Z534: Search

Project Report

Yelp Dataset Challenge

Report by,
Suhas Jagadish
Supreeth Suryaprakash
Raghuveer Krishnamurthy
Sanjana Agarwal

Contents

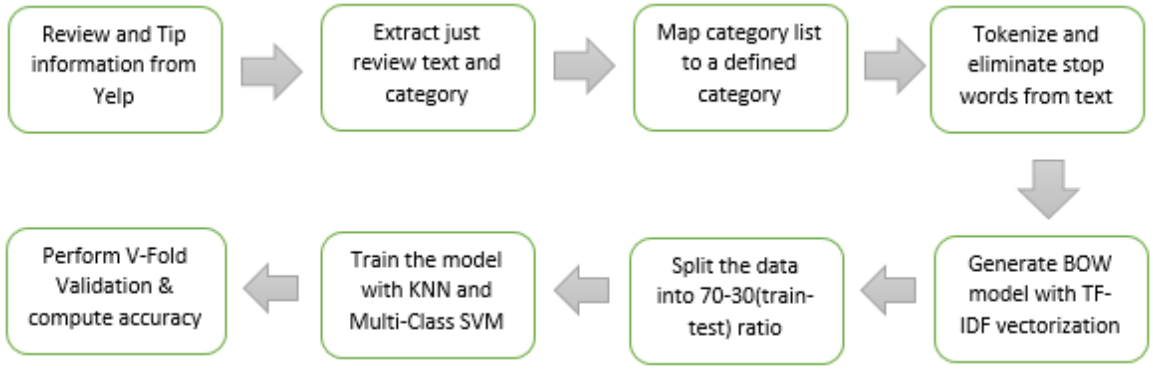
1 Task: Category prediction for Business

1.1 Research question

Each business in Yelp has a category associated with it. A business may belong to single or multiple categories. For example, ['Fast Food', 'Restaurants'] and ['Gluten-Free', 'Asian Fusion', 'Chinese', 'Restaurants'].

Yelp also has the review or tip information provided by the user. Our task is to predict the category of each business by just using the review or tip information.

1.2 Flow diagram



1.3 Text preprocessing

The data was in .json format and the first thing we did was to convert the json files to csv files. We then eliminated the unwanted information and extracted only the relevant data. For example, we extracted ‘review text’ and ‘business_id’ from reviews and ‘business_id’ and ‘category’ from business.

We then compared the business and review files to extract just the ‘Review/tip Text’ and ‘Category’ information because this is all we need to predict the business category based on review text.

When we extracted the business data, we found that the category information was a list of keywords, i.e., ['Fast Food', 'Restaurants'] or ['Gluten-Free', 'Asian Fusion', 'Chinese', 'Restaurants']. We found this a little inefficient to train the model. Hence we made a list of possible categories and a map of keywords to categories, so that we would get a defined category for each business.

For example, the list ['Gluten-Free', 'Asian Fusion', 'Chinese', 'Restaurants'] would be mapped to '**Chinese**' category or ['Burgers', 'Fast Food', 'Restaurants'] would be mapped to '**American**'.

Hence the final csv file contained 'review text' and corresponding 'defined category' information.

1.4 Method/Algorithm

Steps:

- Preprocess the data and extract relevant information using the above step.
- For text based feature extraction, we split each review, tokenize it, remove stop words and create a bag of words representation, which is used as features.
- TF-IDF values are computed for each word in a review across the bag of words and a model file is generated. The model file can be thought of as a 2D matrix where x-axis represents each review, y-axis represents the features (bag of words) and the value is the TF-IDF value of the word.
- Use the model file to train KNN and Multi-class SVM algorithms.
- Divide the dataset in the ratio 70-30 where 70% is used to train the model and 30% to test it.
- Use 5-fold cross validation to evaluate the results on the test set.
- Determine precision, recall and F1-score for each class label and finally compute the accuracy of our model.

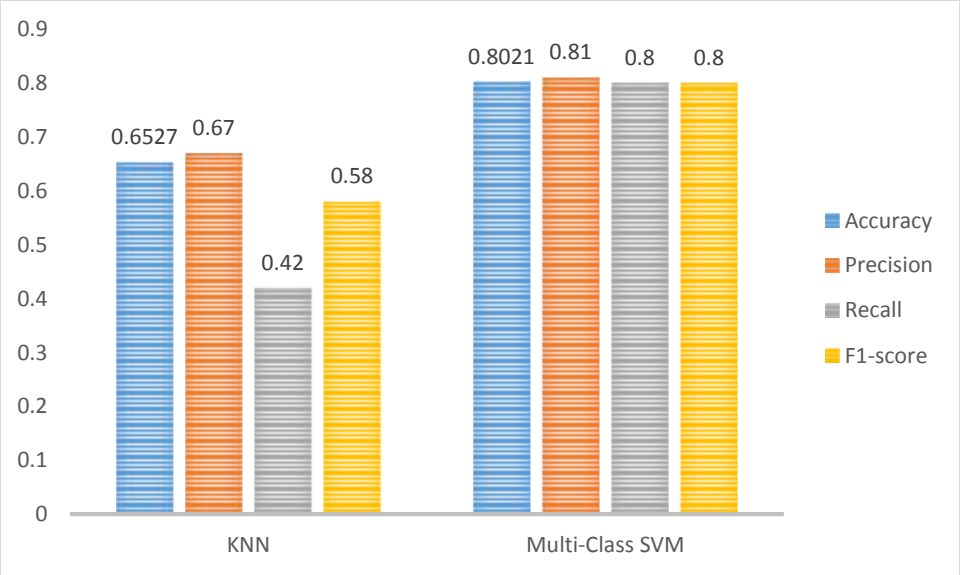
We implemented the BOW and TF-IDF calculations from scratch, without using any existing libraries. However KNN, SVM and Cross Validation functions were imported from python sklearn package [4] [5] [6].

1.5 Experiments and Evaluation

Considering the entire feature space:

- With KNN, we achieved an accuracy **65.27**% with the value of k set as 5.

- We experimented with different values of and found the highest accuracy rating with k=5. (k is the number of neighbors). The value of k is non-parametric and as [1] defines, general rule of thumb is choosing k is k=sqrt(N)/2 where N is the number of samples in the training set. However with this value of k, the observed accuracy was less than the one with k=5, hence we decided to choose the value as 5.
- With Multi-class SVM, we achieved an accuracy of **80.21**%. We decided to use the LinearSVC implementation of sklearn as this implements "one-vs-the-rest" multiclass strategy.
- The comparison between KNN and SVM is shown below –



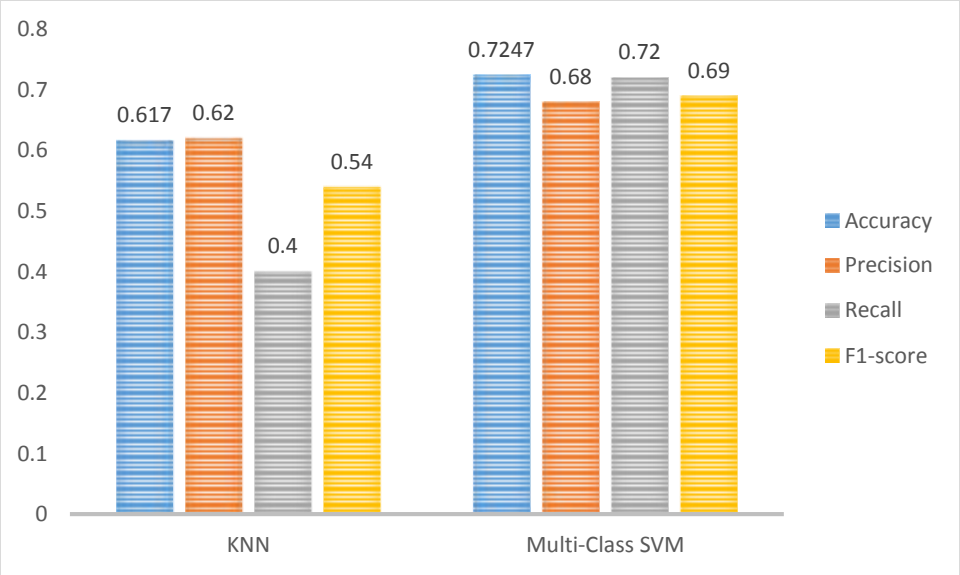
- As seen from the above results, linear kernelized SVM outperforms KNN for text classification. This is also evident from [2] where the authors prove that SVMs are well suited for text categorization, mainly because of its high dimensional feature spaces and sparse instance vectors. Furthermore, SVMs do not require any parameter tuning, since they can find good parameter settings automatically.

<u>After dimensionality reduction using Mutual Information (MI)</u>:

- As suggested by the Professor, we employed a feature selection or dimensionality reduction technique called **Mutual Information**, which

measures the dependency between the variables and reduce the number of features.

- One of the inputs for MI is to set the number of features, say k, beforehand. The MI algorithm will compute the top k features, eliminate the other features and compute the accuracies.
- We used 4 different values for k where k = 80%, 75%, 70% and 60% of the actual number of features, which is our baseline. We found the best accuracy when k was chosen to be 75% of the baseline features.
- The comparison between KNN and SVM after dimensionality reduction is shown below –



- As seen from the above results, the accuracies has decreased, both for KNN and SVM. After the dimensionality reduction, we actually lost some useful information about the review text. While we generated the bag of words model, we already removed stop words to generate the optimized feature space but with further reduction in feature space, it is evident that we lost some useful information about the review/tip text.

2 Task: Personalization of user's dashboard

2.1 Research question

Building a prediction system has been one of the state of the art problem. Powerful prediction system can bring more customers due to its reliability, especially for companies like Yelp whose business has great value for suggesting quality destinations to their customers.

Using the business, review and user file information, we can personalize a user's dashboard and can suggest what business they would like to review next, what neighborhood they prefer, what category of business they like or whom can they follow. Hence the research topic we are proposing is "Personalizing user's dashboard".

2.2 Find similar users to follow

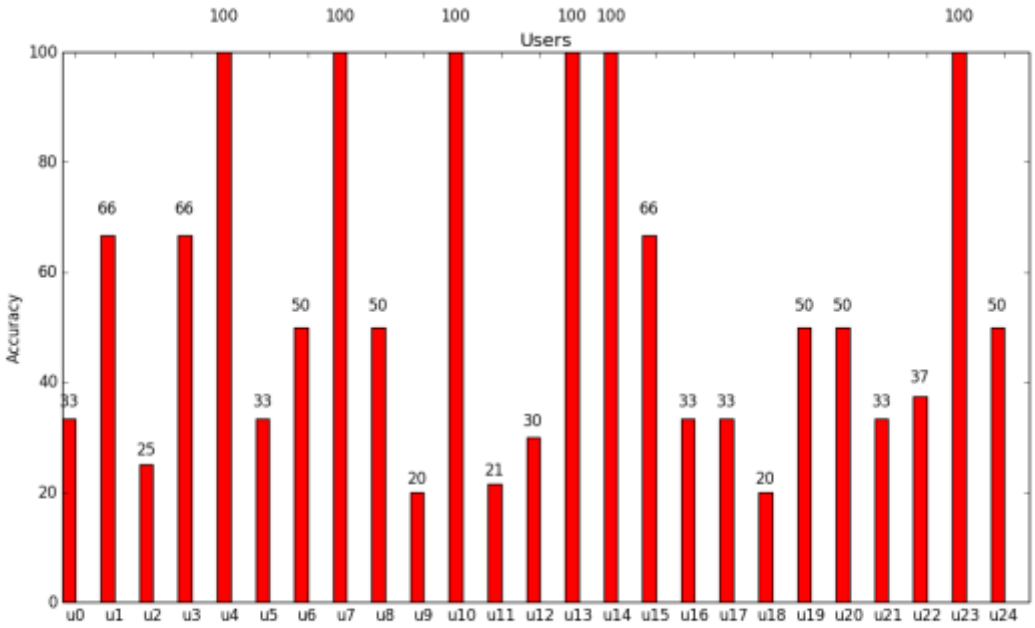
Method/Algorithm:

- The motive of this task is to predict set of followers for a user, who have higher correlation with them. We initially thought of using Pearson Correlation but later found out that we actually couldn't do it because of the random distribution of the data. We instead had to use Spearman correlation to rank top Users.
- Pearson correlation formula: https://statistics.laerd.com/statistical-guides/img/spearman-3.jpg
- Theoretically, Spearman correlation calculates the Pearson correlation for variables that are converted to ranks. Similar to Pearson's bivariate correlation, the Spearman correlation also tests the null hypothesis of independence between two variables [3] based on the rankings given by the Spearman correlation.
- We then take the top 100 possible followers for each user who have highest correlation.

Evaluation:

- Evaluation was pretty tough as this is a prediction problem. We removed the friends list from Yelp for each user and predicted a new set of friends using the above approach.

- Accuracy was calculated based on the friends list provided by Yelp. It's rather amusing that the top 100 probable followers we predicted gave an accuracy of **62.3**%.
- The below graph shows the accuracy for top 25 users. We calculated the same for top 100-200 users and got an overall accuracy of **56.7**%.



2.3 Predict user's favorite category

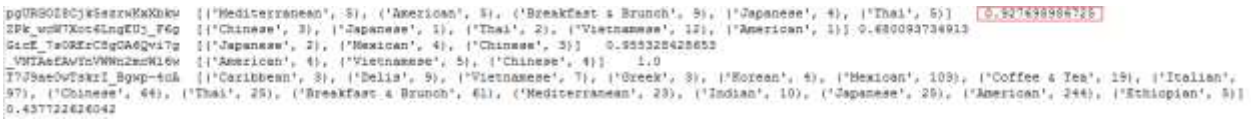
Method/Algorithm:

- The motive of this task is to predict a user's favorite category based on his/her reviews across different business.
- If the user has provided a rating greater than 3, we assume that the user likes to visit that particular business.
- We take the count of such business and the category with the highest count is chosen to be user's favorite category.

Evaluation:

- We use **Normalized discounted cumulative gain** (NDCG) [7] to measure the performance of our model. The value varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities.

- The evaluation result consists of userID, predicted category using the above method and the corresponding NDCG value –



- We calculated the mean NDCG value for the above model and it came up to **0.986218104425**.

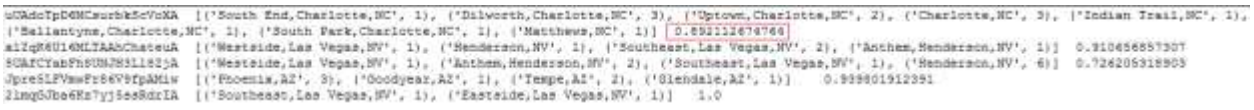
2.4 Predict user's favorite neighborhood

Method/Algorithm:

- The motive of this task is to predict a user's favorite neighborhood based on the number of visits across each neighborhood. The neighborhood information for each business is available in the business file.
- For each user review, we find out the neighborhood of the corresponding business and take a count of all the neighborhoods visited by the user.
- The neighborhood with the maximum count is chosen to be user's favorite neighborhood.

Evaluation:

- We once again use **Normalized discounted cumulative gain** (NDCG) to measure the performance of our model.
- The evaluation result consists of userID, predicted neighborhood using the above method and the corresponding NDCG value –

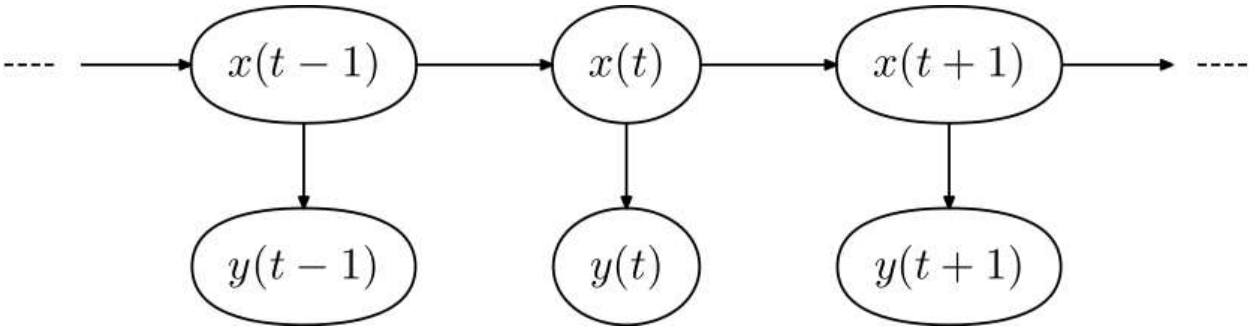


- We calculated the mean NDCG value for the above model and it came up to **0.987825871765**.

2.5 Predict Business using HMM

Method/Algorithm:

- The motive of this task is to model yelp data into a Hidden Markov Model for predicting the next category that the user may like.
- The input data for HMM is in the format "UserID, Review, Category, Date" where **UserID** is the unique ID for a user, **Review** has been converted into binary values where a rating above 3 is considered as good and a rating less than 3 is considered bad, **Category** is the business category and **Date** is the user check in date for a particular business [8].



- The HMM model consists of ***reviews as the observed state*** and the ***category as the hidden state***. We use forward algorithm to calculate the belief state at time t+1, given data until time t. The forward algorithm states:

  alpha_t(x_t) = p(y_t|x_t) sigma(p(x_t|x_t-1) alpha_t-1(x_t-1))

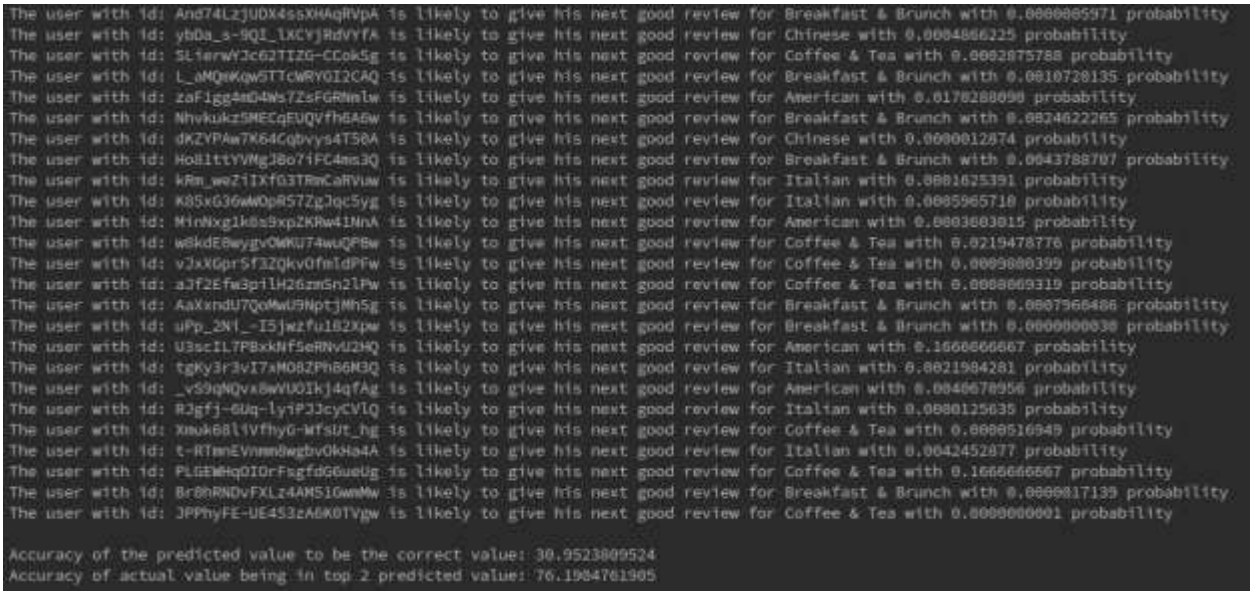
  where, alpha is the partial probability or the belief that a state x will be seen at time t, p(x_t|x_t-1) is the transition probability of state x from time t-1 to t, p(y_t|x_t) is the emission probability of seeing the output y given state x at time t and alpha_t-1(x_t-1) is the message from time t-1.
- Since we will not have transition probability for time = 0 (p(x_t|x_t-1)), we make use of a special condition to find alpha,

  alpha_t(x_0) = p(y_t|x_t) * initial_probability
- The ***initial vector pi_0*** is considered to have equal probability i.e., 1/number of different types of categories.

- We make use of MLE (maximum likelihood estimation) to predict the final value. Since we only have partial data and we want to predict the final data, we find out the distribution using the forward algorithm and plug the parameters to check the maximum probability. In our case we plug in the values of ***good/bad*** which are our states and estimate the maximum likelihood of observing the categories, which are our hidden states.

Evaluation:

- The final row for each user is stripped and passed to the generated model.
- Accuracy is measured as total number of correct predictions/ total number of records for each user.
- The results are displayed below –



- The accuracy is around **30%** and is due to the lack of data our model has seen. The MLE tends to perform better when it sees enough data to make the prediction. Hence with more number of data, the model will perform better.
- The initial vector that we chose assumes equal probability of all the categories. This can be tweaked for each user based on his most visited restaurants to achieve even better results.

3 Assumptions

- We normalized the category list to generate a defined category for each business. While doing so, we included only labels related to food activities. So any review, which is not related to food, is ignored in our computation.
- The review file was close to 2GB and we had trouble running experiments with it and hence we decided to use first 25K rows for all our calculations.
- While calculating HMM, if a user has only visited single restaurant, he will not have a transition probability. In such cases, we assume the transition probability to be a very low value (10^-7 in our case).
- For HMM, if a user has visited only a single catgeory and has provided a good review, then our system recommends same categories in future.

4 Conclusion

Task 1:

We used a supervised learning approach using KNN and Multi-class SVM to compute the accuracy for each category. Bag of words model coupled with TF-IDF vectorization was used to generate the model file. We saw that SVM performs better than KNN, even with dimensionality reduction and this proves that SVM is a better approach for text-based classification.

Task 2:

Personalizing user's dashboard can be very important to Yelp, which can be used to attract far many users. With our approach, user's typical dashboard would consist of –

- Favorite category
- Favorite neighborhood
- Next business to review
- Possible followers

Evaluating the performance of a predictive system can be challenging, without user feedback. Hence we tried multiple approaches, including NDCG to evaluate the accuracy of our model.

5 Future scope

- Task 1 can be further improved by considering all the available categories and reviews from Yelp, which will help to better train the model.
- We could index the data using Lucene, which will help to improve the overall runtime of the program with faster data lookup.
- In task 2, we predict user's favorite category and the next business he/she is likely to review. We can further enhance this functionality by recommending the top restaurants or in general top categories a user can visit.

6 How to run the code

Task 1:

- Run the ***jsontocsv.py*** file to convert the json text files to respective csv files.
- Then run ***viewcsv.py*** file to extract only relevant information. The file currently has code to extract business information, it can be altered to extract review information by just changing the columns to be extracted.
- Then run ***finalcsv.py*** file to extract the "review text -> defined category" information. This file will be saved as "final_csv.csv".
- Finally run ***Task1.py*** to compute the accuracies of task 1.
- All the code related to task 1 is in Python version **3.5.**

Task 2:

To Find Correlation between users

- From the command Terminal, python2.7 ***correlation.py***
- This file generates a model file named correlatedModelFile, further code makes use of this model file to provide the output.

To Find Neighborhood information

- From the command Terminal, python2.7 ***findNeighborhood.py***
- Before that run viewcsv.py to generate a CSV with Business IDs and categories. This file generates an intermediate model file businessFile.csv which will be used further in the program.

To Find Most Likeable Category

- From the command Terminal, python2.7 ***mostLikeableCategory.py***
- Before that run viewcsv.py to generate a CSV with Business IDs and categories. This file generates an intermediate model file businessFile.csv which will be used further in the program. It prints user's favorite Categories to a text file called userFavCuisines.txt

Hidden Markov Model

- Run the ***hmm_input.py*** to generate the input file for HMM. This program makes use of newBusiness.csv file which contains the defined category for each business('review_text, category'), hmm_extract.csv file which contains 'userID, businessID, stars, date' information.
- From the command Terminal, python2.7 ***hmm.py***

7 References

1. Markus Maier, Matthias Hein and Ulrike von Luxburg, "Optimal construction of k-nearest neighbor graphs for identifying noisy clusters"
2. Thorsten Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features"
3. http://www.statisticssolutions.com/spearman-rank-correlation/
4. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
5. http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
6. http://scikit-learn.org/stable/model_selection.html
7. Normalized Discounted Cumulative Gain: http://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf
8. https://en.wikipedia.org/wiki/Hidden_Markov_model

8 Team contribution

Task 1:

Name	Tasks
Suhas Jagadish	Text preprocessing
Suhas Jagadish and Sanjana Agarwal	Approach(BOW, TF-IDF, KNN, SVM)
Suhas Jagadish and Sanjana Agarwal	Evaluation(MI, V FOLD VALIDATION)

Task 2:

Name	Tasks
Raghuveer Krishnamurthy	Approach(HMM)
Supreeth Suryaprakash	Approach(Most liked Category, Neighborhood, potential followers)
Raghuveer Krishnamurthy and Supreeth Suryaprakash	Evaluation(NDCG and bootstrapping)

Github link: https://github.iu.edu/skeragod/search-project