# F.R.I.E.N.D.S

Jin Pu, Jingshan Shi, Junyao Wang, Kewei Chen
Offer+ 12/19/2019

Our project is aimed to establish a recommendation system taking advantage of Twitter API to help users find the most ideal and suitable friends of themselves.

- **Assumption:** Network (real life distance), interest, personality, location are the most influential factors of friendship. Personality and location choices are personal preference whereas network and interest yield to computer calculation.
- **Assumption Basis:** Psychological study, intuition, available data from Twitter

## 1.1 Dataset Built

We used Twitter API to download raw data from Twitter database and tried to simulate a real-world community.

- **Problem:** The Twitter users randomly chosen are so disperse (no followings or followers overlapping) that made later network analysis impossible.
- **Solution:**
    - Hash tag research: First, we used the HASHTAGS built in the Twitter research page to reduce the randomness. Like, we searched *campuslife* to concentrate on in-campus individuals, then *collegelife* to further concentrate on college individuals, then *columbiauniversity* to further concentrate on columbia individuals. But it turned out not that desirable because many not student but social media stuff were contained.
    - Followers of the *Columbia University* Twitter account and their first-layer network: Then, we became more concentrated and chose the followers of *Columbia University* Twitter account and their first-layer network to make sure that the community was at least connected and at least had something in common.
    - Delete extreme users: To assure the accuracy of text analysis, we dropped the users who posted less than 30 tweets (which occupy 45% of original dataset), who had extremely sparse WordCloud and who were not English users (using Langdetect Python package).
- **Results:** We established two dataset, USER DATASET (2,292 rows) and TWEETS DATASET (6,445,878 rows).
- **Files:** [Dataset] *all users' tweet.txt*, *user_info_2292.csv*
  [Code] *Step 1.1_Prepare Dataset.ipynb*

## 1.2 Data Preprocessing – Language

Twitter users spread all over the world. But in terms of analysis, non-English users (identified based on its tweets and profile description) should be filtered out. We use *langdetect* package to fulfill this.

- **Files:** [Dataset] *user_info_2292.csv*
  [Code] *Step 1.1_Prepare Dataset.ipynb*

## 1.3 Data Preprocessing - Location

Geometry distance is one of the four dark matters of friendship. Because it determines whether two individuals share background or culture similarity and influences online interactions and offline meetings transformation, which is a more subjective area left for personal choice.

- **Problem:** The LOCATION data directly scrapped from Twitter's personal homepage is a typical VARCHAR-field with no validation and no preset format, so that any non-sense location could be accepted ('blue marble', 'emoji') and countries, regions, cities are all mixed together.
- **Solution:**
    - We first did web-scrapping from WIKIPEDIA and generated a dictionary with countries as keys and corresponding cities as values and then looped through the dataset to initially change the format to be prettier.
    - Then we took advantages of google GEOMAP API to switch the cleaned location into coordinates which would automatically ignore the non-sense ones and then reversed back into (country, region) pairs.
- **Files:** [Dataset] *user_info_2292.csv*
  [Code] *Step 1.1_Prepare Dataset.ipynb*

### 1.4 Descriptive Analysis

After creating the dataset, we did simple descriptive analysis.

- **Files:** [Dataset] user_info_2292.*csv*
  [Code] *Descriptive Analysis.ipynb*

## 2. Layer I of recommender system: MBTI personality

Based on our assumption, personality is another dark matter of friendship. People tend to be more comfortable with some special types of personality, or they may be attracted with people with opposite personality. Thus, it is hard to determine which one is better, so it is also a subjective area left for personal choice.

In this part, we classified users in our dataset to 16 MBTI personality types according to the content of their tweets with Python library FastAI. We tuned the pretrained NLP model AWD_LSTM to serve our goal. Our training dataset is downloaded from Kaggle.

Our reference: https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/the-16-mbti-types.htm?bhcp=1

- **Problem 1:** The model was time consuming.
- **Solution 1:** Run the model on Google Cloud.
- **Problem 2:** There is other models with accuracy over 90% while our accuracy was only 59%.
- **Solution 2:** We found out that the model on Kaggle falsely used one-hot-encoding to calculate accuracy and 59% is close to the best accuracy in recent research, so we still used the model with accuracy of 59%.
- **Files:** [Training Dataset] *mbti.csv*
  [Training Code] *FastAI MBTI-trained on Google Cloud.ipynb* (**Note:** Running the code is time-consuming, so we leave the results in the notebook)
  [Dataset] *tweet.csv models/export.pkl*
  [Code] *MBTI-prediction.ipynb* (**Note:** Before running the code, please save *MBTI-prediction.ipynb* and *models* directory in the directory.)

## 3. Layer Ⅱ of recommender system: Recommendation Based on Machine Learning

From layer 1 we get a pool of valid users and next we are going to apply machine learning to recommend based on "interest" and "network" dimensions.

### A. Feature design

- **Problem:** Till now what we have are only some ids and tweets, thus we need to understand the logic in network and tweets, and then design features.
- **Solution:**
    - Common following [interest]: First, we found out that following (A→B) indicates user A's interest. Therefore, we constructed the feature "common following" by counting the number of common following. Second, we further found that some users have a long following list so he/she tends to have more common following in most cases. To eliminate the scale factor, we standardized our feature: $\frac{common\ following}{\sqrt{following_{user1} \times following_{user2}}}$. Finally, to avoid the denominator from being 0 (invalid), we adjusted it to $\frac{common\ following}{\sqrt{(following_{user1}+1)\times(following_{user2}+1)}}$. (* In our dataset, we don't need to worry about this because all our users have at least one following – Columbia University. But to ensure our method to be more applicable, we still make this adjustment.)
    - Distance [network]: Instead of one-way following, 2-way following can indicates real-life friendship. Thus, we built the feature "distance". The problem was that some users don't have any path between each other, then the distance should be infinity which would cause error. To solve this, we chose 10,000 to represent infinity. It was found that distance in our dataset can be at most 2, thus we believe that 10,000 is big enough.
    - Similarity [interest]: LSI measures the similarity in words and topics, which can measure similarity in interest. All other users' tweets (all users except A) are inputted as dictionary and corpus, then input A's tweets to output the similarity with other users.
- **Files:** [Dataset] all users' *tweet.txt*, *friends.txt*, *following.txt*
  [Code] *Step 3&4_Using ML to Recommend & Visual*

### B. Rebuild dataset

- **Problem:**
    a. To build a classifier, we need datapoints with 0 and 1 label but we now only have each user's following (label 1).
    b. To analyze, we need the user' data as well as user's following's data to find the pattern. However, Twitter API has limitation on how many user ids we can get or how many tweets we can get at one run. It is super hard to get a large dataset ( some user follows 'Fox News' who has millions of followers.)
- **Solution:**
    a. Chose the same number of non-following users and labeled as 0.
    b. Through descriptive analysis we used median level to find "ordinary users" who can be more representative than others. Finally, we have 1974 datapoints.
- **Files:** [dataset] *user_info_2292.csv* [Code] *Step 3&4_Using ML to Recommend & Visual*

### C. Machine Learning

- We tried Decision Tree, Random Forest, Bagging on our dataset using GridSearchCV to find the best parameter. Finally we chose Random Forest because it has best performance.
- **Files:** [Code] *Step 3&4_Using ML to Recommend & Visual*

## 4. Visualization

When a username in our dataset is passed into our program, we will firstly, generate this user's profile description, like the location, personality, frequently used emojis and topics, WordCloud; secondly, recommend 3 friends to this user, with each one's recommendation score (0-100%), common following network with this user, and their detailed profile. In order to provide better user experience, we add more elements into visualization, including common topics, emojis and WordCloud.

- **Files:** [Code] *Step 3&4_Using ML to Recommend & Visual*

### BONUS 1: Personal Topic Analysis

Based on tweets she/he has posted before; we try to find a pattern and deduce the TOP 3 TOPICS a specific user focuses on in his/her daily life so as to clarify target use's interests and complete the match more efficiently. We can even provide them with some insights on what topic they might both be interested in when creating the first conversation.

- **Problem:** We need to generate common topics for friends. Tweets are in unrestrained and in vigorous styles so that every piece of tweet could be about anything or even without main idea. If we just used a person's frequently used words, firstly, we may include meaningless words; secondly, two people's frequently used words may not overlap. If we used LDA model, our result topics would be just piles of words without certain meaning. Therefore, how to do reasonable pre-classification is a concern.
- **Solution:**
    - We chose an auxiliary dataset which contains around 200k news headlines and short descriptions from the year 2012 to 2018 (which is just aligned with the time when Twitter becomes more and more popular) obtained from HuffPost with 40 main categories like arts, healthy-life and sports. We believe there is a high correlation between external news and our impulse to tweet or comment on Twitter. Then we collected all of the news in each category and viewed them as bag of documents and then did the LSI comparison.
- **Files:** [Dataset] *user_info_2292.csv, News_Category_Dataset_v2.json*
  [Code] *Step 3&4_Using ML to Recommend & Visual*

### BONUS 2: WordCloud

We generated WordCloud of tweets of each users appeared in our results.

- **Files:** [Dataset] *user_info_2292.csv.json*
  [Code] *Step 3&4_Using ML to Recommend & Visual*

### BONUS 3: Emoji

- **Problem:** We need to detect emojis in the text. It is complicated to fetch all the emojis, and we need to encode python string via utf-8 to clearly identify them.
- **Solution:**
    - We used Demoji package to accurately find or remove emojis from a blob of text. We select top 50 emojis of each person and generate common emoji with the recommended friends.
- **Files:** [Dataset] *emoji_count_2292.pickle*
  [Code] *Step 1.2 Data Preprocessing (language+emoji).ipynb, Step 3&4_Using ML to Recommend & Visual*