Advanced Recommender Systems Lab

JB. Griesner

MSc Data Science for Business

1. Introduction

The purpose of this lab is to implement a location-based recommender system from regular location-based social network (*LBSN*) data. Usually LBSN data give precise information regarding the locations visited by the users. These data also reveal who are the friends of the users. These two information (geographical + social) can be exploited into a standard collaborative filtering approach. At the end of this lab you should have all the building blocks of the model iGSLR, a *geo-social location recommendation* model. We suggest you to use the Surprise library as a main framework to implement the model. In this lab the part 2 has to be first. Then parts 3 and 4 can be achieved in parallel. Finally the part 5 combine the results of previous parts.

2. Processing of the Dataset

The input dataset for this lab was collected from Gowalla, a popular LBSN that is often used to test recommendation methods with geographical dimensions. In practice the dataset contains user profiles, user friendship, location profiles, and users' check-in history made before June 1, 2011. It contains 36,001,959 check-ins made by 407,533 users over 2,724,891 locations. You can download the dataset from here. First of all, you have to extract the dataset and load it into a pandas dataframe. Filter users who have less than 5 check-ins in total. Associate for each user the list of his friends. Put the result in a new dataframe `df_user_friends`. In the same way, associate for each user the list of the successive locations he visited. Put the result in a new dataframe `df_user_locations`. Then you have to compute for each pair `(user, location)` its corresponding visit frequency in order to build a new `df_frequencies` dataframe. Finally, transform and update the frequencies from `df_frequencies` into the range $[0, 10]$ with the following normalization transformation:

$$x \mapsto 10 \cdot \tanh\left[10 \cdot \frac{x - f_{\min}}{f_{\max} - f_{\min}}\right] \tag{1}$$

where $f_{\min}$ and $f_{\max}$ refers respectively to the minimum and maximum visit frequencies of the dataset. This transformation is necessary in order to take into account of the long tail. You can then load `df_frequencies` into the Surprise framework with the `load_from_df()` function. Finally use the `train_test_split` function to divide `df_frequencies` into a 75 % train set, and 25 % test set.

3. Geographical Computations

From `df_user_locations` you have access for every given user u to the list L_u of all locations he has been located to. Use this list to compute for each user the distances d_{ij} between each pair of locations of the list: $\forall l_i, l_j \in L_u, d_{ij} = \text{distance}(l_i, l_j)$. You will obtain a list D of distances. This list of distances can be used to compute the density $\hat{f}_u$ of any new given distance as follows:

$$\hat{f}_u(d_{ij}) = \frac{1}{|D|\,h} \sum_{d' \in D} K(\frac{d_{ij} - d'}{h}) \tag{2}$$

where $K(\cdot)$ is the normal kernel function with fixed bandwidth:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

with $h = 1{,}06\hat{\sigma}n^{-1/5}$ (n is the number of locations in L_u, $\hat{\sigma}$ is the standard deviation of D). The density $\hat{f}_u$ will be used to estimate the likelihood that a new unvisited location l_i matches the user geographical preferences based on his previously visited locations. This is achieved by computing the distances between l_i and every locations of $L_u = \{l_1, ..., l_n\}$, and then estimating the likelihood of each distance given the distribution of the user. Finally we can take the empirical mean probability $p(l_i|L_u)$ of any new location for any given user:

$$p(l_i|L_u) = \frac{1}{n} \sum_{j=1}^{n} \hat{f}_u(d_{ij}) \tag{3}$$

This probability encodes the geographical likelihood that user u will visit location l_i. You can first implement Equation 2 into a function `density()`. Then you can put Equation 3 into another function `geo_proba()` that will take a user's list of visited locations and a new location as input, and return the probability of this new location as output.

4. Social computations

Every user u_i is associated with his set $F(u_i)$ of friends. From `df_user_friends` you can get the set $F(u_i)$. For every pair of users (u_i, u_k), we can compute their *social similarity* score with the Jaccard coefficient as follows:

$$\text{Sim}(u_i, u_k) = \frac{|F(u_i) \cap F(u_k)|}{|F(u_i) \cup F(u_k)|} \tag{4}$$

Implement this coefficient into a function of two arguments `social_similarity()`. Then this score can be exploited into the standard collaborative filtering model:

$$\hat{r}_{i,j} = \frac{\sum_{u_k \in F(u_i)} r_{k,j} \cdot \text{Sim}(u_i, u_k)}{\sum_{u_k \in F(u_i)} \text{Sim}(u_i, u_k)}$$

Finally we cant to transform the prediction $\hat{r}_{i,j}$ into a probability as follows:

$$\hat{p}_{i,j} = \frac{\hat{r}_{i,j}}{\max_{l_j \in L \setminus L_j} \{\hat{r}_{i,j}\}} \tag{5}$$

5. Generate & Test Recommendations

The goal is to generate a recommendation score $\hat{s}_{i,j}$ for a given user i and a given location j. This recommendation score can be computed with equations 3 and 5 as follows:

$$\hat{s}_{i,j} = \frac{\hat{p}_{i,j} + p(l_i | L_u)}{2} \tag{6}$$

Generate recommendations for each user in the test set and get the precision@K and recall@K for $K \in \{5, 10, 15\}$. To implement Equation 6 you have to create a new prediction algorithm as you can see here. You have to create a `GSLR` class (i.e. *Geographical Social Location Recommendation*) and populate a `fit()` and a `estimate()` functions. These functions will of course depend on the functions you have implemented in parts 3 and 4. To compute the recall and precision, you can reproduce this part of the documentation. You can also use cross validation as shown here.

Bonus: Measure the model performance in comparison with standard baselines.