## Team members

username: chinwen
email: cc3677@columbia.edu

username: shazamo
email: mg3845@columbia.edu

username: DarkKnight
email: sn2738@columbia.edu

## Installations:

Code has to be run in python 3.6 environment with following packages:

```
pip install editdistance
pip install sklearn
pip install numpy
pip install pandas
Pip install nltk
```

## Process

### 1) Data Cleaning

The purpose of data cleaning is to ensure that both datasets are in the same format.

**“phone”:** phone numbers were converted into a string with 10 digits.
**“website”:** patterns such as “http://”, “https://” and ”www.” were removed.
**“name”, “street_address”:** all the non-alphanumeric characters were removed. Phrases of numbers from “one” to “nine” were replaced with the digits from 1 to 9. Words like “street” and “avenue” were replaced with their abbreviation “st” and “ave”, respectively.
**“country”, “region” and “locality“:** these fields were removed due to low variance.

### 2) Candidate Pairs Generation

For each locu datapoint we generated a set of candidates points from foursquare and used these pairs in a supervised model.

To generate the candidate points for a locu data point, we sorted the foursquare points based on the latitude and longitude. We got all the foursquare points in the square [ latitude - δ, latitude + δ ] and [ longitude - δ , longitude + δ ] using binary search. We chose δ to be 0.002 to keep the model complexity small and recall to be fairly high. By sorting and using binary search, the complexity of the model is O(NlogN) as opposed to O(N^2).

Moreover, we looked for candidate pairs which may be missed out from the above approach. We indexed the foursquare points by their name, phone and stemmed words in the name. We looked for the phone, name and words in the name of locu data point in foursquare data using the indices and added the corresponding foursquare data points to the candidates of locu data point. The purpose of using the words in the name as indices is to identify the pairs of names such as “55 Gansevoort”(foursquare) and “Hotel Gansevoort”(locu), which were missed during the first location-based candidate pairs search.

## 3) Feature Engineering

We generated 16 features based on the following seven fields of the data points.

**“postal_code”:** we created a binary feature that indicates if the postal code of the locu and foursquare points match.
**“website”:** one feature was generated from the “website” field. We calculated the Levenshtein distance between the websites of locu and foursquare datapoint, and normalized it by the maximum length of both websites.
**“phone”:** we generated 2 binary features, which indicate if the phone numbers match or if the last 4 digits of the phone number match, from this field.
**“street_address”:** two features were created from this field. Firstly, we calculated the levenshtein distance between two addresses to create a feature. Secondly, we extracted numeric values and calculated the jaccard similarity of the list of numbers from both addresses:
**“latitude”:** difference between latitudes was used as a feature.
**“longitude”:** difference between longitudes was used as a feature.
**“name”:** we created four features based on this “name” field. (1) a binary indicator to check if both the names are identical; (2) levenshtein distance between the names; (3)&(4) cosine similarity and fraction of words that match between the two names: we split the name into words, stemmed the words and used the stemmed words to generate the two features.

Apart from the 12 features mentioned above, we derived 4 additional name-related features to ensure that the model can well distinguish true and false labels when the fields “postal_code”, ”website”, “address”, “phone” are all empty in foursquare data. The 4 features are levenshtein distance of the names, fractions of words that match, jaccard similarity of the words in the names, length of the words that match normalized by the minimum length of both names.

### 4) Training

We used the features mentioned above in the section 3 and then split the training data into train and validation set. We applied the random forest classifier and used the number of trees as a parameter in grid search to find the model that gave the best accuracy. Once we obtained the model, we trained the model on the whole training set.

### 5) Testing

We did the same processes mentioned above to obtain the test data set and predicted the probabilities of the test samples using the random forest classifier. We used a threshold of 0.2 to identify true pairs. To remove many to many mappings of locu and foursquare points, we picked pairs with the highest probability.

## Summary

- We achieved 100% precision, 98.75% recall and f1-score of 99.37% by going through the above approach.

- To avoid pairwise comparison of all venues across both datasets, we used the concept of candidate sets. This approach reduced recall, but made the model simpler. Instead of comparing 360,000(600*600) in the training data, we compared ~3,400 pairs.

- The features that have most impact on the random forest classifier are
    - If the phone numbers match
    - Fraction of words that match in the names
    - Levenshtein distance of the names
    - Levenshtein distance of the addresses
    - Jaccard similarity of the numbers in the addresses
    - Jaccard similarity of the words in the names
    - Normalized length of the words that match in the names