Pin-Hao Chen (phc2121)
Sharon Tsao (sjt2141)

Username and registered email address on the competition site:

- User Name: blooraspberry
- Email Address: sjt2141@columbia.edu

Describe your entity resolution technique, as well as its precision, recall, and F1 score.

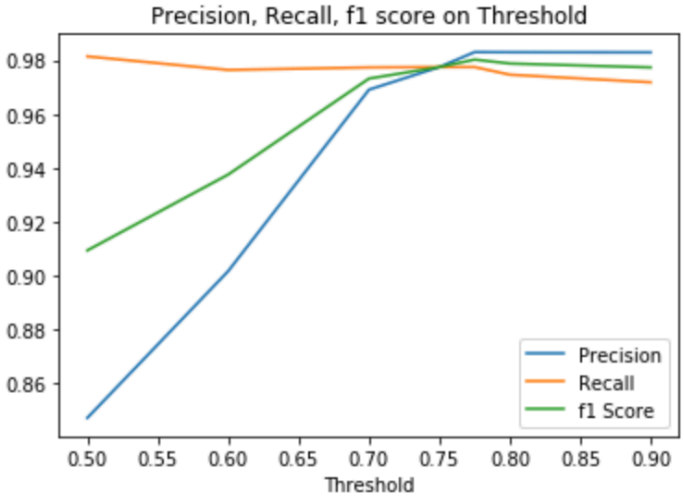
In calculating our precision, recall, and F1 score, we first preprocessed the data, created an entity resolution technique, then built the confusion matrix.

In preprocessing the data, we have standardized the forms of the data from Foursquare and Locu to prevent further confusion. We are working with only phone number, address, store name, and longitude/latitude as there are more complete data here. Moreover, any missing values are converted into NANs. Within preprocessing, phone numbers that exist are converted into integers, so any punctuation are removed. Next, all alphabets are set to lower case, and "east"/"west" are converted into "e."/"w.". We have also standardized all store names so that the first letter of each word is capitalized. Finally, we've moved "locality" into the full address for Foursquare and Locu. Within latitude and longitude, we're rounding them to 3 decimal places.

In our entity resolution technique, we have assigned a score for any matches in store name, phone number, street address, and longitude/latitude across Foursquare and Locu, and a threshold for considering each other as matches. In our example, any matches in phone number or address is given one point, longitude/latitude is given half point, and store name is assigned out of one point based on its correctness in the Levenshtein Distance formula. (We will explain how we placed the importance of each feature in the next question.) The one match that is placed with the highest score would be automatically assigned as a match between Foursquare and Locu, and each store from Foursquare and Locu would be removed from the current pool of possible matches. The algorithm finishes until all of the guesses that passes the threshold have been made.

After the model has been implemented, get_matches() searches through a range of possible threshold from 0.5 to 0.9 to help us decide the threshold which yields the most optimal F1 score. The score maximizes at 0.775, as shown below, which is what we've used to define our final threshold. The implementation of precision, recall, and f1 score is explained next.

Pin-Hao Chen (phc2121)
Sharon Tsao (sjt2141)



In order to define precision, recall, and F1 score, we're first going to clarify how we're calculating true positive (TP), true negative (TN), and false negative (FN). In TP, we make a guess and our guess from the entity resolution matches the real values. In FP, we make a guess but the guess is different from real values. When we make a guess, whether the guess is correct or not, that guess is removed from the pool of all possible guesses; we do this so we can calculate the FN, which is when we want to make a guess but it has already been guessed, thus, removed from the pool of possible guesses. With this logic, we calculate precision, which is TP/(TP + FP), and recall, which is TP/(TP + FN). F1 score is determined by the average ore precision and recall.

What were the most important features that powered your technique?

From implementing our model, we have also done a test run on assigning a match based on a match, once for each of the features. For example, we looked at an assignment based on a match in only address, then for each of the features. From our observation, address gave the best F1 score overall, while longitude/latitude had the lowest F1 score. Store name came as a close second. Hence, we have given a match in longitude/latitude a score of 0.5, and address a score of 1. While phone numbers didn't give the highest F1 score, we did think it placed an importance on matches, and that its lower F1 score may have stemmed from the amount of missing data. Store names is a strong indicator when the store names are similar, but there are many examples of stores that are spelled differently in Foursquare/Locu, or placed in a different order. Hence, we are giving store name it a score based on its performance in the Levenshtein Distance formula.

How did you avoid pairwise comparison of all venues across both datasets?

We have avoided pairwise comparison of all venues across both datasets by removing the highest scorers from the pool of possible candidates. Once a candidate is matched and removed, it won't reappear in all subsequent matches. Thus, there is one fewer candidate to match with in each subsequent iteration.