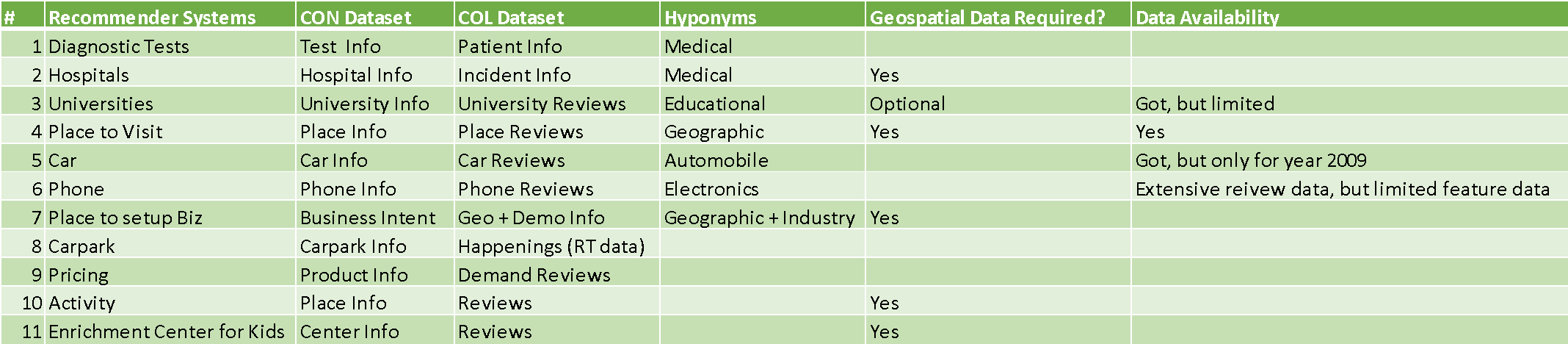
**Restaurant Recommender System – Yelp**

# Overview

Initial understanding about a recommender system (prior to RS lessons), was mere filtering and sorting. Through the course, developed a great deal of understanding and how powerful even the anonymous data can be.

Given the free format of the project, the following options were explored for building recommender system.



Together with the team, analyzed various datasets on both collaborative and content-based aspects, along with NLP component, before finalizing to proceed with option #4. Yelp Data from Kaggle site was quite appropriate, although it had its challenges. In the following sections, I have bulleted them with a brief detail.

Datasets explored:



# Challenges during Data preparation

1. Yelp Data was massive and was available in 7 different sets of csv files (above screen shot has site details). Data exploration across these files was very time consuming. Business and Review data being key further analysis, we focused only on 2 files i.e., yelp\_business.csv (Business Profile Set) and yelp\_review.csv (Review Set), to identify what kind of businesses the Yelp deals with and how is the review data spread across. Few insights are

Yelp\_review.csv

* 9 features/columns
* 5.2 million unique reviews
* 1.3 million unique users
* 174K unique business

Yelp\_business.csv

* 13 features/columns
* 174K unique businesses

1. Data was not clean.
   1. Business names and addresses had non-ascii characters
   2. All business categories for each business\_id was a single composite value, separated by semi-colon
   3. Review Text ranged from 0 size to > 3000 char length. Further scrutiny revealed that the review text had non-ascii characters, http refs, and some unrecognizable characters.
2. In an effort to limit the data size (PC supportability and time efficient) without compromising the key inputs required for recommender system,
   1. truncated review text and sparse columns in both the sets and joined them by ‘business\_id’ column.
   2. plotted the data state wise and filtered only for businesses that contain categories as ‘restaurant’.
   3. Random sampled reviews for 2000 restaurants from this set.
   4. Effect of these on review set is: 289K unique reviews across 2000 unique restaurants in state ‘Naveda (NV)’.

# challenges during building model/system

As review data had explicit ranking, building CF model using ALS was quite straight forward. However, building content-based model was different. Business/ Review data lacked explicit features for businesses (except the category itself). Hence, we had to perform NLP to extract information associated with the business and relate them to the explicit ratings to find the association of the rating corresponding to the category of the restaurant.

NLP was performed over 289K records, to extract feature matrix (binary-dtm), for Verbs, Nouns and Adjectives, for all 2000 unique restaurants. This task caused PC to hang, crash, etc., after running for a few hours. This challenge was overcome on one of the group member’s PC which could complete the NLP, in overnight + 7hours time.

Another challenge we faced during evaluation of content-based model. Content-based model was built over the Noun/Verb/Adjective dtm matrix (after reducing sparsity with 0.995 the matrix had 12K+ features). We did random sample split the data into to 2 parts 9:1 ratio for train and test sets respectively. On train data, these implicit features were further substantiated with respective user ratings (stars 1 to 5) to get normalized scoring across the features. This normalized component was used to predict the scoring for each test restaurant. RMSE values between predicted and actual star was used to evaluate the performance of CB model (Average ~1.2).

While there are plenty more challenges in coding and visualizing every transformation of the data, one more specific challenge that I want to highlight here is, combining the output of CF and CB models and ranking them according to their respective ratings. After multiple rounds of discussions, we concluded to weight the output of each model (top 10 recommendations) based on the inverse of RMSEs (1-RMSE, i.e., correctness in prediction) as ‘weight’ for that model. This helped us to sort the output of both models (hybrid) with right statistical significance.

One other challenge towards packaging the system was to get the shiny UI behave appropriate when the system is started. UI was either throwing the error in the beginning due blank input values or doing multiple recommendations as we type in the input boxes. Looks minor, but GUI being front face of the core system, needs to portray sophistication that went in building the brain behind.

# Lessons learnt

Some of the significant lessons learnt are:

1. Understand the data better, before starting the model building. The re-work can be reduced by many folds.
2. Explore libraries such as ‘slam’ which internalize the memory usage and thus improve the efficiencies
3. Task identification, time-lines and distribution of tasks along with critical path for completion needs to be laid out to eliminate workload imbalances.
4. Develop statistical significance of normalizing, methods to be applied and evaluation techniques. This helps a long way in defining the right architecture and justifying the model.