Finding the right business, for you!

Sid Reddy

November 8, 2015

Introduction

This paper analyzes the Yelp academic dataset to present the best recommendations to a given user, for a given category and city. Users choose businesses based on their preferences, and standard approaches like choosing businesses based on star ratings do not take such preferences into consideration. This paper details algorithms that can "personalize" the recommendations to a given user. We will evaluate and quantify the benefits offered by such recommender systems.

Methods

In this section, we will load the data, and present some initial analysis. We use the stream_in function from the jsonlite package to speed up the loading of large datasets like reviews.

```r
library(jsonlite, quietly = TRUE)
dataPath <- "yelp_dataset_challenge_academic_dataset"
# Read business file
dfb <- stream_in(file(paste(dataPath, "yelp_academic_dataset_business.json",
    sep = "/")), verbose = FALSE)
# Read users file
dfu <- stream_in(file(paste(dataPath, "yelp_academic_dataset_user.json", sep = "/")),
    verbose = FALSE)
# Read review file
dfr <- stream_in(file(paste(dataPath, "yelp_academic_dataset_review.json", sep = "/")),
    verbose = FALSE)
```

We will make a simplifying assumption that people will search only at the city level for categories (the "right" approach would be to use a zipcode, along with a radius for our search; however, this does not impact our algorithms significantly). Examining the data (we have omitted the code for space reasons), we note that the unique (category, city) combinations are only 8% of the total available combinations, suggesting that multiple choices for a (category, city) exist, and that it is possible to order them for a given user.

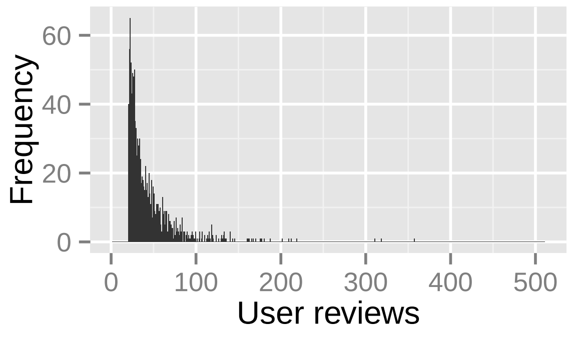
We will also make additional simplifying assumptions, primarily to limit the amount of data for processing (the algorithms described below are pretty expensive to run in R). We will restrict our analysis as follows:

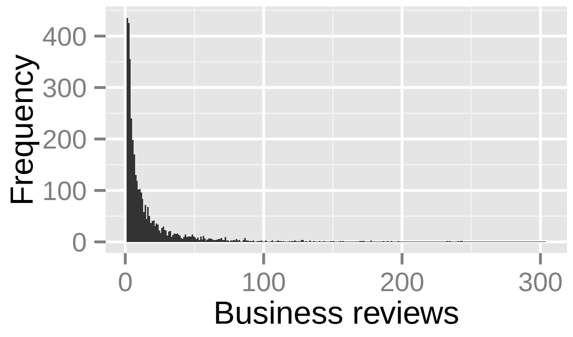
- Limit our analysis to users and businesses that appear in reviews (it's hard to generate good recommendations for users who do not appear in reviews, other than recommending the popular businesses. This is the standard cold start problem that our algorithms below suffer from).
- Limit to users who have written a minimum number of reviews, for the same reason as above. When a user has has written only a small number of reviews, the algorithms do not have enough information to make good recommendations. This minimum number is also used to evaluate the predictions of the algorithms (see the min_known_ratings argument to the recommender_results function below).
- Limit the analysis to the "Restaurants" category (see the restrict_categories function in the code snippet below for more details), to keep the data size small.
- Limit to the first 10 cities (see the restrict_cities function below), again to keep the data size small.

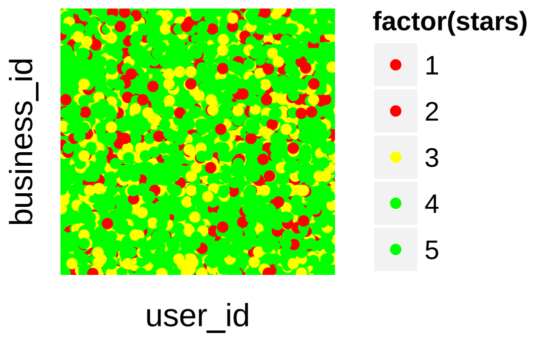
```r
# Restrict analysis to a small dataset (R and libraries have issues handling
# large number of rows; for example, dcast in reshape2 barfs even for 100K
# rows)

# Return businesses that fall in the first num_cities
restrict_cities <- function(num_cities) {
    # Find businesses that got reviews, and look up their records
    business_ids_reviewed <- unique(dfr[, "business_id"])
    businesses_reviewed <- dfb[match(business_ids_reviewed, dfb[, "business_id"]),
        ]
    # Find cities these businesses are located in
    cities_with_businesses_reviewed <- unique(businesses_reviewed[, "city"])
    # Only consider those businesses that are in the first 'x' cities (x = 1
    # here)
    cities_to_consider <- cities_with_businesses_reviewed[1:num_cities]
    # Now consider only businesses (that we already know have reviews) that are
    # in such cities
    businesses_to_consider <- businesses_reviewed[which(businesses_reviewed[,
        "city"] %in% cities_to_consider), ]
}

# Return businesses that fall in the categories
restrict_categories <- function(categories) {
    # Limit businesses to the restaurants category
    businesses_to_consider <- dfb[which(sapply(dfb$categories, function(x) categories %in%
        x)), ]
}

# Return users that have at least min_user_reviews number of reviews
restrict_users <- function(reviews_to_consider, min_user_reviews, max_reviews) {
    library(dplyr)
    # Limit users to those who have written reviews
    users_to_consider <- reviews_to_consider %>% group_by(user_id, business_id) %>%
        summarize(count1 = n()) %>% group_by(user_id) %>% summarize(count = n()) %>%
        filter(count >= min_user_reviews) %>% arrange(desc(count)) %>% mutate(total_count = cumsum(coun
        filter(total_count < max_reviews) %>% select(user_id)
}

# Return a set of (user_id, business_id, rating) data frame Filter data to
# only the given categories, top num_cities, with users who have written
# min_user_reviews number of reviews
construct_review_set <- function(categories, num_cities, min_user_reviews, max_reviews = 1e+05) {
    # Use only stars for rating
    reviews <- dfr[, c("user_id", "business_id", "stars")]

    # Restrict reviews to businesses by cities and categories
    b1 <- restrict_cities(num_cities)
    b2 <- restrict_categories(categories)
    b <- merge(b1, b2, by = "business_id")
    reviews_to_consider <- reviews[which(reviews[, "business_id"] %in% b[, "business_id"]),
        ]

    # Restrict reviews further to users by number of reviews
    u <- restrict_users(reviews_to_consider, min_user_reviews, max_reviews)
    reviews_to_consider <- reviews_to_consider[which(reviews_to_consider[, "user_id"] %in%
        u[, "user_id"]), ]
}

reviews_to_consider <- construct_review_set(categories = "Restaurants", num_cities = 10,
    min_user_reviews = 20)
```

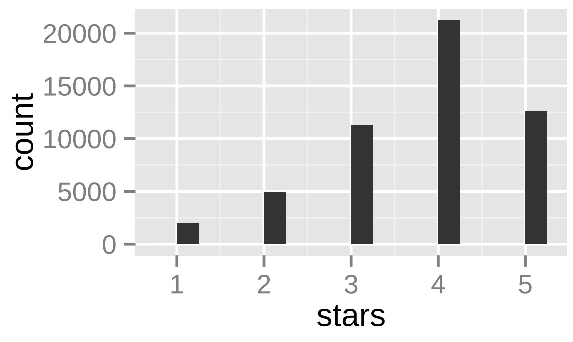
A few interesting observations from this exploratory analysis, and the graphs below (the code for generating the graphs has been omitted for space reasons):

- A large number of users and businesses have very few reviews.
- Only one user wrote > 500 reviews, and one business got > 200 reviews.
- Businesses get more ratings at 4 or 5, than 1 or 2 (probably indicating that people write reviews more often when they like a business).









Results

We will now evaluate several algorithms to provide personalized recommendations for users. We will consider the following models:

1. Random: Each user rating for a given business is chosen at random. This is a baseline comparison.
2. Popular: This is the standard average-stars-based algorithm. Businesses with the highest average ratings are recommended first.
3. User-Based Collaborative Filtering (UBCF): Each user is associated with other users based on their rating similarity, and businesses are recommended in order of other similar users' ratings. A full paper describing these algorithms is here.

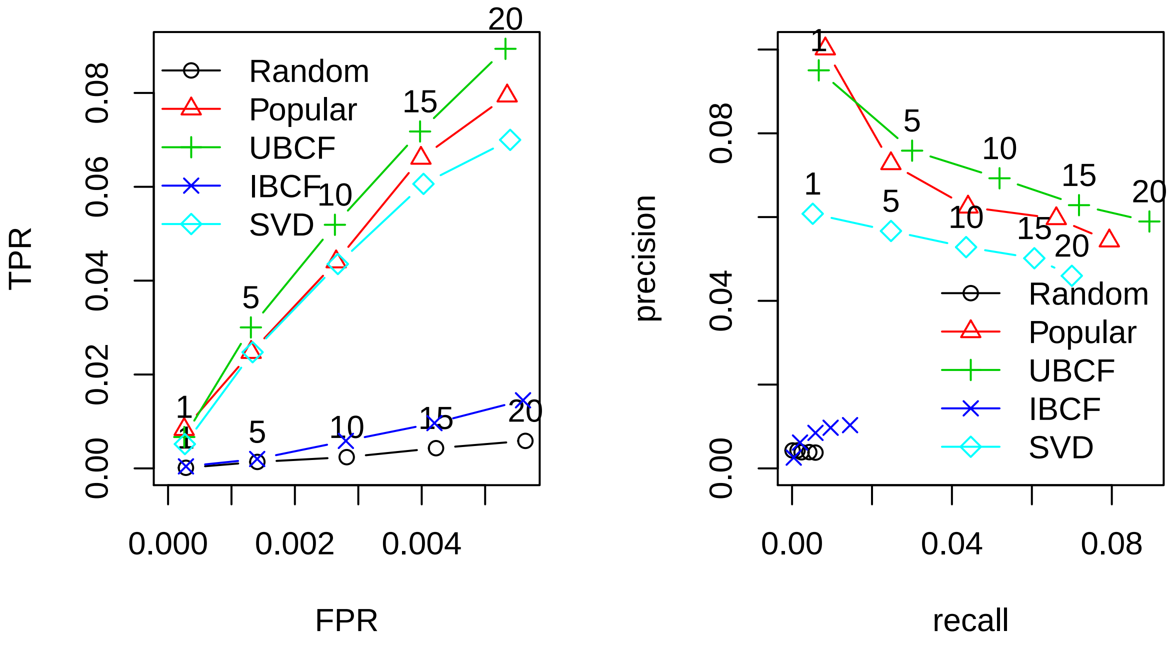
4. Item-Based Collaborative Filtering (IBCF): Each business is associated with other businesses based on their rating similarity, and businesses are recommended based on this user's rating of other similar business.
5. Singular Value Decomposition (SVD): This algorithm takes the data as a whole, finds groupings of data, and eliminates lower-significance dimensions.

The code below implements all the algorithms, and provides the graphs showing the results. We note that UBCF works the best; we will discuss the results in more detail in the section below.

```r
recommender_results <- function(reviews_to_consider, min_known_ratings, number_runs = 2,
    algorithms_to_run = c("Random", "Popular", "UBCF", "IBCF", "SVD")) {
    library(recommenderlab)
    algorithms <- list(Random = list(name = "RANDOM", param = NULL), Popular = list(name = "POPULAR",
        param = NULL), UBCF = list(name = "UBCF", param = list(method = "Cosine",
        nn = 50, minRating = 1)), IBCF = list(name = "IBCF", param = NULL),
        PCA = list(name = "PCA", param = NULL), SVD = list(name = "SVD", param = NULL))

    # Run the above algorithms with cross-validtion
    r <- as(reviews_to_consider, "realRatingMatrix")
    scheme <- evaluationScheme(r, method = "cross", k = number_runs, given = min_known_ratings,
        goodRating = 4)
    results <- evaluate(scheme, algorithms[algorithms_to_run], n = c(1, 5, 10,
        15, 20))
}
```

```r
set.seed(6789)
num_cv_folds <- 4
results <- recommender_results(reviews_to_consider, min_known_ratings = 20,
    number_runs = num_cv_folds)
```

```r
par(mfrow = c(1, 2))
plot(results, legend = "topleft", annotate = c(1, 3))
plot(results, "prec/rec", annotate = c(3, 5))
```



Discussion

We will now discuss the results from running the algorithms described in the model above. We have limited ourselves to using star ratings only, considered only businesses in the top 10 cities in the Restaurants category, and users who have rated at least 20 businesses. Some notes:

1. User-Based Collaborative Filtering (UBCF) is the best performing of all the algorithms. Note that it dominates all other algorithms in both the TPR vs FPR (True Positive Rate vs False Positive Rate) and Precision/Recall graphs (I have omitted further interpretation of these graphs because of space constraints, see this paper instead on ROC curves and how they relate to precision/recall graphs).
    - In general, the Popular algorithm is pretty hard to beat.
    - We need at least 20 ratings from a user, before the UBCF algorithm can "learn" the user's preferences, and make personalized recommendations. Again, we have omitted graphs that show how UBCF's performance suffers when there are less than 20 ratings per user (because of space constraints).
    - UBCF had to be customized to make it performant. In particular, see the 'nn' (nearest neighbor) and the minRating parameters.
2. Cross-validation (4-fold) has been done to limit overfitting (see the num_cv_folds parameter below).
3. All algorithms fare better than the baseline Random algorithm.
4. IBCF and SVD do not outperform the Popular algorithm. The fact that IBCF underperforms indicates that user preferences do matter in recommending businesses.

Future Work

There are several directions for future work in improving the recommendations.

- We have only used star ratings in this paper. We can enhance it with other signals like the reviews, votes, tips etc.
- We can generate recommendations within a given radius of a zipcode, instead of at a city-level.
- Different algorithms might work better for different categories/cities. Blending the results appropriately from different algorithms should be explored further.
- Investigate algorithms for cold start situations, where a user did not give enough ratings to learn their preferences.

Conclusions

We have shown in this paper that recommending the businesses best suited to a given user is feasible from the data set provided. We have shown that the User-Based Collaborative Filtering (UBCF) algorithm performs the best, when users have provided at least 20 other ratings. We have also outlined future directions for work in perfecting recommendations to users.