Joint analysis of geospatial and “friendship” of Gowalla data

15 March 2017

Juan Benjumea

# Introduction

Gowalla is a location-based social networking website where users share their locations by checking-in. The friendship network is undirected and was collected using their public API, and consists of 196591 nodes and 950327 edges. It has a total of 6,442,890 check-ins of these users over the period of Feb. 2009 - Oct. 2010. The aim of the project is to use Data Science tools to analyse and build a recommendation system for both the geospatial check-ins and the social network links in that database. As a part of the project I will tag a user’ as a locals or tourists (who mostly visit monuments, stay in hotels ...) and adapt responses given that tag. I will use the 17400 Paris check-ins from 1366 users and I will join Gowalla location data with Google Places to separate the type of venues reported Gowalla's users.

# Datasets description

Multiple sources of information were gather and used in the developing of this project; the datasets includes the Stanford's SNAP Gowalla data [11], queries to Google Maps API and vectorial geodata from OpenStreetMaps.

The [Gowalla](https://snap.stanford.edu/data/loc-gowalla.html) is an anonymized and clean dataset collected from February 2009 to October 2010 from the Gowalla's startup to capture human mobility in a location-based social network (LBSN). In Gowalla LBSN, people were able to **check-in** at places or **spots** that they visited near to their local vicinity; Check-ins were collected from a mobile application or through a mobile website, the incentive for users was to get advantages in the places they check-in[[1]](#footnote-1). SNAP's Gowalla data is split in two datasets: a) a collection of more than 6.4 million of individual check-ins[[2]](#footnote-2) and b) an undirected graph[[3]](#footnote-3) with 196591 nodes and 950327 edges [11]

For this project a subset of the check-in in Gowalla data was extracted; the subset corresponds a selection of data from Paris, extracting all datapoints in a radius of 30km from the position 48.86°N, 2.35°E (Google’s Paris location). The selected Gowalla data contains for Paris consists in 17496 check-ins between September 2009 and October 2010 [[see github](https://github.com/jubenjum/dssp5-proj/blob/master/data/loc-gowalla_totalCheckins_Paris.txt)]. The undirected graph for Gowalla is not used in this Data Science pipeline given the limits on the selected data as a low number of connections for within Paris, searching time, and other constrains.

The data from Gowalla was enriched using information fetched from Google Places using their web-service API (function [nearbysearch](https://developers.google.com/places/web-service/search)). I developed scripts to do multiple web-scrapping and parsing Google's JSON geodata. The data of Gowalla for Paris has 4178 different check-ins locations (different latitude, longitude pairs), for each one of those locations I searched and fetched all the Google Places within 100m radius of that position, in total I got a total of 19089 individual Google Places.

Other data used in the project is OpenStreetMap are vectors in GEOJSON format for the Paris districts and the Seine River, data that is used on the visualization of Paris geodata.

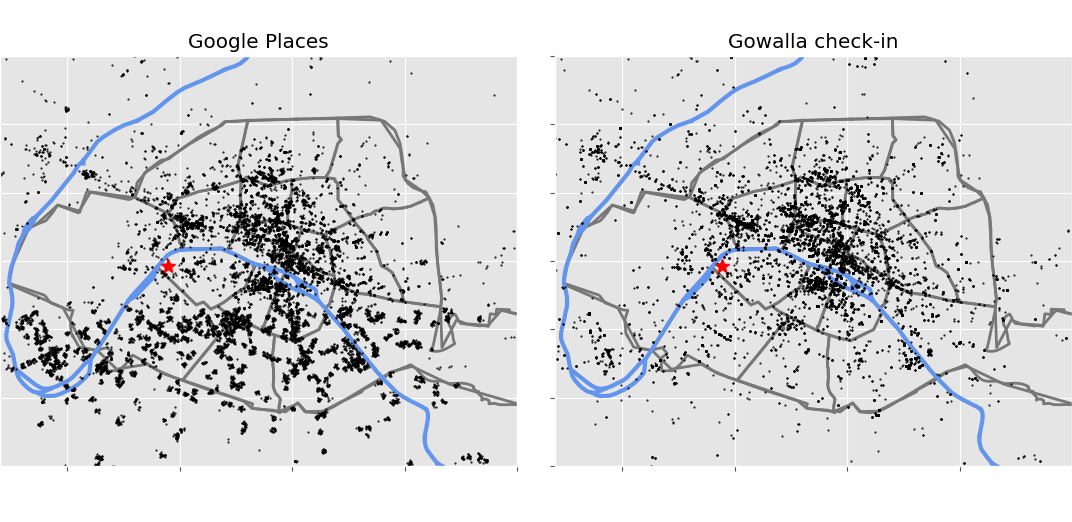


Figure 1: Map of selected datasets, left are the returned POI from Google Places, right are the Gowalla check-ins in Paris, the red start is the location of Eiffel Tower

# Preprocessing

Raw data was given in multiples formats depending of the source; Gowalla format consist on fixed width the columns, that contains: user identification number, the UTC time of the check-in, the location in decimal latitude and longitude, the column is the a unique spot id. For these datapoints the only needed modifications done on the data is on the UTC time, that time was corrected using a day saving time function, this dataset was already clean for research by SNAP, then no extra modifications were done on this data. Gowalla’s selected data was converted to a csv file to do the data processing.

To obtain the Google Places data, for all unique locations in Paris from Gowalla I made requests to for all places around 100m these points in Google using their web-service and requesting all types of places (hotel, monuments, zoo …). Google Places web-service returns a JSON file for each query, as I have 4178 queries and Google set a quota of 2500 queries/day per developer key, I got the information in two batches. A selection of the JSON fields for this data were selected, see Feature engineering section for details on the selection.

Given the proximity of Gowalla check-in locations and the radius of query on Google Sites, I got duplicated places from Google; I searched and removed all that duplicated data from the parsed data using Google data hash code. The total number of POI in Google Places is 30286 POIs, and 23074 POIs after cleaning.

From the map Figure [1](#fig:1), it can be see that there is a fraction of points in Gowalla that are not present on Google, also the density of Google is higher, that can happened because:

* The Gowalla spots doesn't exist anymore, closed restaurants, shops etc.
* Check-ins in private properties (e.g. apartment, offices)
* Google Places POI is gather around 100m of Gowalla's data, though there is a higher density of information.

## Analysis of Gowalla data for Paris

From the analysis of 17496 Gowalla check-in locations, the places that are most visited are in Table 1. The first place corresponds to the CDG Airport and analyzing the data it was found that check-in places at the airport correspond to car rental agencies, that could be due to promotional coupons or reductions when using the application and doing those check-ins. The other 9 major check-ins corresponds to places near-to or in site seeing, which is around 10% of all the check-ins in Paris, meaning that a proportion of data is provided by tourists. However there are two places that are not exclusive from tourists that are Gare du Nord and Montreuil:

|  |  |
| --- | --- |
| **# Check-ins** | **Venue** |
| 402 | CDG Airport |
| 198 | Louvre |
| 194 | Pont des Arts |
| 171 | Eiffel Tower |
| 267 | BNF/François-Mitterrand |
| 114 | Gare du Nord (local people+Eurostar) |
| 106 | Notre Dame |
| 100 | A place near to BNF |
| 93 | Arc de Triomphe |
| 91 | Montreuil (local people) |

Table 1: The top 10 in Paris from Gowalla crossing information with Google Places

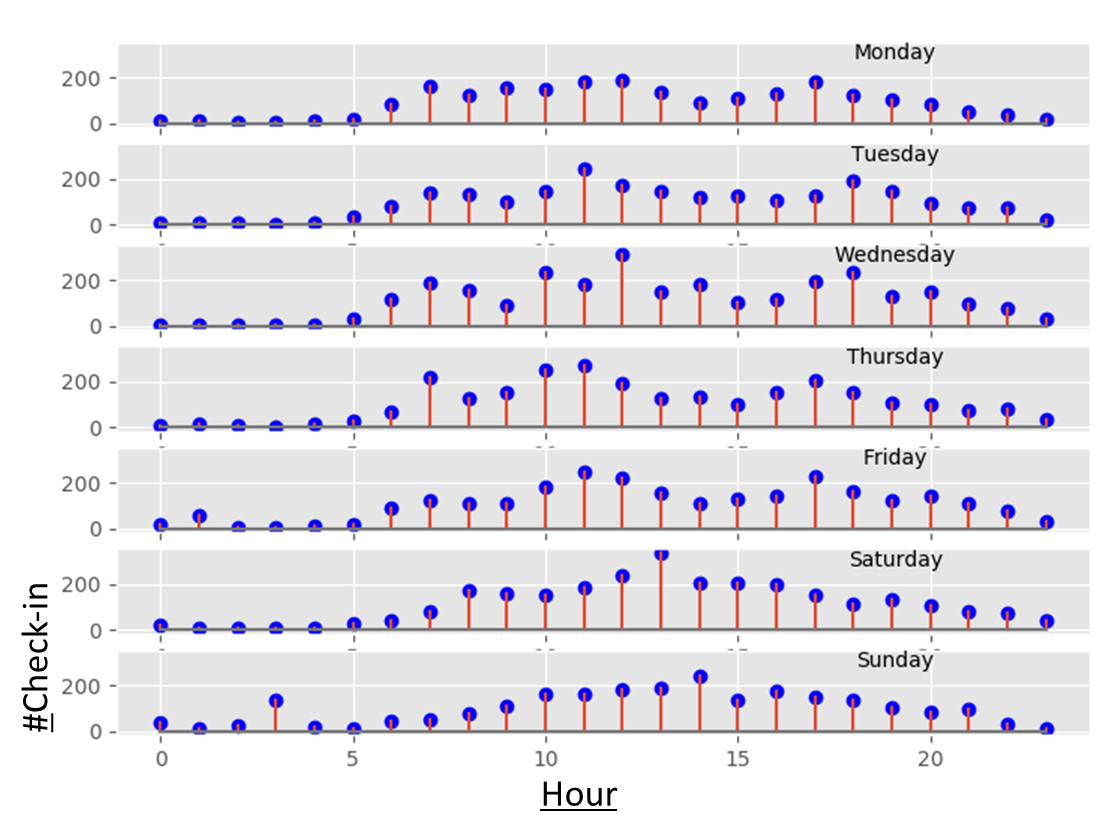


Figure 2: Temporal distribution of Gowalla check-ins in Paris check-ins - hourly

The temporal distributions of Gowalla check-ins in Paris is shown in Figure 2 and 3. The week distribution of data is almost constant (fig 3), small reductions in the number of check-ins on Monday and Tuesday and maybe due to that museums are close. For the hours of the day it’s clear that there is a pattern, most of the check-ins are done during day time (5am to 8pm), and pick around midday, and from Monday to Friday at 7-8am and 5-6pm, that can be a contribution in check-ins from local people.

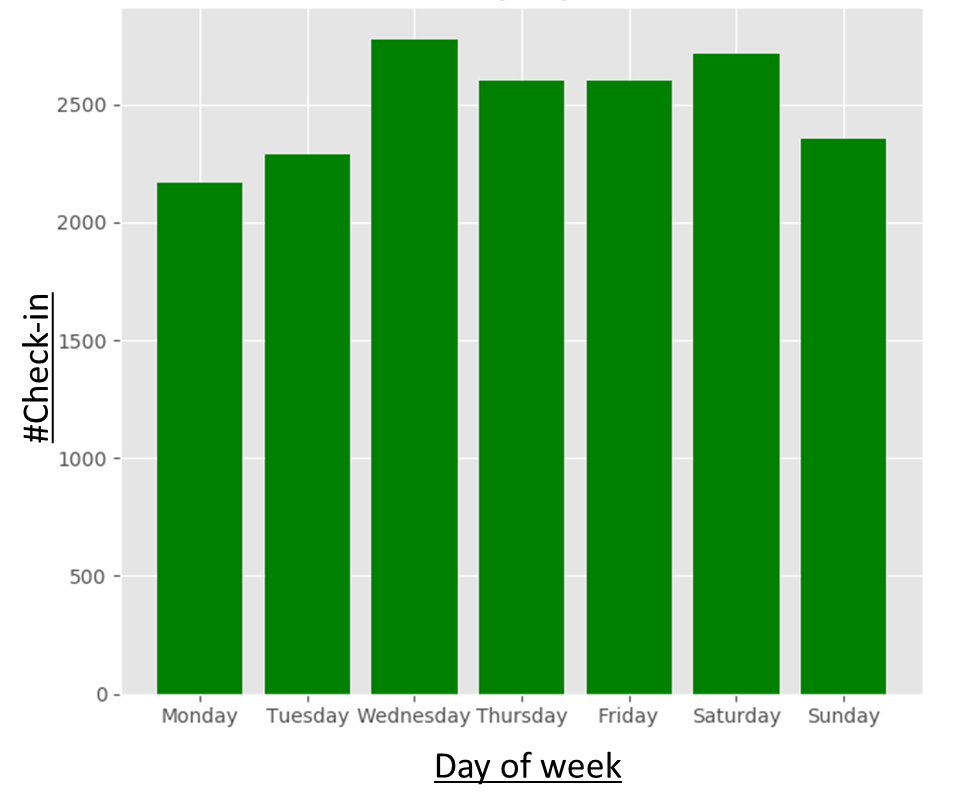


Figure 3: Temporal distribution of Gowalla check-ins in Paris - daily

# Feature Engineering

Two parts were necessary to complete with this project from the engineering point of view, the first is the selection of tools required to process the data in an efficient way given the size and complexity of the processed data, and the second is the feature selection from the data on itself.

## Software packages and used Tools

One part of the entire Data Science pipeline used to produce the results in this report is done with standard Linux command (bash, awk, sed, cut …), the size of the data and the number of samples allows making a fast exploration of data, and do searching, cleaning and querying efficiently using those command line tools. For the Data Science part I used the packages that we learned on the DSSP5 at Ecole Polythenique, for the developing part I used python and the following packages:

* **Matplotlib**: to make figures and maps.
* **Scikit-learn**: the non-supervised classification with KNN - Graph[[4]](#footnote-4)
* **FAISS**: a new KNN implementation from Facebook AI Research [10], which allow to project features in low dimensions and it does searches in high dimension of data, also uses CUDA/CPU to accelerate the computations.
* **Pandas**: as a container for all data into dataframes.
* **Networkx**: I used to this package to explore and plot the undirected graph resulting from FAISS/Scikit implementations of KNN-Graph.

I learn about Scikit-learn and Pandas during the DataScience classes at the Ecole Polythecnique, and I found invaluable the ideas and tools that were presented during the DSSP5 program.

## Feature selection

From Gowalla’s dataset its was selected only the check-in data for Paris, the data was transformed to a pandas dataframe, t time stamps were corrected to take into account day time saving.

Google Places returns JSON containers with the structure described at their webpage[[5]](#footnote-5), for this DSSP pipeline used only a subset of Google data and the selected is:

* **geometry/location**: latitude and longitude, numerical values
* **id**: location Google’s hash as a 10 character string
* **name**: name of the place, a string
* **type**: a string with one of Google's supported see Table A1 or [types](https://developers.google.com/places/supported_types)

Google classify their places in 130 types, in this project I removed two types (establishment and point\_of\_interst) due that there were redundant, the full list of all supported types are listed on the Table A1 and Google’s developer webpage. All Google parsed values are kept on dataframes with the fields: latitude, longitude, place\_id (Google’s hash code), name of venue[[6]](#footnote-6) and the type is transformed from string to a space with d=128.

The spot in Google Places can contain multiple types, for example for CDG-Airport location it contains: *airport*, *bus\_station, transit\_station*, and in nearby to the his location there are places that as *car\_rental* (x4 times) and *travel\_agency*, *bar* and *night\_club,* it makes that each spot (places around a location) have a different signature depending on what is on it; I translated these Google string types to a categorical vector of dimension d=128, containing 1 when the type is present is present the and 0 it’s absent; in the code this panda element is spot signature. These spot signatures are used to create a user profile (user signature in the code).

To create the user signature I extended the concept of binary check-in vectors developed in [1], [2] and [3] to take into account the created spot signatures from Google Places. To compute the user signature I searched the n-closest geographical (n=10) spot signatures for each one of the spots the user has been (n-places), the number of single spot signatures will be then n-closes\*n-places, then the user signature is the normalized sum of all selected stacks (see fig 4 as example). After these operations it will be a collection of people signatures that are vectors also in the space.

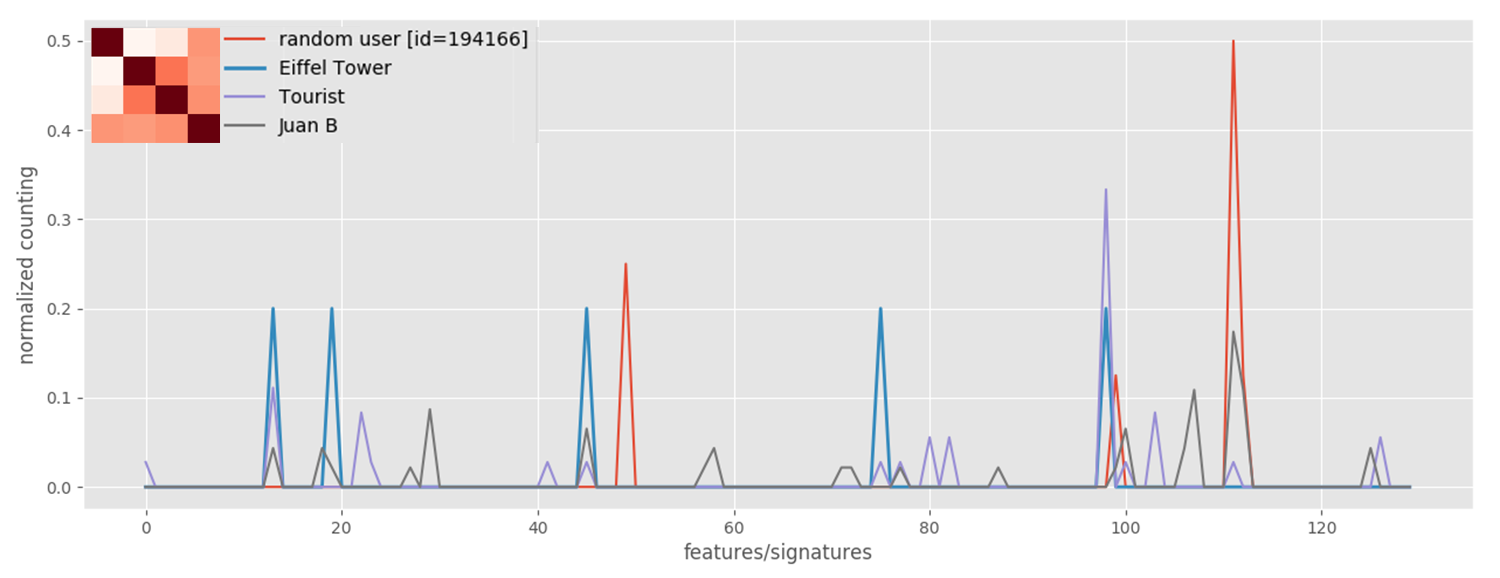


Figure 4. Spot signature for Eiffel Tower (blue) and three examples of user signatures, a random selected user (red), a made up tourist that visits the spots in table 1 (violet) and Juan B (gray), the matrix in the label is the L2-Similarity.

From fig 3 it can be seen that there some patterns emerge: Eiffel Tower and lambda tourist are similar because the tourist was in Eiffel Tower spot, the random person is different than a tourist then it may be doing check-ins in non-touristic places, Juan B is similar to all because check-ins nearby touristic and non-touristic places.

# Method used unsupervised classification

In this DS project I am doing non-supervised classification (pattern recognition) to search similarities between peoples signatures and/or spot signatures, specifically I was interested on learning and testing a new algorithm developed by Facebook AI Research[[7]](#footnote-7) (FAIR). The algorithm is described in [10] and it is available on Facebook github[[8]](#footnote-8) and can be used for free for non-profit applications. The main application of this algorithm is similarity search in high-dimensional features space with indexing, for example: images, sounds, and all human generated content.

The classification implementation of FAIR is the k nearest neighbors graph (KNN-G) from vector collections; the similarity distance used on the algorithm is L2. The implementation of the algorithm can project the features space d (for example here is 128) to a lower binary space (8, 32, 64, 128 bits) and do the searching in batches. The implementation can work in parallel CPU and/or GPU. For the DSSP pipeline I used the CPU in all the full d-space.

As a baseline I used is the unsupervised learner implementation of nearest-neighbor from scikit-learn[[9]](#footnote-9) and using as a metric L2 to have comparable results with Facebook implementation.

# Analysis and interpretations of the results

I build two graphs with networkx, one using faiss implementation of the KNN-graph and the other with the scikit-learn using all data (1366 user\_id), the parameters used on the comparison are in the compare\_search.py script. For k=5 and k=50 the results are not consistent, the graphs are not isomorphic (nx.is\_isomorphic(G\_scikit,G\_fais))

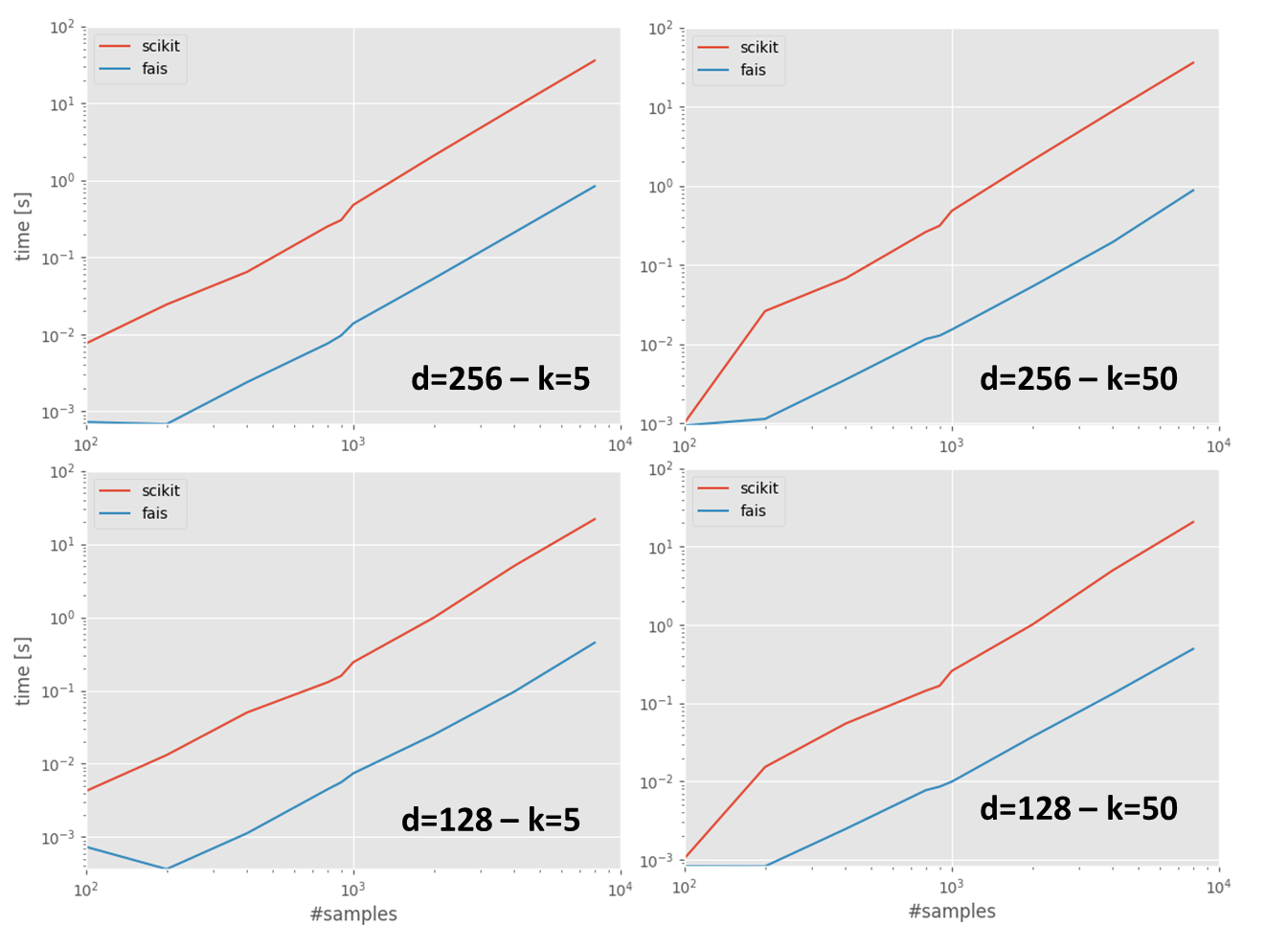


Figure 5. Computing time for KNN-G for scikit-learn (red) and FAISS (blue) implementations for the cases d=128 and 256 and k=5 and 50.



Figure 6. Result of network connections for the three selected users and the selected site, the number of neighbors Is k=50 using FAIR KNN-G algorithm.

# Conclusions and potential further work

From the analysis of Gowalla it could be seen that it exist a pattern in the time of check-ins and location of those check-ins, joining these location with Google Site enrich the database, and I found that it could be possible to produce a system that learns from the check-ins and it is possible to find patterns on user location check-ins, however in that aspect of the this job I feel that it is needed to do more work in the collection of data for Paris from check-ins (more check-ins), also to have a better database of venues that covers the entire city.

From the comparison of the two tested KNN-G algorithms, I found that both, FAISS and Scikit-learn gives the same results (similar graphs), however the computing time is at least one order of magnitude lower with in the CPU using FAISS in comparison with Scikit-learn.

As potential further work, I will continue to work with the FAISS implementation KNN-G, I will do test on the GPU version of the library. The main objective in the near future is to use that library in the Zero Speech Challenge[[10]](#footnote-10); the challenge consists in implementing a non-supervised algorithm to search for pseudo-terms/words in speech records, I will use this KNN-G algorithm to search for patterns in waveforms.

# References

[1] G. Ference, M. Ye, and W.-C. Lee, “Location recommendation for out-of-town users in location-based social networks,” in *Proceedings of the 22Nd aCM international conference on information & knowledge management*, 2013, pp. 721–726 [Online]. Available: <http://doi.acm.org/10.1145/2505515.2505637>

[2] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, “Time-aware point-of-interest recommendation,” in *Proceedings of the 36th international aCM sIGIR conference on research and development in information retrieval*, 2013, pp. 363–372 [Online]. Available: <http://doi.acm.org/10.1145/2484028.2484030>

[3] H. Wang, Z. Li, and W.-C. Lee, “PGT: Measuring mobility relationship using personal, global and temporal factors,” in *Data mining (iCDM), 2014 iEEE international conference on*, 2014, pp. 570–579.

[4] S. M. Rahimi and X. Wang, “Location recommendation based on periodicity of human activities and location categories,” in *Advances in knowledge discovery and data mining: 17th pacific-asia conference, pAKDD 2013, gold coast, australia, april 14-17, 2013, proceedings, part iI*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 377–389 [Online]. Available: <http://dx.doi.org/10.1007/978-3-642-37456-2_32>

[5] A. Narayanan and V. Shmatikov, “De-anonymizing social networks,” in *Security and privacy, 2009 30th iEEE symposium on*, 2009, pp. 173–187.

[6] J.-D. Zhang and C.-Y. Chow, “GeoSoCa: Exploiting geographical, social and categorical correlations for point-of-interest recommendations,” in *Proceedings of the 38th international aCM sIGIR conference on research and development in information retrieval*, 2015, pp. 443–452 [Online]. Available: <http://doi.acm.org/10.1145/2766462.2767711>

[7] Q. Yuan, W. Zhang, C. Zhang, X. Geng, G. Cong, and J. Han, “PRED: Periodic region detection for mobility modeling of social media users,” 2016.

[8] L. Rossi and M. Musolesi, “It’s the way you check-in: Identifying users in location-based social networks,” in *Proceedings of the second aCM conference on online social networks*, 2014, pp. 215–226 [Online]. Available: <http://doi.acm.org/10.1145/2660460.2660485>

[9] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, “A random walk around the city: New venue recommendation in location-based social networks,” in *2012 international conference on privacy, security, risk and trust and 2012 international confernece on social computing*, 2012, pp. 144–153.

[10] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with gPUs,” *CoRR*, vol. abs/1702.08734, 2017 [Online]. Available: <http://arxiv.org/abs/1702.08734>

[11] E. Cho, S. A. Myers, and J. Leskovec, “Friendship and mobility: User movement in location-based social networks,” in *Proceedings of the 17th aCM sIGKDD international conference on knowledge discovery and data mining*, 2011, pp. 1082–1090 [Online]. Available: <http://doi.acm.org/10.1145/2020408.2020579>

# Appendices

|  |  |  |  |
| --- | --- | --- | --- |
| accounting | pet\_store | embassy | taxi\_stand |
| cemetery | storage | light\_rail\_station | campground |
| gym | amusement\_park | political | funeral\_home |
| natural\_feature | courthouse | sublocality\_level\_3 | meal\_takeaway |
| rv\_park | insurance\_agency | beauty\_salon | premise |
| administrative\_area\_level\_1 | pharmacy | ~~establishment~~ | train\_station |
| church | store | liquor\_store | car\_dealer |
| hair\_care | aquarium | postal\_code | furniture\_store |
| neighborhood | dentist | sublocality\_level\_4 | mosque |
| school | intersection | bicycle\_store | real\_estate\_agency |
| administrative\_area\_level\_2 | physiotherapist | finance | transit\_station |
| city\_hall | street\_address | local\_government\_office | car\_rental |
| hardware\_store | art\_gallery | postal\_code\_prefix | gas\_station |
| night\_club | department\_store | sublocality\_level\_5 | movie\_rental |
| shoe\_store | jewelry\_store | book\_store | restaurant |
| administrative\_area\_level\_3 | place\_of\_worship | fire\_station | travel\_agency |
| clothing\_store | street\_number | locality | car\_repair |
| health | atm | postal\_code\_suffix | general\_contractor |
| painter | doctor | subpremise | movie\_theater |
| shopping\_mall | laundry | bowling\_alley | roofing\_contractor |
| administrative\_area\_level\_4 | plumber | floor | university |
| colloquial\_area | sublocality | locksmith | car\_wash |
| hindu\_temple | bakery | postal\_town | geocode |
| park | electrician | subway\_station | moving\_company |
| spa | lawyer | bus\_station | room |
| administrative\_area\_level\_5 | ~~point\_of\_interest~~ | florist | veterinary\_care |
| convenience\_store | sublocality\_level\_1 | lodging | casino |
| home\_goods\_store | bank | post\_box | grocery\_or\_supermarket |
| parking | electronics\_store | synagogue | museum |
| stadium | library | cafe | route |
| airport | police | food | zoo |
| country | sublocality\_level\_2 | meal\_delivery |  |
| hospital | bar | post\_office |  |

Note: Google types that were eliminated are ~~strikethrough~~.

Table A1. List of all types returned by Google Place API

All code and data files used on this report are kept in the github repository:

**https://github.com/jubenjum/dssp5-proj**

1. https://en.wikipedia.org/wiki/Gowalla [↑](#footnote-ref-1)
2. the columns are 1-user\_id, 2-UTC\_Time, 3-latitude, 4-longitude and 5-spot\_id; 1,3,4 are directly used on the DSSP's workflow; 2 is corrected for day saving time, however all variables are keep on the dataframes [↑](#footnote-ref-2)
3. Contains in the columns user\_id friend\_id. [↑](#footnote-ref-3)
4. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.kneighbors\_graph.html [↑](#footnote-ref-4)
5. https://developers.google.com/places/web-service/search [↑](#footnote-ref-5)
6. UTF-8 string UTF-8 with possible multiple language description of each place, for example “L'Arc de Triomphe de l'Etoile” is the same than “停点” [↑](#footnote-ref-6)
7. https://research.fb.com/category/facebook-ai-research-fair/ [↑](#footnote-ref-7)
8. https://github.com/facebookresearch/faiss [↑](#footnote-ref-8)
9. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html [↑](#footnote-ref-9)
10. http://sapience.dec.ens.fr/bootphon/ [↑](#footnote-ref-10)