



# Yelp Dataset Challenge

ILS-Z534 Final Project

12.09.2019

---

Indiana University Bloomington

Arpit Shah

Neha Pai

Nikita Bafna

## Introduction

The Yelp dataset challenge provided business, reviews, ratings, user, checkin and tip data to research, analyze the data and figure out new patterns that may turn out to be favorable to solve interesting problems.

## Goals

In this project we dealt with two interesting tasks.

1. First task was to recommend businesses to users using two approaches
2. Second was an open task wherein we could propose our own problem statement and solution.

## Languages and Tools used

1. Python
2. Flask
3. Whoosh
4. Bootstrap

## Task 1: Recommend business to users

Question : Our aim is to recommend business restaurants to users using two approaches and evaluate both the methods in order to determine which is a better approach to recommend business with data that we have generated.

## Exploratory Data Analysis

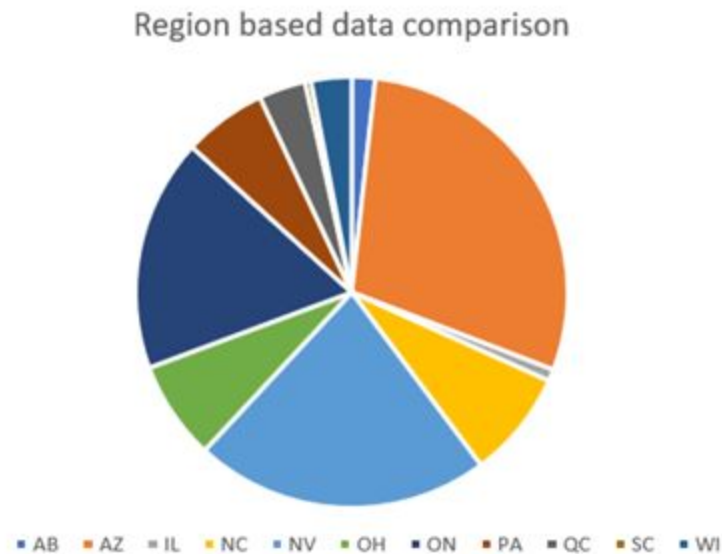
The dataset comes from Yelp Dataset challenge ([https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge)). There are three json dataset files used, which are business.json, review.json and user.json. The Yelp data we extracted has below data distribution:

- 61K businesses
- 481K business attributes
- 366K users
- 2.9M edge social graph
- 500K tips
- 1.6M reviews

Since the dataset was huge, we had to filter the data as per our objective.

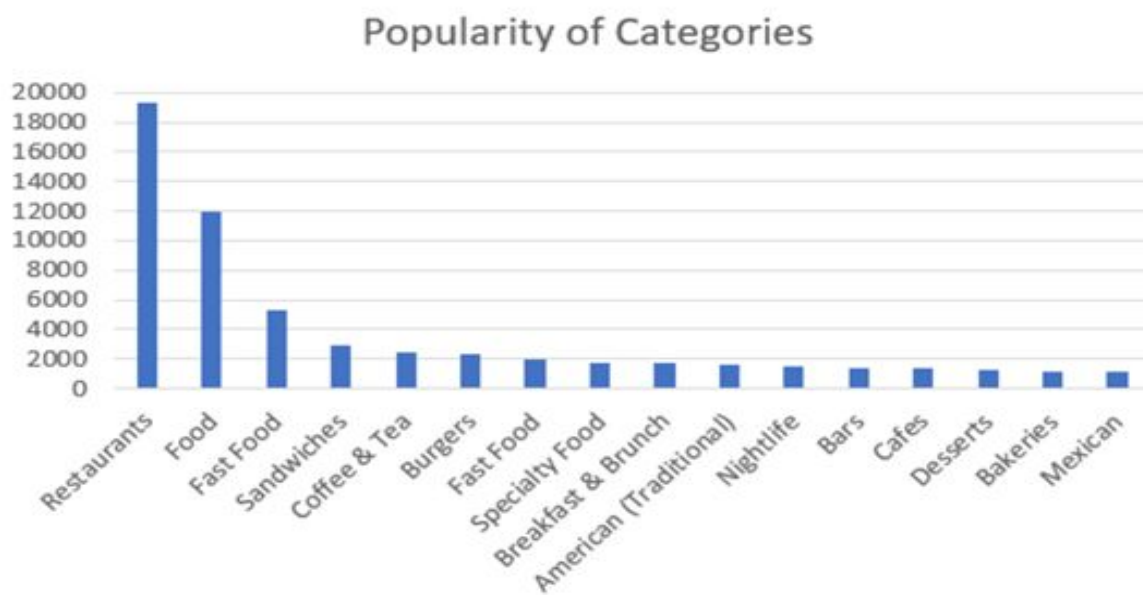
To filter our dataset we tried to visualize the various aspects of the data.

Below is the distribution of top businesses state wise:



Arizona and Nevada are regions with maximum data

Below is Category popularity Chart:



Restaurants and Food were the most popular categories

We also populated word cloud based on the Categories:

- Chose Food and Restaurants category
- We consider only 'OPEN' restaurants
- Businesses having at least 50 reviews
- State = NEVADA

Business Dataset		User Dataset	Review Dataset
type:business	latitude	type:user	type
business_id	longitude	user_id	business_id
name	stars:rating	name	user_id
neighbourhoods	review_count	review_count	stars:rating
full_address	categories	average_stars	text
city	open;True/False	votes	date
state			votes:useful,funny,cool

### Table 1 Variables in Raw dataset

## Approach:

In our recommendation system for Yelp, we use different methods to predict ratings from users to businesses.

We consider similarity between users and similarity between businesses. On one hand, we hope to find users with similar preference of businesses, then recommend the businesses they like to each other. For example, user A and his friend have same taste, so it is very likely that A would go to the restaurant his friend recommended to him and likes it. On the other hand, based on a user's rating history, we gain information of those businesses he/she gave high ratings, then recommend businesses similar to those preferred businesses.

Also, enhanced method is applied to decompose user-business matrix from features of users and features of businesses at the same time. We expect that using matrix operations would make the recommendation easier computational realized, more scalable and efficient.

Our approach started with implementation of :

1. Collaborative filtering
  - 1.1. Memory Based approach :
    - 1.1.1. User based
    - 1.1.2. Item based
  - 1.2. Model Based approach:
    - 1.2.1. SVD Decomposition
2. Hybrid approach : Content based + Collaborative approach

### 1. Collaborative filtering:

The key idea behind CF is that similar users share the same interest and that similar items are liked by a user.

#### 1.1 Memory Based Approach:

	Business 001	Business 002	Business 003	Business 004	Business 005	Business 006
User 001	4	0	0	5	0	4.5
User 002	4.5	3.5	4.5	0	2	4

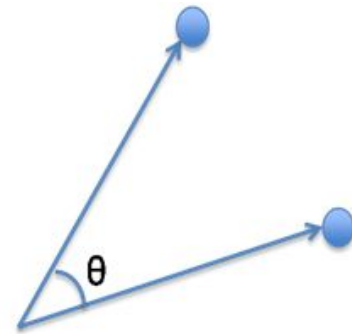
User 003	0	2.5	0	0	1	0
User 004	0	4	0	4	0	4.5
User 005	0	0	2	0	3	0
User 006	3.5	3	0	4.5	0	0

Table 2 User-Business Matrix

We first construct a user-item matrix  $X \in R^{U \times B}$ , where U is the total number of users and B is the total number of restaurants. Each element in the matrix is the rating user u gave to the restaurant b.

We name each row of X to be the user character vector of user, denoted by  $X_u^U$  for each user. Each column in X to be the item characteristic vector  $X_b^B$ , for the restaurant. With these characteristic vectors, we can compute the similarities between two users or two restaurants.

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$



The two approaches: User Based and Item Based, depending on which dimension of the user-item rating matrix is used to find similarities. Each one of these two strategies has pros and cons. User Based algorithms will search for “like minded individuals” following the assumption that similar users like similar items. Item Based algorithms will search for “item rated similarly by various user” following the assumption that if many users rated two items similarly, they will likely be similar items and worthy of recommendation; again, similar items are generally likely by similar users.

### 1) User-based prediction

The Rating prediction can take into consideration the rating normalizations. When using the mean centering approach, the equations become:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

To predict the rating the user will give to the restaurant, we can first compute the weighted-average of the ratings from all the users to the restaurants, with the similarities between user and other users as the weights, here we're subtracting the mean ratings of each user to eliminate the bias that some users tend to always give higher rating.

### 2) Item-based prediction

In order to produce an estimation of a rating for an unrated item, a weighted average of all available ratings is done, using as weights the correlation values computed with the similarity functions. The equation to compute a rating estimate in an item-based system is given as follows:

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|},$$

This method is also termed Item-Item recommendation. The aim of Item Based approach is to fix what is believed to be an important drawback of the User Based algorithm in real practice: it does not scale well.

To address this problem of scalability and Sparsity, we implement Model based approach using SVD Decomposition.

## 1.2 Model Based Approach:

We leverage a latent factor model to capture the similarity between users and items. Essentially, we want to turn the recommendation problem into an optimization problem. We can view it as how good we are in predicting the rating for items given a user.

$X$  denotes the utility matrix, and  $U$  is a left singular matrix, representing the relationship between users and latent factors.  $S$  is a diagonal matrix describing the strength of each latent factor, while  $V$  transpose is a right singular matrix, indicating the similarity between items and latent factors.

$$\begin{pmatrix} \hat{X} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} \\ m \times n \end{pmatrix} \approx \begin{pmatrix} U \\ \begin{pmatrix} u_{11} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix} \\ m \times r \end{pmatrix} \begin{pmatrix} S \\ \begin{pmatrix} s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix} \\ r \times r \end{pmatrix} \begin{pmatrix} V^T \\ \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix} \\ r \times n \end{pmatrix}$$

### Testing:

For evaluation of the predictions, we divided our data into 75:25 set.

We trained our model on 75% dataset and tested the results on 25% of the dataset.

### Evaluation of Collaborative filtering approaches:

We used two metrics for evaluation:

1. **Root Mean Square Error:** RMSE is just the square root of MSE. The square root is introduced to make the scale of the errors to be the same as the scale of targets.

$$RMSE = \sqrt{\frac{1}{n} \sum_{\{i,j\}} (p_{i,j} - r_{i,j})^2},$$



2. **Mean Average Error** : MAE computes the average of the absolute difference between the predictions and true ratings.

$$MAE = \frac{\sum_{\{i,j\}} |p_{i,j} - r_{i,j}|}{n},$$

Our Comparison results are:

RMSE Metric:

RMSE	
Memory Based	
User - Based	3.844
Item - Based	3.846
Model Based	
SVD Decomposition	3.835

MAE Metric:

MAE	
Memory Based	
User - Based	3.804
Item - Based	3.806
Model Based	
SVD Decomposition	3.783

## Hybrid Approach:

The approaches we considered so far, based on collaborative filtering only considered ratings, however it completely denies any information that can be extracted from contents. Another problem that arises is the cold start problem, i.e. the system fails to recommend businesses that no user has rated yet. A method to overcome these limitations is to use a hybrid approach that incorporates a content based system into the existing collaborative filtering framework.

In this approach, we apply a clustering technique to integrate the contents of items into the item-based collaborative filtering framework. The group rating information that is obtained from the clustering result provides a way to introduce content information into collaborative recommendation and solves the cold start problem. The results show that our approach contributes to the improvement of prediction quality of the item-based collaborative filtering, especially for the cold start problem.

For this approach we worked on - Categories and Attributes data of Business.json file

- Categories Picked : Food or Restaurant
- Only picked attributes where the value is **True**
- Nested attributes were concatenated with **underscore('\_')**
- Combined Attributes and Categories

**Category** : ['Mexican', 'Restaurants', 'Patisserie/Cake Shop', 'Food', 'Bars', 'Nightlife']

**Attribute** : {'BikeParking': 'True', 'BusinessParking': {'garage': False, 'street': False, 'validated': False, 'lot': **True**, 'valet': False}}

**Combined\_features**: ['Mexican', 'Restaurants', 'Patisserie/Cake Shop', 'Food', 'Bars', 'Nightlife', 'BikeParking', 'BusinessParking\_lot']

In [0]: data

Out[0]:

	business_id	combined_features
25	tstimHoMcYbkSC4eBA1wEg	Mexican Restaurants Patisserie/Cake Shop Food ...
128	sKhDrZFCJqfRNYlkHrIDsQ	Food Coffee & Tea RestaurantsPriceRange2_2 Res...
156	jScBTQtdAt-8RshaiBEHgw	Ethnic Food American (New) Burgers Food Restau...
174	6fPQJq4f_yiq1NHn0fd11Q	French Restaurants Creperies RestaurantsTakeOu...
176	k-dDZvTeLysoJvjHI-qr9g	Buffets Restaurants RestaurantsGoodForGroups R...
...	...	...
192456	qRymrsLmlA34bC8PvNoujg	Mediterranean Caterers Event Planning & Servic...
192458	P8uECqGqXWTwEndkh-6bQw	Sandwiches Pizza Chicken Wings Italian Restaur...
192475	p5rpYtxS5xPQjt3MXYYVEwA	Sandwiches Cafes Pizza Vegetarian Gluten-Free ...
192562	ngs16C2M_uTq2zXamlthVw	Farmers Market Caterers Food Street Vendors Ba...
192598	vIAEWbTJc657yN8I4z7whQ	Food Coffee & Tea OutdoorSeating WiFi_u'free' ...

3520 rows × 2 columns

## Implementation:

- Create a dataset with columns business names and corresponding combined features of businesses
- Clustered businesses using K - means algorithm into 50 classes based on TF-IDF Vectorization on textual information created before.
- Similar businesses were clustered together.
- For Business user, based on his restaurant preferences (restaurants rated by the user) we selected the cluster that best represents the user
- Used these clusters to generate predicted ratings based on collaborative filtering.
- Got better performance.

## Evaluation:

Hybrid model Based	
RMSE	3.654
MAE	3.601

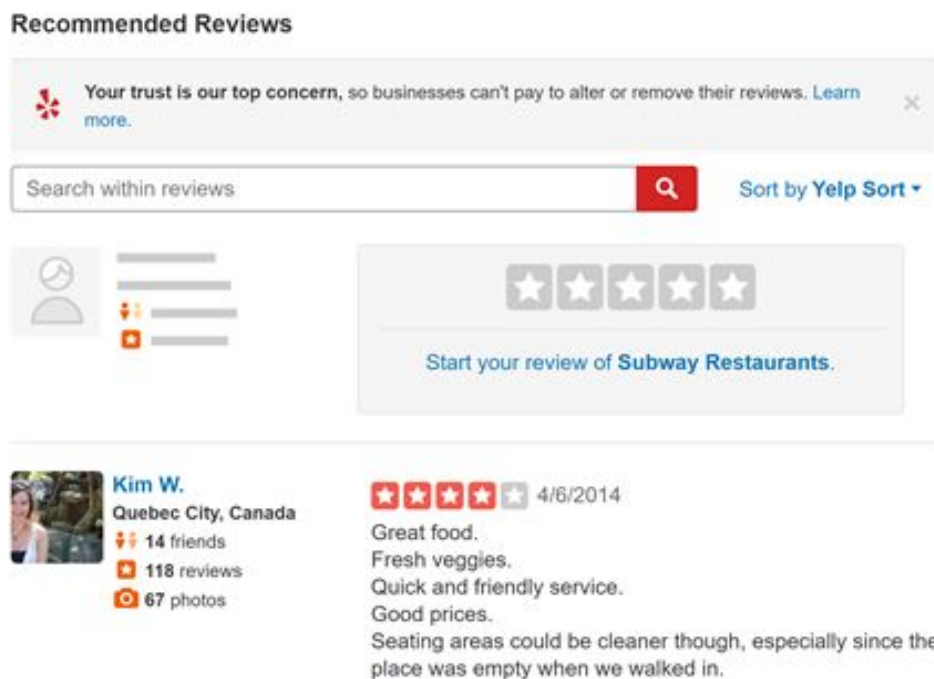
## Task 2 : Keyword Extraction

### Problem Statement:

- Extract relevant key phrases from reviews of businesses
- Indexation and Retrieval of Business Reviews based on selected tags.

### Yelp current system:

As per current Yelp system, a User cannot view popular review phrases and use it to extract reviews of his choice. He only has option to search tags and retrieve reviews. User himself has to come up with search query.



### Significance of Our Problem Statement:

Users may want to read reviews related to specific features/tags, say, some users may want to read reviews that contain information about particular dishes in restaurants, or the service, or prices. Our aim is to extract relevant and important features associated with businesses from the business reviews given by users. Using these tags, users can identify important features of a business without having any prior knowledge of the business and also filter reviews to make an informed decision.

## Task 2.1 Extracting relevant keyword phrase from review data

### Exploratory data analysis and Data Preprocessing:

We want to generate the popular review tags of businesses. For this purpose, we would need to test our model on Restaurants with at least 50 reviews. We continue to work with filtered data from task 1 and select a small subset of 15 businesses. We used below filters on review data:

- Restaurants of Nevada State
- We selected 15 Businesses with >50 reviews

We are interested in the popular words and phrases. We performed the following data preprocessing functions:

- Converting string into lower case
- Lemmatized words
- Removed stop words
- Removed special characters and numbers

### Implementation:

We extensively researched and found out few algorithms that were useful to extract relevant keywords while others were useful in extracting keyphrases from the text. The text is formed by merging all the reviews for a business and preprocessing the text to remove irrelevant information.

We compared the results of 3 different algorithms such as:

1. Rapid Automatic Keyword Extraction (RAKE)
2. Python Keyphrase Extraction (PKE)
3. Our Approach

#### 1. Rapid Automatic Keyword Extraction (RAKE)

RAKE short for Rapid Automatic Keyword Extraction algorithm, is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text.

#### 2. Python Keyphrase Extraction (PKE)

PKE is an open source python-based keyphrase extraction toolkit. It provides an end-to-end keyphrase extraction pipeline in which each component can be easily modified or extended

to develop new models. pke also allows for easy benchmarking of state-of-the-art keyphrase extraction models, and ships with supervised models trained on the SemEval-2010 dataset.

### 3. Our Approach

Since we did not have data with labels, we couldn't use any supervised algorithm to train our model. Hence we came up with an unsupervised method where we used PageRank along with N-grams to extract relevant keyphrases from the text.

We only consider unigrams, bigrams and trigrams as phrases longer than 3 words may contain irrelevant information. After this we construct a graph using networkx package. The nodes of the graph are phrases instead of just words. The nodes of the graph are then scored using the default PageRank algorithm using the given formula:

$$PR(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|}$$

Score of each vertex in the graph represents the "importance" or "power" of that vertex within the graph. Once the score of all the edges in the graph is computed, the score is sorted in descending order to extract top relevant keyphrases.

### Results:

After going through the keyphrases we realized that unigrams have a high score because that unigram is also a part of different bigrams and trigrams which leads to the unigrams having a high score. Additionally, in majority of the cases unigrams fail to convey the gist of the text and are not useful to understand the context of the review. This being the reason, we remove unigrams from the extracted keyphrases.

Below is an output for one of the businesses, where the first row is one of the reviews for that business and the second row is a list of keyphrases extracted for that business.

"I am a huge fan of Little Caesars in general. I like the Detroit style pizza, I love the prices, and I prefer to pickup my pizza myself without paying a tip to the delivery driver. The Hot-N-Ready idea is great. No need to wait because they will stockpile a bunch of cheese and pepperoni pizzas in anticipation of the peak hour. The location that is closest to me (Rhodes Ranch Town Center) doesn't understand the concept and guaranteed, you will be waiting 10min+ so I have stopped going there and will only come here now even though it is a few miles farther away. Occasionally you still have to wait here, but they make up for it by ALWAYS having excellent customer service and ALWAYS having the cookies for the kids. I have never had any problem with the quality of pizza at this location."

['pizza location', 'pizza time', 'good pizza', 'pizza ready', 'get pizza', 'pepperoni pizza', 'cheese pizza', 'pizza cheap', 'wait pizza', 'wrong pizza', 'pizza deal', 'crust pizza', 'pizza delicious', 'pizza home', 'pizza deep', 'bad pizza', 'pizza hot ready fresh pizza little', 'get pizza time price', 'cheese pepperoni pizza', 'cold pizza little', 'pizza pepperoni hot', 'large cheese pizza', 'pizza town delicious', 'little caesars pizza good', 'order ready', 'u fresh hot pizza', 'food service', 'good food', 'food ready', 'pepperoni order', 'get food', 'friendly staff large pizza', 'large pizza bad deal', 'order quick', 'little time', 'order deep', 'place order little time', 'min order', 'call order', 'location customer service', 'service get', 'food hot ready', 'wait food', 'location staff', 'decent food', 'food delicious', 'location star', 'location busy', 'time hot', 'wrong time', 'cheap food decent cheap food', 'customer service', 'change time', 'time price cant', 'food cheap delicious', 'location clean staff', 'get im', 'get coupon', 'bread ready', 'hot ready', 'good little caesars', 'time ready min wait fresh', 'people get', 'cheese bread good', 'food home cold', 'phone get', 'decent customer service', 'good price cant', 'get busy', 'bad customer service', 'delicious get', 'nice service quick', 'hot ready deal', 'cheese pepperoni', 'staff u get', 'little caesars', 'price phone', 'cheese bread', 'price decent', 'u customer', 'cheese pepperoni call', 'decent quick', 'little caesars nice clean', 'cheese cold', 'large cheese', 'quick meal', 'hot fresh', 'coupon free', 'fresh u', 'cheap meal', 'clean people friendly', 'friendly people', 'minute fresh', 'place clean staff', 'friendly staff', 'nice wrong', 'girl phone', 'decent cant']

Here we can notice that we get a very good match between our generated phrase and the review. From the above keyphrases, we have removed keywords like 'pizza' as it doesn't give much information about the type of pizza and also what was the experience with that pizza. We can see generated tags like 'good pizza', 'cheap pizza', 'cheese pizza', etc. These tags are much more informative and appealing to the user and have a higher probability of being used by the user to read relevant reviews.

## Evaluation:

The evaluation technique used for keyword generation is **ROUGE**.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is essentially a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summaries (manually generated tags). ROUGE is a good evaluation metric for our system as the tags generated by us and the algorithms might not exactly match, so ROUGE takes that into consideration and calculate the values on the basis of text similarity.

The performance of our algorithms is evaluated on the basis of Precision and Recall values. The formula for Precision and Recall is given by:

$$Precision = \frac{\text{No of overlapping words}}{\text{Total words in True Labels}}$$

$$Recall = \frac{\text{No of overlapping words}}{\text{Total words in Model Generated Tags}}$$

We got below results compared with the baseline models PKE and RAKE:

	precision	recall	f1
algorithms			
RAKE	0.904193	0.140130	0.241220
PKE	0.979417	0.055337	0.104462
OUR APPROACH	0.955263	0.239836	0.381079

## Task 2.2 Indexation and Retrieval of relevant Reviews:

The motivation behind this task, is to retrieve the relevant reviews of the business when the user selects the most popular keyphrase tags.

### Indexation :

Since there is a lot of review data for the businesses, we had to index the reviews for retrieval purpose. The indexation was done using Python API called Whoosh.

We index on below fields:

**Business\_id | User\_id | Review | Business\_name | Date | Username | Review\_id**

Below indexes were created:

<input type="checkbox"/> Name	Date modified	Type	Size
<input type="checkbox"/> _MAIN_1.toc	12/8/2019 6:59 PM	TOC File	3 KB
<input type="checkbox"/> MAIN_qphpwbudze9rojc5.seg	12/8/2019 6:59 PM	SEG File	2,140 KB
<input type="checkbox"/> MAIN_WRITELOCK	12/8/2019 6:58 PM	File	0 KB

### Retrieval:

For retrieval of reviews on selecting a tag, we Implemented the most efficient BM25F scoring algorithm using Whoosh Searcher.

- BM25 stands for Best Match 25 algorithm.



- The idf calculation of BM25 is similar to Classic similarity.
- BM25 - Bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document BM25's IDF has the potential for giving negative scores for terms with very high document frequency.
- BM25 has its roots in probabilistic information retrieval.
- Basically, it casts relevance as a probability problem. A relevance score, according to probabilistic information retrieval, ought to reflect the probability a user will consider the result relevant.
- For short as well as long queries, BM25 works as the best model.

We then used Whoosh QueryParser to parse Reviews.

Based on the scores calculated, retrieved top Search results.

## Prototype:

To demonstrate the above, we created a small prototype using Python, Flask and Bootstrap. The prototype looks as below:

**Yelp Review Retrieval**

Business:

Select a Business ▼

Submit

Reviews for Selected Business: Little Caesar's

pizza pepperoni hot bad pizza change time good price cant customer service place cheese pepperoni pizza good

pizza crust ordered deep hot ready deal clean people friendly

In this, we have displayed a dropdown for Business and when a business is selected all the popular tags related to that Business would be displayed in the blue color. When the tag is clicked, we retrieve all the relevant reviews with that keyphrase tag. So the user can read all the reviews that he is interested in.

## Evaluation

Since it was difficult to evaluate the results on such a large dataset, we thought of testing our model on the real Yelp website. One such example for

**Business = "Little Caesar's" | Tag Chosen = "cheese pepperoni"**

Reviews for Selected Business: Little Caesar's

[pizza pepperoni hot](#)
[bad pizza](#)
[change time](#)
[good price cant](#)
[customer service place](#)
[cheese pepperoni](#)
[pizza good](#)

[pizza crust ordered deep](#)
[hot ready deal](#)
[clean people friendly](#)

**Melissa | 2017-04-03 04:20:08**


For a cheap on the go food they have something always ready. I'm also very pleased to see that they have deep dish pizza **pepperoni** on the hot n ready option.

**Tyler | 2017-01-30 04:55:51**

I am a huge fan of Little Caesars in general. I like the Detroit style pizza, I love the prices, and I prefer to pickup my pizza myself without paying a tip to the delivery driver. The Hot-N-Ready idea is great. No need to wait because they will stockpile a bunch of cheese and **pepperoni** pizzas in anticipation of the peak hour. The location that is closest to me (Rhodes Ranch Town Center) doesn't understand the concept and guaranteed, you will be waiting 10min+ so I have have stopped going there and will only come here now even though it is a few miles farther away. Occasionally you still have to wait here, but they make up for it by ALWAYS having excellent customer service and ALWAYS having the cookies for the kids. I have never had any problem with the quality of pizza at this location.

Actual Yelp website:

← → ↺ ↻ 🔒 yelp.com/biz/little-caesars-las-vegas-11?osq=Little%20Caesars&q=pepperoni%20pizza ☆ 📱 📄



**Tyler S.**  
Las Vegas, NV  
👤 93 friends  
📝 19 reviews

★★★★★ 1/29/2017


I am a huge fan of Little Caesars in general. I like the Detroit style pizza, I love the prices, and I prefer to pickup my pizza myself without paying a tip to the delivery driver.

The Hot-N-Ready idea is great. No need to wait because they will stockpile a bunch of cheese and pepperoni pizzas in anticipation of the peak hour.

The location that is closest to me (Rhodes Ranch Town Center) doesn't understand the concept and guaranteed, you will be waiting 10min+ so I have have stopped going there and will only come here now even though it is a few miles farther away.

Occasionally you still have to wait here, but they make up for it by ALWAYS having excellent customer service and ALWAYS having the cookies for the kids. I have never had any problem with the quality of pizza at this location.

👍 Useful 🤔 Funny 🍷 Cool



**Melissa G.**  
Pasadena, CA  
👤 129 friends  
📝 124 reviews

★★★★☆ 4/2/2017

📍 1 check-in

For a cheap on the go food they have something always ready. I'm also very pleased to see that they have deep dish pizza pepperoni on the hot n ready option.


👍 Useful 🤔 Funny 🍷 Cool

[littlecaesars.com](#)

📞 (702) 240-6252

[Get Directions](#)  
3545 S Fort Apache Rd  
Unit 115  
Las Vegas, NV 89147

🍴 [Full menu](#)



**Is this your business?**

Claim your business to immediately update business information, respond to reviews, and more!

[Claim This Business](#)

## Conclusion

For task one, According to the analysis of the models, the best model we recommend for predicting the rating and making recommendation is the collaborative filtering by SVD model. This model provides significantly lower training and testing RMSE, and it appears to be the most promising algorithm for all datasets when optimizing for performance and accuracy. When the model is clubbed and hybrid model is used, we got even better results and accuracy.

For task two, Our approach implemented for keyphrase generation performed much better than the baseline model. Since we did not have data with labels, we couldn't use any supervised algorithm to train our model. Hence we came up with an unsupervised method where we used PageRank along with N-grams to extract relevant keyphrases from the text. During implementation we gave more weightage to bigrams and trigrams and excluded unigrams as they had high score but not the gist of the review and excluded more than 4 word keyphrase as they had a lot of redundant information.

## Future Scope and Improvement

Although the models have shown a fairly good performance, we are still seeking more advanced models which can predict in a more accurate way. For example, we are thinking about different ways a hybrid model could take advantage of all the models we built. As for the data, if we could get the real time location information of the user then employing location information in making prediction would make the recommendation system much more intelligent. We could also extend our scope to other businesses and more states.

## Work Distribution

Arpit : Implement Task2 Keyphrase extraction algorithm. Implemented other baseline models like RAKE and PKE. Worked on Evaluation of the above algorithms. Worked on data analysis of business reviews data.

Neha : Implemented indexation of reviews data using Whoosh. Worked on Search and retrieval of relevant reviews using BM25. Built prototype using Python Flask. Evaluated the retrieved results with real Yelp data. Implemented hybrid approach. Analysis of evaluation techniques for Keyphrase extraction

Nikita : Implemented Task1 Exploratory data analysis. Implemented collaborative filtering different approaches using memory-based and model-based. Worked on Evaluation of Task1.

