CSE 158 Assignment 1 report

Kaggle name for visit prediction: ah, score: 0.82235

Kaggle name for categorization: alexhor, score: 0.59599

Task 1:

To predict the visit of given a user to a business, the model read in all those reviews from the train.json. Then store each of the business ID from each review to a data structure dict, in order to keep the information of how many unique business and the frequency of each business. There is other dict to do the same for user as well. Next, we have all the frequency of each business so we can sort those businesses from highest frequency to lowest, because we want to find out which businesses are popular. Then we will do the same for user as well to find out the most active user to less, they will be sorted by the number of reviews they wrote.

After that, we will let k % and J % of total visit and add the sorted users and businesses to two different list, say mostActive and mostPopular respectively, and sum up the frequency of each list until the total from each list is larger than the k% and j% of total review (or visit). Here k and j are adjustable because initially I just set it to be 50% for both, but I found out increasing both variables will gain more accuracy.

Finally, we can predict each pair of visit by using the pearson similarity, which takes the input of a user and business and see whether they have common category tags, here we assumed that user will visit a business only if they have been visited a similar one previously. If no tags in common, it will be predict false immediately, but if it is true, the model will check whether the business or user in that mostpoplar or mostactive list which have created previously. If either of the conditions is true, It will predict the user visit the business. In this model, the whole train.json is used to be a train set and the same set for validation. However, we tuned the model by adjusting k%, j% and the minimum score of pearson similarity to return true. By changing three parameters, it achieved higher accuracy from avoiding overfitting.

Task 2:

To predict what category of a given review from a user, I extracted the reviews that contain categoryID from train.json. Then split the extracted reviews to 50% for train and 50% for valid. Here we used one SVM in our model to classify 10 labels. The features are (the most popular k words is in and given review or not) among the reviews in train set. Here k is not specified because I set it as a variable to test whether increasing k will gain more accuracy. It turns out that it will increase the accuracy of valid set and test set. However, k has a limit, once it hit 1700 (which

means user the most 1700 popular words to identify the label) the accuracy starts to decrease and consumed much more time. To reduce the time consuming of training 10 binary SVMs for each category and for each category we have to test all five C values to gain higher validation, I chose C =1 for training SVM to classify 10 labels. The reason for choosing C =1 because 1 is in the middle of the given parameters so it can avoid underfit or overfit, moreover, it is small to run the svm much quicker.

Finally, after training the SVM, I tune the SVM by adjusting the number of features k rather than C to gain higher accuracy.