming that similar reviews are comparable
  - i.e. Is user A's 3-star rating the same as user B's 3-star rating?
- Recommendations
  - The code only recommends for 1 person at a time
  - Not all average ratings (for specific businesses) are based off the same number of reviews
    - Sparse matrix - some recommendations better than others
  - In an ideal world, we would have a perfectly populated matrix
- Error Checking
  - Again, because of the sparsity of the matrix, some predictions were stronger for some randomly sampled businesses than for others

Conclusions

- The recommender system is useful to Yelp users seeking a refined user experience that shows businesses closer to their preferences.

- Taking this further:

  - We aim to apply this algorithm to multiple users at a time, instead of just one user.

  - Ideally, acquiring updated data would help make the recommendation system more accurate as we would have a better idea of the businesses that are still open/have opened since 2012.

Thank You

Please email any questions you may have.

Andrew Tom - Collaborator & Analyst

andrewtom.careers@gmail.com