ILS-Z534 SEARCH FINAL PROJECT REPORT
Yelp Dataset Challenge 2017

Bhargavi Chalasani, Sowmya Ravi, Priyanka Cherukuri, Bhargava Vadlamani

ABSTRACT

Recommendation systems help users quickly sift through available information to find the most interesting and valuable information for them. The first task in project aims to recommend business to users using two approaches content based recommendations and collaborative filtering based recommendation approach. The second task aims to understand business features from review text. Yelp dataset is used for both the tasks. For the first task, the model is evaluated across the following metrics: coverage, % of new businesses recommended and % of old businesses recommended. It was observed that the Item based Top N Recommender had a better coverage ~2 % than ALS for task one and the percentage of new and old businesses recommended are almost the same.

Index Terms — *collaborative filtering, item-based recommendation, matrix factorization, topic modeling.*

1. INTRODUCTION

In everyday life, people rely on recommendations from other people through reference letters, news media, travel guides etc. in decision making. Recommender systems assist and augment this social process by quickly sifting through available resources to find the information that is most valuable and interesting to the users.

A recommender system is a subclass of information filtering system that aims to predict the rating or preference that a user would give to an item. Recommender systems have become increasingly popular and are widely used in movies, music, news, books, research articles, search queries, social tags and products in general. Recommender systems typically produce recommendations lists in two ways – through collaborative filtering or through content-based filtering.

Collaborative filtering approach builds a model from user's past behavior such as items previously purchased or past numerical ratings given to those items and makes predictions on items or ratings for items that the user may have interest in. Content-based filtering approach uses a series of discrete characteristics of an item to recommend additional items with similar properties. These approaches are often combined in Hybrid recommender systems.

2. COLLABORATIVE FILTERING

Collaborative filtering is one of the most successful approaches to building recommender systems that uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences of users. It is based on the fundamental assumption that if users X and Y rate n items similarly or have similar behaviors (buying, watching, listening), then they will rate or act on other items similarly.

If there is a list of m users $\{u_1, u_1, \ldots u_m\}$ and a list of n items $\{i_1, i_2, \ldots\ldots i_n\}$ and each user has a list of items that he explicitly rated or about which the user's preferences have been inferred through implicit behavior such as click-throughs. The list of users and items rated by them can be converted into a user-item ratings matrix and recommendations can be made for another active user in the system.

2.1. Collaborative Filtering Techniques

2.1.1. Memory based Collaborative Filtering
Memory based CF algorithms generate predictions by considering the entire or a sample of the user-item database. Every user is a part of a group of people with similar interests. By identifying the neighbors (users with similar interests) of a new or active user, the algorithm generates a prediction of preferences on new items for that user.

The neighborhood based CF algorithm calculates the similarity weight, w_{ij} reflecting the distance correlation or weight between two users or items i and j; a weighted average of all the ratings of the user or item on a certain item or user is taken to generate a prediction for the active user. For the task involving generating top N recommendations, involves finding k most similar users or items (nearest neighbors) after computing similarities and aggregating the neighbors to get the top-N most frequent items as the recommendation.

For item-based CF algorithm, similarity computation between two items i and j involves identifying users who rated both these items and applying a similarity computation to determine the similarity w_{ij} between the two co-rated items of the users. For user-based CF algorithm, the similarity $w_{u,v}$ between the users u and v who both rated the same items is computed.

Similarity or weight between users or items is computed using different methods.

2.1.1.1 Correlation based Similarity

The similarity $w_{u,v}$ between the users u and v or w_{ij} between the two co-rated items i and j is measured by computing the Pearson correlation or other correlation based similarities.

Pearson correlation measures the extent to which two variables linearly relate to each other. For user based algorithm, the Pearson correlation between users u and v is

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}}, \qquad (1)$$

For item based algorithm, here U denotes the set of users who rated both items i and j

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}}, \qquad (2)$$

where $r_{u,i}$ is the rating of user u on item i.

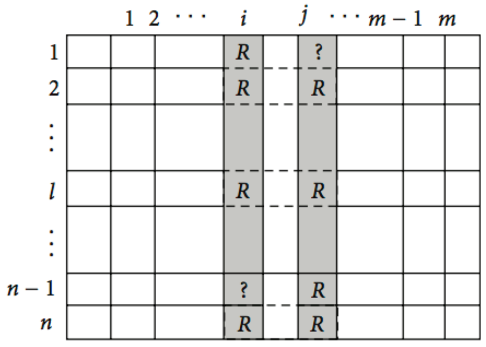

Figure 1: Item based similarity calculation on the rated items

2.1.2. Model based Collaborative Filtering

Memory based recommendations are not always as fast as they should work and require large amount of memory. The model based recommendation system extracts the information from the model and uses that to predict the user ratings. The main advantage of using this method is speed and scalability

There are many model-based collaborative filtering techniques such as Bayesian networks, clustering methods, latent semantic models and Markov decision processes. One potential issue that this method has to deal with is the highly sparse data. Dimensionality reduction techniques such as PCA can tackle this problem. The reduces matrix could be used for user-user or item-item recommendations.

2.1.3. Item based top n recommender

User based Collaborative Filtering based approaches have been quite successful. However, the major problems associated with them are sparsity and scalability. Also having good quality historical data is necessary for the algorithm to work well and this is a costly process that may organizations cannot afford. Hence this may lead to poor recommendations. Also, the coverage, which is a measure of diversity of items, may not be good for the CF based recommendation systems.

An alternate approach is to build items based recommendation models. Item based top n recommender is an approach similar to the content based recommender. It constructs a feature array for items being recommended. For each business that the user has rated, the k most similar items are found out. Then, for every user, a unique set of recommended businesses is created and the businesses that the user already rated is removed from that set. For each business in this set, a new ranking metric is calculated as the sum of distances from the user rated business is found out. This ranking metric is used to rank the list of recommended businesses for each user and the top n businesses are shown.

The K Nearest Neighbors algorithm is used to kind the k most similar businesses. Different distance metrics such as cosine similarity, Euclidean distance, Manhattan distance, Pearson's correlation etc. can be used as the similarity metric. Depending upon this the recommended businesses will vary.

2.4. Evaluation metrics

To assess how effectively the information retrieval system meets the information needs of users, traditional evaluation metrics such as precision and recall are used. The evaluation performance measurement generally considers a collection of documents to be searched and the search query.

Precision, recall, and the F measure are set-based measures computed using unordered sets of documents. MAP (Mean Average Precision) is used in evaluation ranked retrieval results.

2.4.1. Precision

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

2.4.2. Recall

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

2.4.3. Mean Average Precision

Precision and recall are single valued metrics based on the whole collection of records returned by the system. A precision-recall curve (p(r) versus r) can be plotted by computing precision and recall at every position in a ranked sequence of documents. Average precision computes the average value of p(r) over the interval from r=0 to r=1.

$$\text{AveP} = \int_0^1 p(r)dr$$

Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

where Q = number of queries.

2.4.4. F measure

F-measure or balanced F-score is the weighted harmonic mean of precision and recall. Precision and recall are evenly weighted, so it is also known as F_1 measure.

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{(\text{precision} + \text{recall})}$$

2.4.5. Normalized Discounted Cumulative Gain (NDCG)

Normalized discounted cumulative gain (NDCG) measures the performance of a recommendation system based on the graded relevance of the recommended results. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the results.

2.4.6. Root Mean Square Error (RMSE)

RMSE represents the sample standard deviation of the differences between predicted values and observed values and is the most popular metric in evaluating the accuracy of predicted results.

3. TASK 1 - RECOMMENDING BUSINESS TO USERS

3.1. Exploratory Data Analysis

Yelp dataset: https://www.yelp.com/dataset/challenge

156639 Businesses
1183362 Users
4736897 Reviews

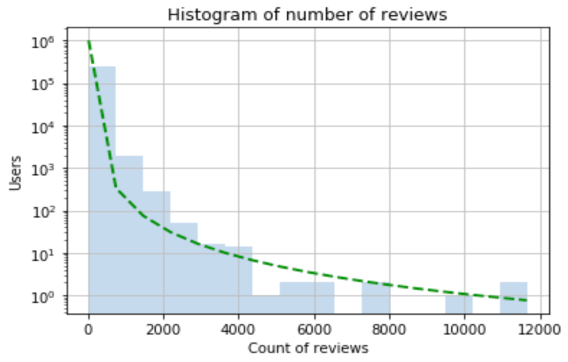

Figure 2: Histogram of users vs. review count

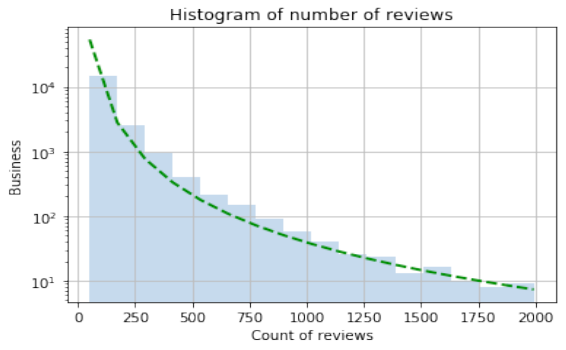

Figure 3: Histogram of businesses vs. review count

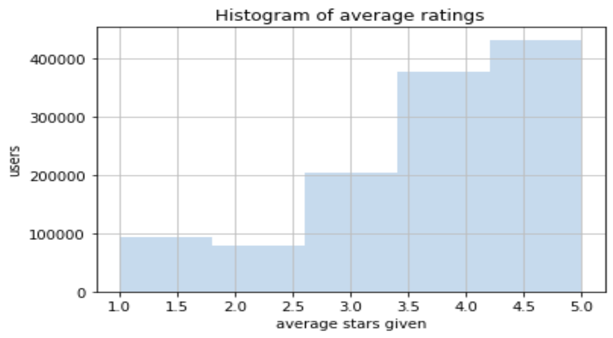

Figure 4: Histogram of users vs. average stars given

From the exploratory data analysis, it is observed that there is a reasonable count of reviews given by users for businesses that would help us in our analysis for Task 1. The histograms visualize the distribution of review count for users in Yelp dataset who have given at least 50 reviews. The distribution for average star ratings given by users is also illustrated above.

3.2. Item based top N Recommendations

The first algorithm that we tried was the item based top N recommender. The algorithm was implemented in Python 2.7.13

3.2.1. Data Processing and Feature Engineering

The data was given in four different json files. First these json files were converted to CSV files for easily importing into python. The steps taken to arrive at the features are enumerated below

- A subset of all the users who reviewed at least 20 businesses were created.
- For each user, a list of businesses that he had already rated is created
- For each business in the inventory, the features are one hot encoded to create a Boolean array.

The algorithm of how the recommendations are arrived at for each user is given in the following section

3.2.2. Algorithm

The algorithm is based on the idea that a user is more likely to avail businesses that are similar or related to businesses that he has already availed.

- For each business, find the k most similar businesses
- For each user C is union of the k most similar businesses for each business j ϵ U excluding the businesses which already exist in set U.
- Then, for each business c ϵ C, the similarity of that business to the set U is computed as the sum of similarities of all businesses in set U and c
- The set C finally contains a list of businesses sorted in decreasing order with respect to similarity and first N businesses are selected as the top N recommended set.

3.2.4. Evaluation Results

For each user 80% of the businesses he/she rated is taken for training and the remaining 20% is used for testing. Here the test data is also called the held-out data so that for each user the number of hits can be calculated. Since there is no prediction of rating that is output by the algorithm, recall was the metric used to evaluate the results.

Here recall is measured in the following way.

$$\frac{Number\ of\ hits\ in\ the\ \ test\ data}{N}$$

There is a hit in the test data if a business in the recommendations of the user also present in the held-out data. Average recall for the data was low around 0.45 (Averaged over multiple) when Euclidean distance was used.

In addition to this three other evaluation metrics were defined so that the performance of this algorithm could be compared to the other algorithm that was tried. The results of this was

- **Coverage: 2%**
- **% of New Businesses Recommended: 51%**
- **% of old Businesses Recommended: 49%**

3.2.3. Experiments

To arrive at the best configuration of business and the best distance metric, several experiments were conducted. The variation of recall with K is shown in Figure 5.

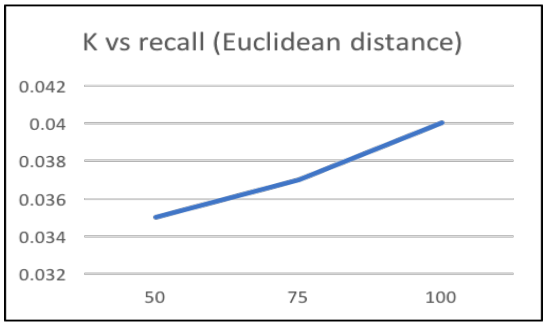

Figure 5: K vs recall

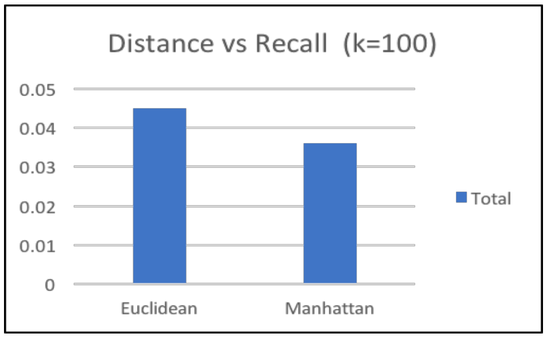

Figure 6: Distance Metric vs Recall

As the K value was increased from 50 to 100, the recall value also increased. But beyond 100 the recall did not increase much and the run times also became too high. Two different distance metrics were tried. One was the Euclidean and the other was the Manhattan distance. Euclidean distance performed better than Manhattan distance.

3.2.5. Challenges

- K Nearest neighbors is very computation heavy and takes a lot of time to train
- Unable to use the complete data set because of computations involved
- High dimensionality of the business features. PCA took a toll on the hit rate.
- In high dimensions the distance metrics would not work well.

3.3. ALS Collaborative Filtering

Alternating Least Squares is a collaborative filtering algorithm used to recommend businesses to users based on ratings of similar users. This algorithm enables companies to recommend their products based on choices and tastes of similar users.

Given the real world data often not being independent and there exists a joint correlation among the entities, it is not possible to definitively cluster(hard grouping) data points. When huge number of clusters are present and if joint interaction is taken a sparse matrix is generated which make computations and similarity estimations difficult. ALS provides a novel solution for this problem by attacking the problem using Matrix Factorization approach.

3.3.1. Algorithm

ALS uses matrix factorization approach in an alternating manner. ALS is essentially a regression algorithm trying to converge the value of R (Recommendation) matrix from matrices U and P (Users and Choices).This happens through a converging mechanism. If we fix U and regress R over P, (or vice versa to fix P and regress R vs U), it essentially boils down to regression problem. Since the objective is to minimize RMSE,ALS uses OLS(Ordinary least squares) approach which promises lower RMSE values.

The goal was to compute the Ratings matrix R $\approx X^T Y$. This can be broken down into an optimization problem where we try to find the optimal X and Y functions. As discussed above, we are trying to minimize the RMSE of the observed ratings and regularize accordingly.

ALS Tries to minimize the following loss function

$$\min_{X,Y} \sum_{r_{ui}\, observed} (r_{ui} - x_u^T y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

ALS is a two step iterative process:

1. Fixes P and solves for U
2. Fixes U and solves for P

This is guaranteed to converge only to the local minima. Since the actual cost function includes a regularization term, it is slightly longer. According to the two-step process, the cost function can be broken down into two cost functions:

$$\forall u_i : J(u_i) = ||R_i - u_i \times P^T||_2 + \lambda \cdot ||u_i||_2$$

$$\forall p_j : J(p_j) = ||R_i - U \times p_j^T||_2 + \lambda \cdot ||p_j||_2$$

3.3.2. PySpark Implementation

ALS involves time consuming and computations and it runs for several iterations before it converges. To speed up computations this algorithm was implemented in Spark. PySpark was used for coding and the ALS implementation in MLLib package was used. In addition using Spark enabled us to use a large dataset to train the model.

The working of the algorithm in spark is detailed below.

Ratings*: RDD((u, i, r_{ui}), . . .)*

X : RDD(x1, . . . , xn)

Predictions*: Y : RDD(y1, . . . , ym)*

We know that the equation of ALS is

$$x_u = (\underbrace{\sum_{r_{ui}\in r_{u*}} y_i y_i^T + \lambda I_k}_{A})^{-1} \underbrace{\sum_{r_{ui}\in r_{u*}} r_{ui} y_i}_{B}$$

Spark computes the A and B parts in the following manner,

To compute parts A and B, we can follow the steps below:

1. Join Ratings with Y factors using key i (items)
2. Map to compute $y_i y_i^T$ and change key to u (user)
3. Reduce By Key u (user) to compute $\sum y_i * y^T{}_i$

4. Invert
5. Another Reduce By Key u(user) to compute $\sum r_{ui} * y_{i_i}$

3.3.4. Evaluation Results

Given the range of predictions (R matrix) between -80 and 70, the RMSE is 5.43. A few other evaluation metrics

- **Coverage: 1.4%**
- **% of New Businesses Recommended: 36%**
- **% of old Businesses Recommended: 64%**

3.3.3. Challenges

The major challenges faced while doing als are listed below

- The high sparsity of data was a challenge
- The run time was high and this was dealt with by running the algorithm on Spark

3.4. Evaluation of Top N Recommender and ALS Collaborative Filtering

Both the recommenders were evaluated using three following criteria.

1. **Recall**: The number of relevant recommendations out of all the recommendations made
2. **Coverage:** The % of the all the businesses that are covered in the recommendations. This measures how diverse the recommendations are.
3. **% of New Businesses Recommended:** The proportion for recommendations that are new businesses. New businesses are identified as ones that have less than or equal to 120 reviews.

4. **% of Old/ Established businesses recommended:** The proportion for recommendations that are new businesses. New businesses are identified as ones that have greater than 120 reviews.

	Recall	Coverage %	% of new Business Recommended	% of old Businesses Recommended
Item-based Recommender	4%	2%	51%	49%
ALS Collaborative Filtering	4.7%	1.4%	36%	64%

From the table we can see that the content based recommender has a better coverage. Also the % of new businesses and od businesses recommended are almost same for Item based recommender. However, ALS Collaborative Filtering technique is more biased towards recommending old businesses.

4. TASK 2 - PREDICTING THE CATEGORY/FEATURES OF A BUSINESS FROM REVIEW TEXT

4.1. Introduction

There are categories available for Yelp businesses which help point to the general market/operational domain of a business. This is helpful for grouping the businesses together from a data organization perspective and also helpful for Yelp to run complex analysis on these businesses to improve their user recommendation better. Given the importance of this feature, it is useful to be able to predict the category of a business when this data is not available from the business owner.

We attempt to predict these categories for a business by considering all the reviews given to a business by considering both supervised and unsupervised techniques.

4.2. Dataset

Using the location information provided for the businesses, we noticed that the data points available were concentrated around few cities in the United States and a couple of other countries. We considered data from Las Vegas for our prediction model since it included businesses ranging from Arts & Entertainment, Casinos, Restaurants and the other regular businesses found in every city. We considered the top 100 most common categories based on the number of businesses.

4.3. Methods

4.3.1. Unsupervised Learning

Since we are planning to predict the category of a business, we considered only the nouns present in the review text. We then performed Latent Dirichlet Allocation(LDA) over the noun tags of all reviews across all the categories to obtain the important topics obtained from the reviews.

4.3.1.1. Preprocessing

We considered the review text for businesses and removed the stopwords and punctuation from the text.
Based on the group of words associated with the topic, tried to understand the theme of the topic.

4.3.1.2. Modeling

For performing topic modeling we used the LDA algorithm from python genism library.

LDA is a generative probabilistic model of a corpus. It is based on the idea that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

LDA represents documents as mixtures of topics that spit out words with certain probabilities. It assumes that documents are produced in the following manner: when writing each document,

- Decide on the number of words N the document will have.
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics).
- Generate each word in the document by first picking a topic according to the multinomial distribution. Using the topic to generate the word itself, according to the topic's multinomial distribution

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

Since we were interested in obtaining the topics alone, we considered the nouns available in the review text for topic modeling. We performed this Parts of speech tagging using graphlab package.

Deciding on the number of topics to be generated from LDA proved to be challenging. Choosing fewer topics led to a drop of few important expected themes and choosing too many topics led to similar topics being generated. We iterated over multiple combinations of the parameters num_topics, chunksize, update_every and passes to decide the final number of topics.

Apart from the expected category topics, we also observed few latent topics which are usually discussed in a business review like service, ambience, location etc. These can be very important to understand the sentiment around the features of these business

Category topics
cafe – “starbucks”, “coffee”, "drink", "location", "cup", "tea", "order", "service", "time"

mexican - "tacos’, "food", "place", "taco", “salsa”, “beans”, “chips", "meat", "burrito", "rice"'

car dealer/auto service
"car", "service", "vehicle", "cars", "experience", "auto", "customer", "wash", "dealership", "place"'

banking/finance
"bank", "credit", "card", "cash”, "account", “branch", “fee", "time", “charge", “chicago"'

vet
"staff", “office", "dog", "care", "time", "pain", "doctor", "years", "vet", "dogs"'

nail salon
"nails", "nail", "salon", "time", “gel", "pedicure", "place", "job", “color", "service"'

Latent topics
service/experience
food", “server”, “service", “table”, “place", “water", “experience", “drinks", “restaurant”, “time"'

ambience
"detail", "place", "environment", "mexico", "attention", "vacation", "light", "treat", "back", “time"'

location/service
"store", "location", "staff", "service", "area", "place", "stores", "seating", "customer", "convenience"'

4.3.1.3. Results
While considering the concatenated text of all the reviews of a business we received obtained an accuracy of 33.29% in predicting the category of a business.

Obtaining the topics of the individual reviews of a business and getting the top topics being discussed among these gave an accuracy of 51.39% in predicting the category of a business.

4.3.1.4. Challenges
The main challenge was to infer a name for these topics which was given in bold above. These topics can often be vague and trying to exactly match them to the category names to evaluate the performance. Considering the performance of LDA there was clearly room for improvement and we started exploring supervised techniques to solve the same problem.

4.3.2. Supervised Learning
To overcome the above restrictions, we decided to model the same problem using a supervised approach. We considered the business categories as training labels and implemented an 80-20 train test split to check the performance of our methods.

We considered the following algorithms Multinomial Naïve Bayes, Linear Support Vector Classification, and Stochastic Gradient Descent

4.3.2.1. Feature Engineering
We used Tf-Idf vectorizer to get the important features. Tf-Idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The Tf-Idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general (Wikipedia). The dataset was split into training and testing set in the ratio 80:20 respectively.

To implement multilabel classification we used sklearn.OneVsRestClassifier. This classifier compares each class to all other classes by considering them as a single unit. The results obtained from this classifier are very interpretable. The description of the machine learning models that we used is given below. All the models were used from scikit-learn package.

4.3.2.2. Feature Engineering
4.3.2.2.1. Multinomial Naïve Bayes
The multinomial naive Bayes model is typically used for discrete counts. E.g., if we have a text classification problem, we can take the idea of bernoulli trials one step further and instead of ‘word occurs in the document’ we have ‘count how often word occurs in the document’, you can think of it as "number of times outcome number x_i is observed over the n trials"

We used the sklearn.MultinomialNB for the prediction task with parameters like ‘alpha’ and ‘fit prior’.

4.3.2.2.2. Linear Support Vector Classifier
Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (Wikipedia)
Sklearn.LinearSVC was used for the prediction with parameters like ‘penalty’, ‘loss’ and ‘dual’.
4.3.2.2.3. Stochastic Gradient Descent Classifier

It is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions (Wikipedia).

Prediction was done using Sklearn.SGDClassifier with parameters like 'penalty', 'alpha' and 'iterations'

4.3.2.3. Parameter Tuning

Optimal parameters for each classifier are obtained using the GridSearchCV which is available in scikit learn package. Grid-search is a way to select the best of a family of models, parametrized by a grid of parameters. The best parameters that we obtain are used to classify the posts in the testing set.

4.3.2.4. Evaluation metric

We used f1 score as a success metric for our evaluation and to obtain best parameters over grid search. The F1-score, commonly used in information retrieval, measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives (TP) to all predicted positives (TP + FP). Recall is the ratio of true positives to all actual positives (TP + FN). The F1 score is given by:

F1 = 2(p. r) / (p+r)
Where Precision, p = TP/(TP+FP)
Recall, r = TP/(TP+FN)

The F1 metric weights recall and precision equally, and a good retrieval algorithm will maximize both precision and recall simultaneously. Thus, moderately good performance on both will be favored over extremely good performance on one and poor performance on the other.
We have used the weighted F1-score from sklearn metrics. Weighted F1-score makes sense for a data which has class imbalance.

Method	**Accuracy**	**F-Score**	**Average Precision**
SGDClassifier	83.69%	0.93	0.94
LinearSVC	76.93%	0.89	0.92
MultinomialNB	68.98%	0.87	0.89

4.3.2.4. Learnings and Future Direction

We observed that supervised techniques perform much better and can be preferred over unsupervised techniques when labeled data is available. But unsupervised can be used to be useful to understand other latent features of a business better like service, ambience, availability of delivery which can be used to populate the missing latent feature information for a yelp business.

5. TOOLSET

- Python 2.7.14
- Pandas
- Anaconda
- Jupyter notebook
- Apache Spark
- Google cloud platform
- Gensim library for topic modeling
- Django

12. REFERENCES

[1] Xiaoyuan Su, Taghi M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques".

[2] Recommender systems: https://en.wikipedia.org/wiki/Recommender_system

[3] LDA: http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/