Dish and Restaurant Recommendation

The main objectives of this task were:

a. Recommend dishes to the user given a particular cuisine that he/she might be interested
b. Recommend restaurants to the user given one dish or set of dishes from a particular cuisine that was selected to try.

Indian cuisine was selected for building the dish and restaurant recommendation visualization using different ranking methods. The overall ranking methods were generic enough to test with other cuisines as well.

Dish and Restaurant Statistics:

Based on the Text Retrieval techniques, it was decided to build two set of statistics:

a. For a dish, calculate number of **unique reviews** that mentioned the dish, number of **unique restaurants** mentioned it and **average rating** received (calculated by considering all the review ratings that mentioned the dish). (similar to an *inverted* index - where we maintain word level counts and total unique documents in which they occurred)
b. For a given restaurant and dish combination, number of **unique reviews** that mentioned the dish and **average review rating** received (calculated by taking average of all the reviews that mentioned the dish for a given restaurant) - This is similar to *postings* (where we maintain dish's reviews counts in a particular restaurant)

(***Note:*** *average review rating seem like a better start than sentiment analysis - Stanford sentiment library and R tm.plugin.sentiment packages were experimented to get sentiment scores based on text in review but a lot of scores were completely contradictory. The sentiment scores from these libraries did not match with the actual semantics of the review*)

Detail Process:

The reference details regarding review and restaurant were outputted by modifying existing ***py27_processYelpRestaurants.py*** to produce more details regarding a review such as restaurant_id (restaurant that was reviewed) , stars (rating given) and review_id. Restaurant level reference information was also outputted :- restaurant_id, overall rating given for the restaurant and name of the restaurant.

Using the reference details logged from the above process i.e. all the reviews (along with reference information) and all the restaurants details (along with reference information) for Indian cuisine, the dish level statistics (similar to inverted index) and dish-restaurant level statistics (similar to postings) were calculated from the all Indian restaurants reviews in **Yelp academic data**. A new python script called **dishes_counts.py** was written that will do following tasks:-

a. Read all the quality unique Indian dishes obtained from Task 3
b. Iterate through all the reviews of Indian restaurants and maintain the following details

i. total count of unique reviews that mentioned the dish.
ii. restaurant_ids of review that mentioned the dish
iii. total count of unique reviews that mentioned the dish (by restaurant)
iv. ratings given by each review that mentioned the dish
v. ratings given by each review that mentioned the dish (by restaurant)

c. Calculate total unique restaurants that mentioned the dish from step b (ii)
d. Calculate the overall average rating given to a dish (using data collected from step b (iv))
e. Calculate the average rating given to a dish at a restaurant (using data collected from step b (v)

All the dish level statistics and dish-restaurant level statistics was outputted into 2 csv files:

1) Indian_dishes_reviews_counts.txt (inverted index for dishes - total reviews counts and overall average rating for dishes)
2) Indian_dishes_postings (postings like statistics - reviews counts and average rating for dishes by restaurants)

Above csv files were ingested into R for the dish and restaurant recommendation tasks.

Dish Recommendation:

Some analysis was quickly done for on total reviews counts for Indian dishes.

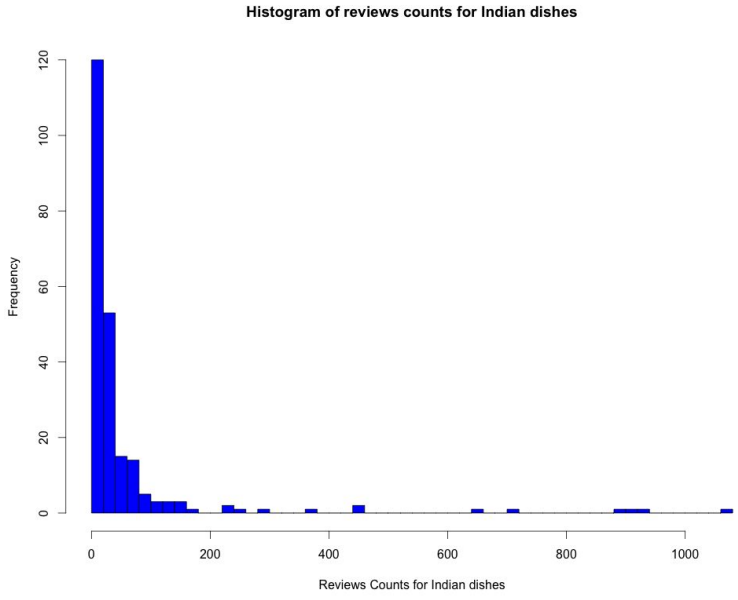


Fig 1(a) - Histogram of Reviews Counts (for Indian dishes)

We can easily see that distribution is skewed (See Fig 1(a)). It was observed that there were few dishes which were mentioned a lot of times than some others. To make sure that few dishes do not dominate the weightage in dish recommendation ranking system, a sub linear transformation was done on overall reviews counts by dishes.

A simple log(1+x) function was applied for sub-linear transformation. A standard technique from the Text Retrieval techniques used for transforming the term frequency counts.

Here is the resulting distribution:

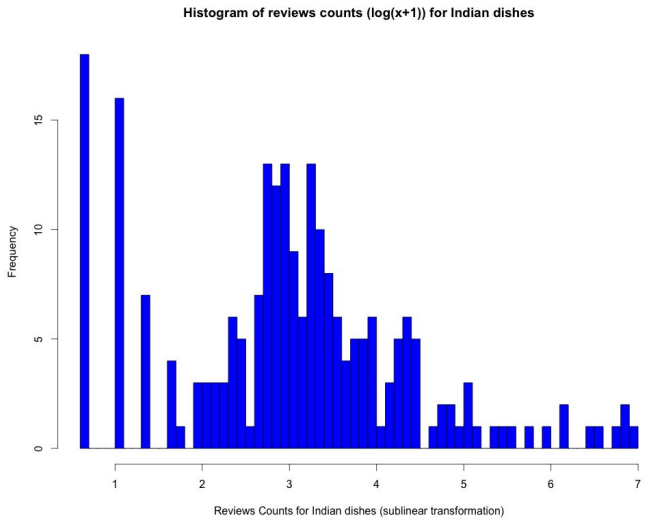


Fig 1(b) - Histogram of Reviews Counts after sublinear transformation

We can see that the reviews counts are now more evenly distributed. The result is much more smooth and close to normal distribution (See Fig 1 (b)). The BM25 transformation was later applied on original reviews counts to get more controlled transformation using **K value.**

The transformed reviews counts for Indian dishes were used in the dish recommendation ranking system. The transformed reviews counts was multiplied by the overall average rating received for the dish to get the overall score for a dish. Based on the above analysis here the three dish recommendation ranking methods that were experimented:-

$d_{score} = c(d) * Average\ Rating(d)$ ……(D1) [Simple **linear** ranking system]

$d_{score} = c(r) * Average\ Rating(d)$ ……(D4) [Simple **linear** ranking system]

$d_{score} = log(1 + c(d)) * Average\ Rating\ (d)$ ……..(D2) ..[**sublinear** transformation]

$d_{score} = \frac{(1+K)*c(d)}{(c(d)+k)} * Average\ Rating(d)$ ……..(D3) [**BM25** transformation]

where:

d_{score} = overall score for the dish

c(d) = Reviews Counts (total unique reviews that mentioned the dish)

c(r) = Restaurant Counts (total unique restaurants that mentioned the dish)

Average Rating(d) = Average Rating obtained from all the reviews that mentioned the dish d

K = tuning parameter for BM25 transformation
Dish scores were obtained using the above methods D1,D2, D3and D4. Implementation was done using basic R functions like lapply and aggregate functions.

Using the above ranking systems, we received the following dish recommendation visualizations from list of around 230 Indian dishes. (Please see Fig 1(c) for visualization using D1 ranking method, Fig 1 (d) for visualization using D2 ranking method) and Fig 1(e) for visualization using D3 ranking method. Fig 1 (f) for visualization using D4 ranking method. **R (ggplot)** was used to obtain the following visualization. The color represents the average rating received for the dish. It goes from orange (for *min* average ratings) to yellow (*medium*) to green(*very good or max*). On x-axis is the Indian dishes and y-axis indicates the score. The high score indicates high rank/highly recommended dish.

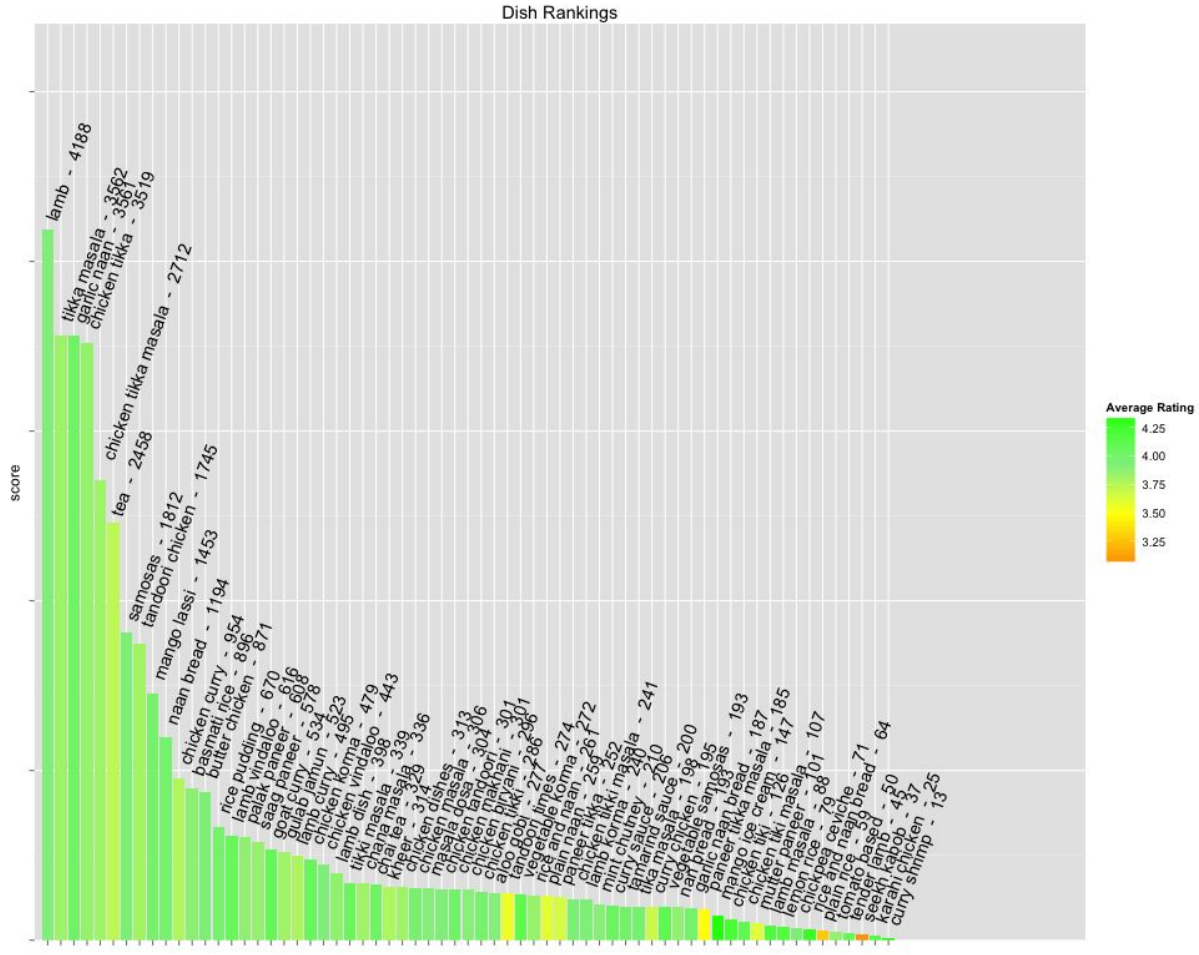


Fig 1 (c) - Dish Rankings using D1 method(for top 50 dishes and then every 10th dish)

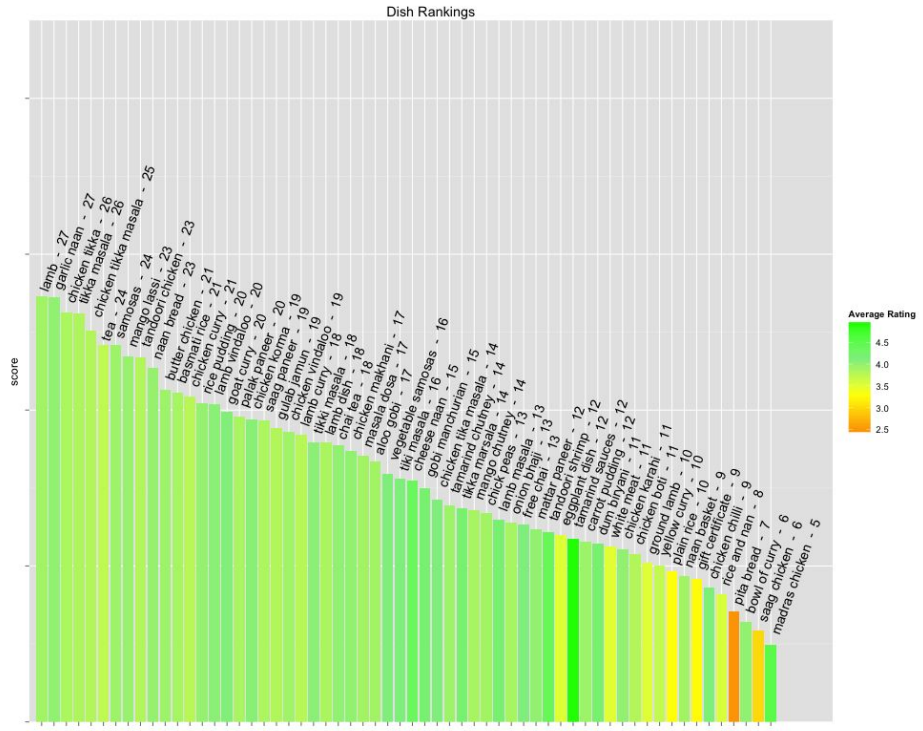


Fig 1(d) - Dish Rankings using D2 method(top 25 dishes and then every 5th dish in order)

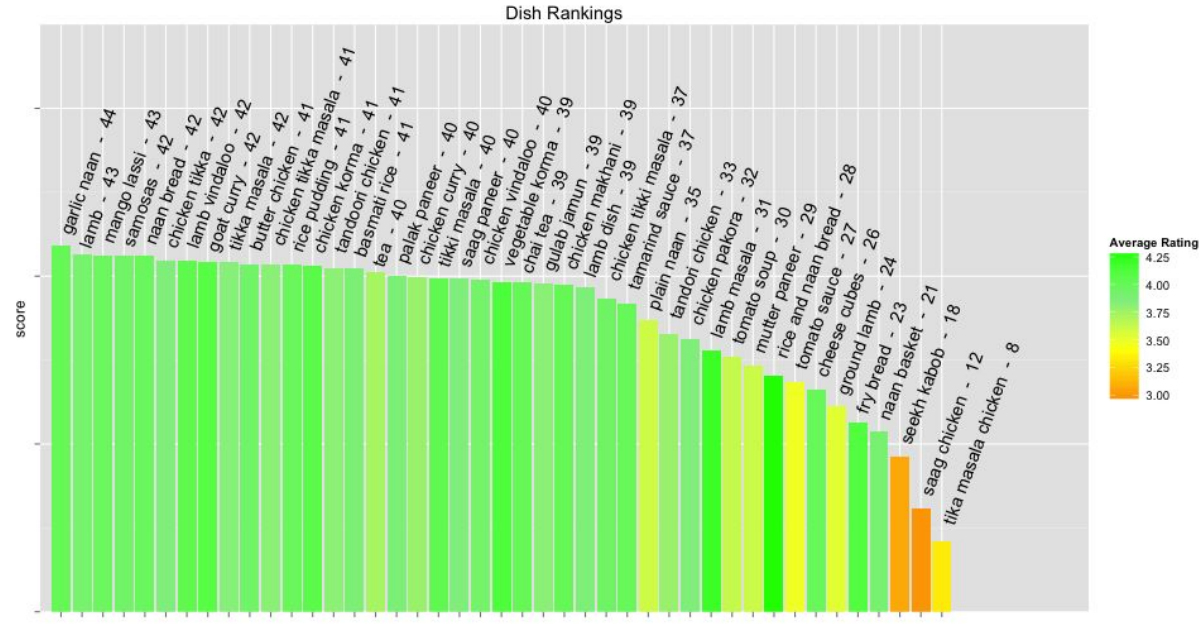


Fig 1(e) - Dish Rankings using D3 method(top 25 dishes and then every 5th dish in order) and K=10

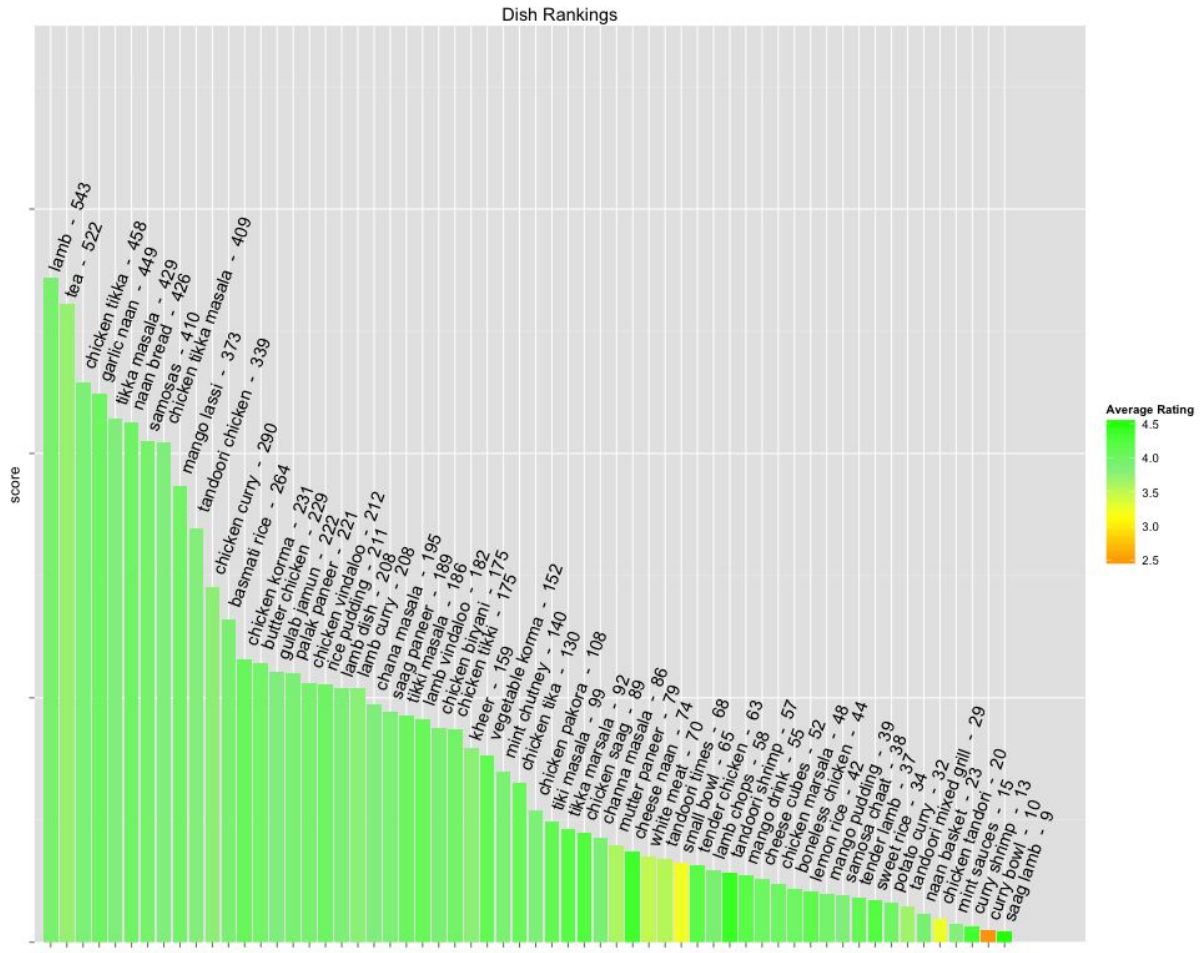


Fig 1(f) - Dish Rankings using D4 method(for top 25 dishes and then every 5th dish)

Dish Visualization analysis:

It was observed from 1(d) and 1(f) that no matter if you take total reviews counts or total restaurants counts for dish ranking, the top dishes are very similar. For visualizations 1(e) and 1(f) using improved ranking methods, we selected reviews counts because that contains more detail level counts (similar to term frequency rather than document frequency).

The top dishes in all the visualizations are pretty close to intuitive popular Indian dishes like **“garlic naan”, “chicken tikka masala”, samosas, mango lassi and tandoori chicken.** The color displays the average ratings from all the reviews that mention that dish. The color shade here vary in the visualizations for same dish because of different min/max range for Orange - Yellow - Green. As expected, since average rating is part of our scoring function - we can see some low rated dishes (according to the review rating) towards the end like *tomato*

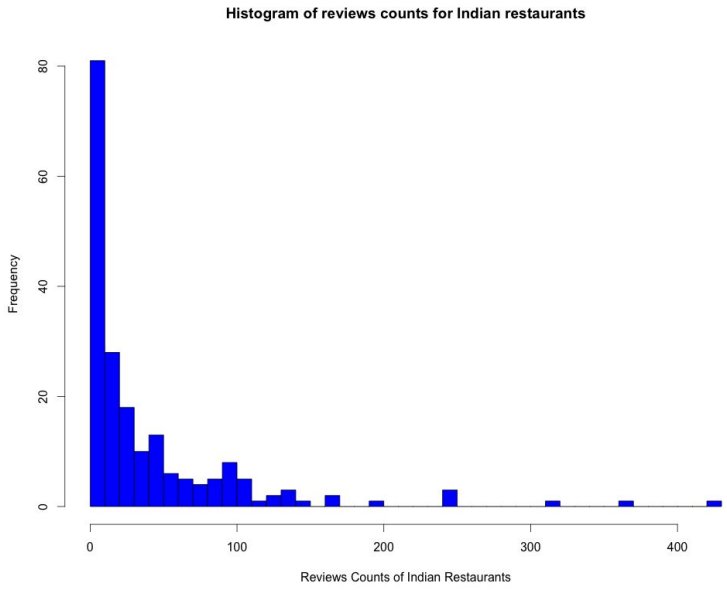
soup. Interestingly, some unusually ordered dish names also automatically goes toward the end like tika masala chicken (supposed to be chicken tikka masala) and fry bread (which is not really an indian dish) or tiki masala chicken (spelling mistake and not correctly ordered).

Dish visualization in Fig 1(d) and (e) - gives much more balanced rankings than in Fig 1 (c) which ranks linearly.

With the evaluation system in place, **K value in D3** can be tuned to get the maximum accuracy.

Restaurant Recommendation:

Some analysis was quickly done on reviews counts by restaurant.



We can see that a large number of Indian restaurants have very few overall reviews. It is very likely that our restaurant recommendation ranking method will end up giving more weight to restaurants with more reviews if we do not normalize the reviews counts for Indian dish at a particular restaurant.

To normalize the counts 3 methods were experimented:-

a. Diminishing returns for very high frequencies (sub-linear transformation like we obtained for dish review counts)
b. Restaurant reviews counts normalization (similar to pivoted length normalization in text retrieval)
c. BM25 transformation and pivoted length normalization combined.

Based on the above three methods, below are three ranking methods that were experimented for restaurant ranking system:-

$r_{score} = \sum_{d \in q \cap r} (log(1 + c(d,r)) * Average\ Rating(d,r)$ …….(R1)[Simple Sublinear transformation]

$r_{score} = \sum_{d \in q \cap r} \frac{c(d,r)}{1-b+b*\frac{|r|}{average|r|}} * Average\ Rating(d,r)$ …….(R2)[Pivoted Length Normalization]

$r_{score} = \sum_{d \in q \cap r} \frac{(K+1)c(d,r)}{(c(d,r)+k(1-b+(b*(\frac{|r|}{average\ |r|})))} * Average\ Rating(d,r)$ …….(R3) [BM25 transformation and pivoted length normalization]

where:

a. r_{score}= total score for the restaurant r
b. $d \in q \cap r$ = indicates dish that is mentioned in at least one of the reviews for restaurant r
c. c(d,r) = count of reviews in which dish d was mentioned for a restaurant r
d. Average Rating(d,r) = Average Rating given by users for a dish d at a restaurant r (based on all the reviews that mentioned the dish d for restaurant r)
e. |r| = Total Reviews Counts for the restaurant r
f. Average |r| = Average Reviews Counts for a restaurant.

Thus total score of restaurant r given set of dishes d in query q is summation of individual scores obtained for dish d in restaurant r. The way we calculate individual dish score at restaurant R varies in R1, R2 and R3.In R1, we use simple sub linear transformation. In R2 we penalize the reviews counts for restaurants which have lot of reviews. For R3 we combine the techniques from R1 and R2 and introduce BM25 transformation for tuning.

Scores were calculated in R using postings information (dish-restaurant statistics). Functions like **merge, lapply** were used to do the necessary joins and transformation. At the end **aggregate** function was used to get total scores for each restaurant.
ggplot function was used for visualizations. The color display the average rating given by user for the **combination of dishes** from the query at that restaurant. R implementation was made configurable for dynamic query with one or more list of dishes as input.

Here are the visualizations of the top 50 restaurants (and then every 10th in order) from list of total 202 restaurants based on R1, R2 and R3 ranking methods:-
Query was assumed *as "garlic naan", "chicken tikka masala" and "gulab jamun".(for* ***2(a),2(b) and 2(c)****) The color range indicate average ratings. It varies from orange (low ratings) to yellow (neutral to slightly positive ratings) to green (very good or positive ratings).*On x-axis is the

Indian restaurants and y-axis indicates the score. The high score indicates high rank restaurant/highly recommended restaurant.

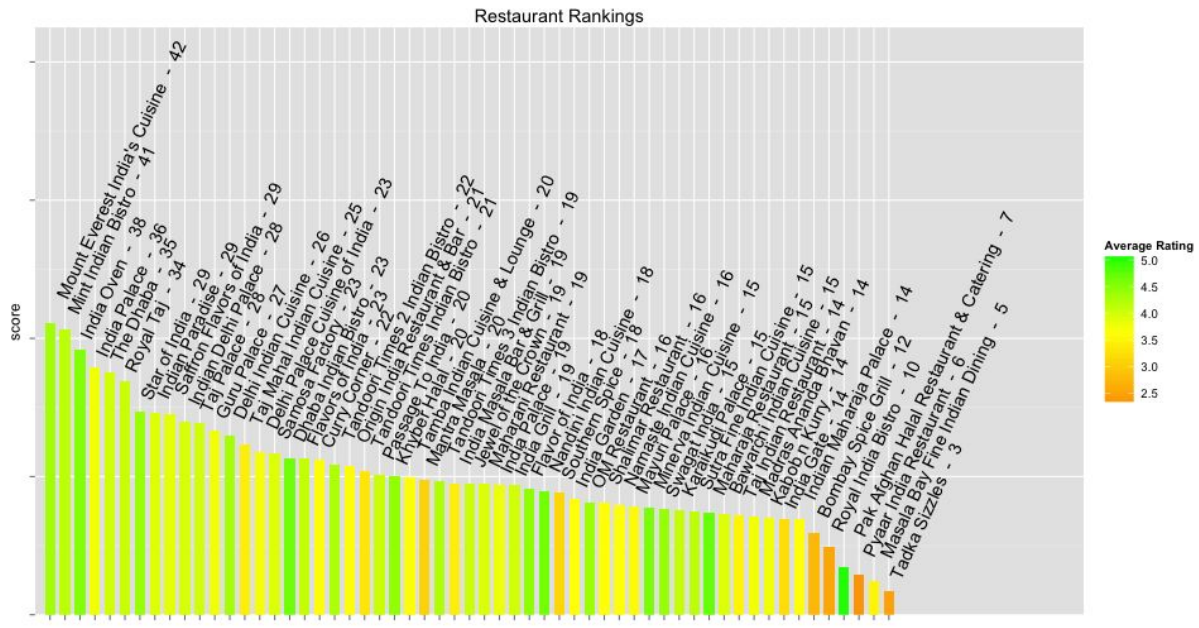


Fig 2 (a) - Restaurant Rankings based on query on Indian dishes “garlic naan”, “chicken tikka masala” and “gulab jamun” and ranking method R1 (sublinear transformation)

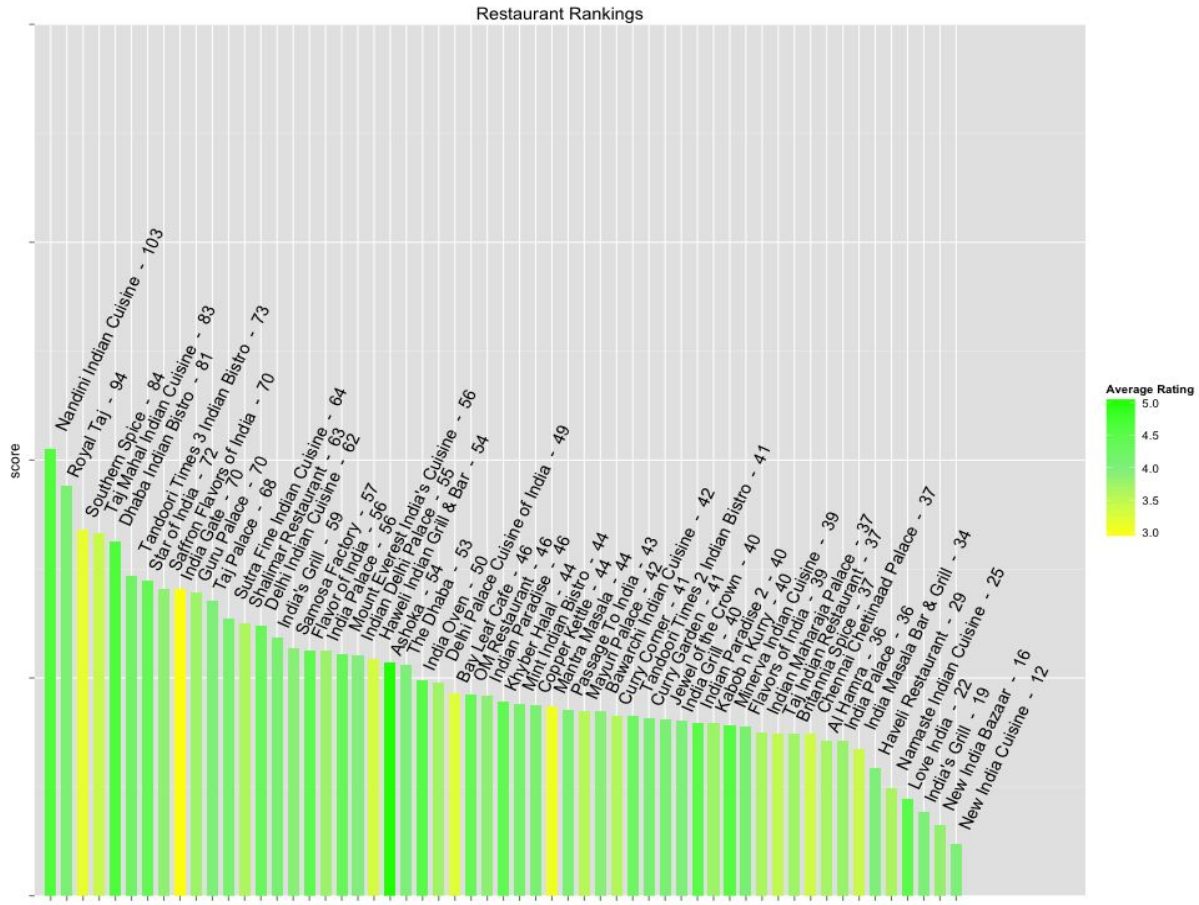


Fig 2 (b) - Restaurant Rankings based on query on Indian dishes “garlic naan”, “chicken tikka masala” and “gulab jamun” and ranking method R2 (pivoted length normalization, b=0.9)

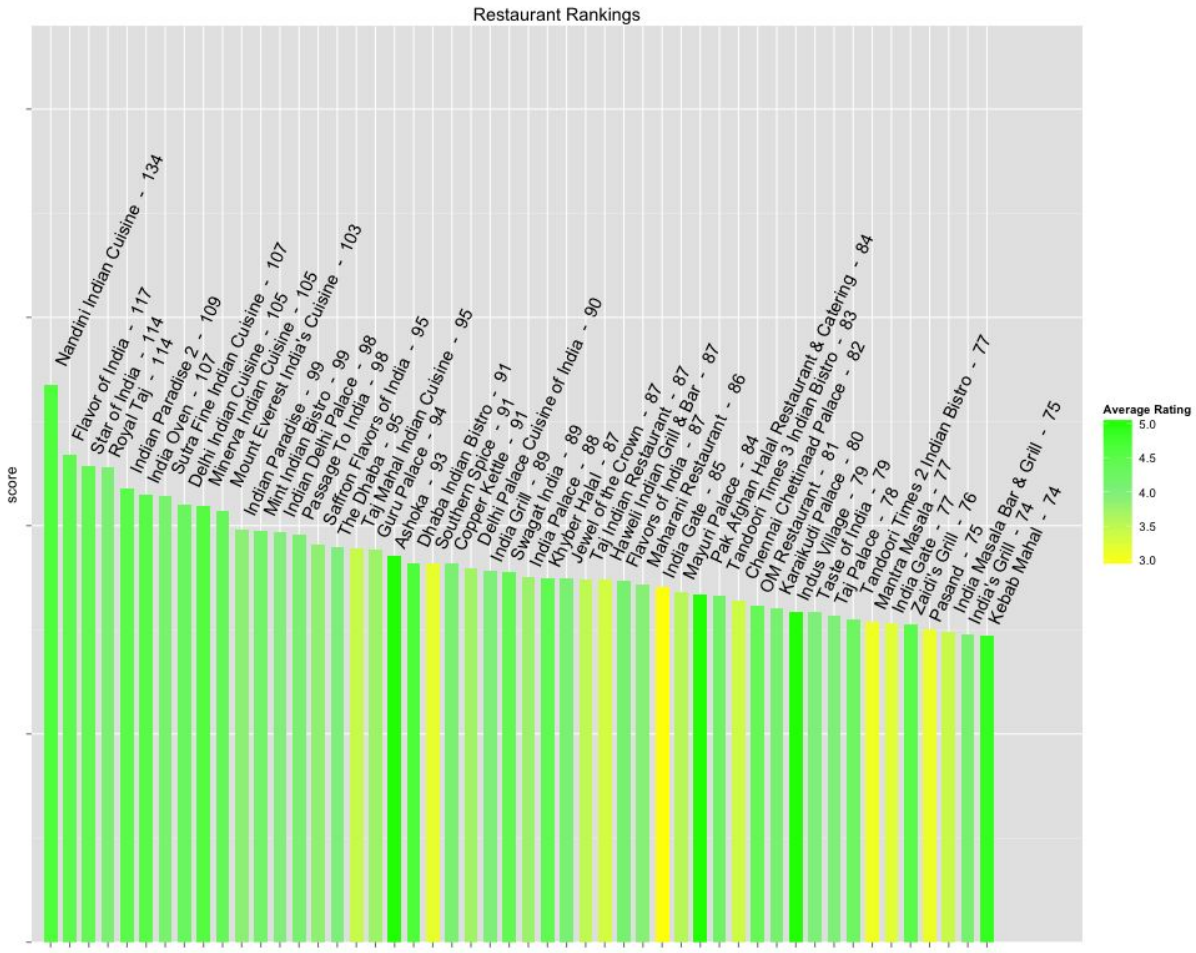


Fig 2 (c) - Restaurant Rankings based on query on Indian dishes "garlic naan", "chicken tikka masala" and "gulab jamun" and ranking method R3 (BM25 transformation and Pivot Length normalization with k=10,b=0.9)

<u>Comparison of 2(a), 2(b) and 2(c):</u>

For 2(a), restaurant rankings weighted a lot on number of reviews counts even after sub-linear transformation. 2 (b) did a little opposite effect on the rankings - it gave lot more weight to restaurants with low number of reviews but high rating because of the pivoted length normalization. A lot of restaurants with less number of reviews came up in the rankings because of very low denominator values. For 2(c) since BM25 Transformation and Pivoted Length normalization was used, use of K and b values controlled the penalty and transformation leading to much more balanced rankings where restaurants with less and large number of reviews both made up the rankings.

Conclusion:

From the above analysis we can observe, that how each component of recommendation system is important in ordering the restaurants and dishes correctly. We compared linear, sublinear and BM25 transformation and observed results for each of the method for both dish and restaurant recommendation. The average rating of the dish from all the reviews and average rating of a given dish(es) in query at a particular restaurant was also one of the key components. The average ratings extract the key highlight ratings given by the users. The sentiment scores associated with the dish could have been even more accurate as part of score. But it was observed that current sentiment libraries (from R package tm.plugin.sentiment and stanford sentiment analysis library) do not give correct score. In fact for some of the reviews where sentiment should have gotten a very high (positive) score were getting very low (negative) score. The formal evaluation of the results was the only thing that was left out because of the time constraints and lack of real user feedback form the recommendations provided. But overall a lot of key concepts like inverted index, postings, sublinear transformation, BM25 and pivoted length normalization from text retrieval methods could be directly applied for dish and restaurant recommendations ranking systems. The formal evaluation can enable to get more accurate value of K in BM25 transformation and b in pivot length normalization leading to more interesting comparison of user preferences and rankings. The ranking system can be further personalized according to user location to consider only the restaurants that are within 5 miles of user's current location as an example. The implementation can be easily plugged into online recommendation system using user level information.