Spying On Runners

Advanced Security Project

Léo Besson   Paolo Daolio

22/01/2021

The aim of this project is to infer social links between people based on their running records found on a community french website. The first part of this study was to collect the maximum amount of data from the website. The second part was to analyse this data using similarity metrics to find matching profiles. The produced tool is able to quantify the probability of two people knowing each other and for one person, reavealing all possible related people. The results of this study still have to be tested by contacting the recorded people and verifying the correctness of the infered social links.

1 Introduction

blablabla

2 Scraping

The python language was used for the scraping phase, in particular a module called Selenium which allows you to easily obtain a "webdriver" that is a construct that can freely surf the net exactly like a user. The driver used is a "chromedriver" which allows you to navigate on Google Chrome. The basic idea of a scraping process is to find a fixed path through which the driver can be able to constantly obtain new information, process and finally store them. Thanks to the analysis of the website it was found a page with all the races chronologically ordered; for each race in the list the driver collected: number of partecipants, date, name of the race, elevation (if present), length and the name of every runner with some related information. Once collected, the data is saved in a open source relational database, postgreSQL was used for its simplicity of installation and for its good functioning with python. in a short time this method proved to be particularly effective allowing to populate the database in a consistent way. Almost six hundred thousands of different people are present in the database, who, in total, have recorded more than one million races. The races scraped from the sites are all about the last 3 years (from early 2018 to 2021). Another intresting data collected is the "Club" in which a runner can enroll and race for, in more than two thousands races over one hundred thousand different clubs partecipated.

3 Data Analysis

All the collected data is processed to guess linked people in the set of collected runners. To do so, we need a function to quantify how probable is the fact that two people know each other based on their race records. This type of function is called similarity metrics. Metrics needs to be analysed to select the correct threshold and to compare how well they perform on this specific application. Finally, the refined metrics are used to build the Python application.

3.1 Selected Similarity Metrics

The metrics chosen for this application are taken from the paper written by Cunche, Kaafar, and Boreli (2012). To infer social link between runners, we need to see the races they have in common in their record and how likely they have been in contact in this race. To measure this second parameter, we use the number of participants in each common race. The less people there are in the race, the most probable it is that the two persons were in contact and may know each other. All the metrics are implemented in the file `analysis_module/metrics.py`

- *Jaccard index*: This metric focuses on the proportion of common races in the record. it is defined as $\mathrm{J}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. It is implemented in the function `jaccard_index`.

- *IDF similarity*: This metric depends on a measure of the rarity of a race defined as $IDF_i = \log \frac{1}{f_i}$ with the frequency of the race f_i the number of participants. Finally the metric is define as followed.

$$\text{Cosine-IDF}(X, Y) = \frac{\sum_{x \in X \cap Y} IDF_x^2}{\sqrt{\sum_{x \in X} IDF_x^2} \sqrt{\sum_{y \in Y} IDF_y^2}}$$

  It is implemented in the function `idf_similarity`.

- *Adamic similarity*: This metric also depends on the frequency f_i defined as the number of participants in the race i. This metric is defined as $\mathrm{Adamic}(X, Y) = \sum_{i \in X \cap Y} \frac{1}{\log f_i}$. It is implemented in the function `adamic_similarity`.

- *Modified Adamic similarity*: In order to put more weight on the rarity of the races, the Adamic similarity is changed to $\mathrm{Psim\text{-}q}(X, Y) = \sum_{i \in X \cap Y} \frac{1}{f_i^q}$. It is implemented in the function `psim_q`.

3.2 Metrics Evaluation

The following part aims at assessing how accurately the previous metrics are at infering social links. As all the metrics return a number, we need to define a threshold such that if the returned value is above the threshold, we consider that the two persons are linked and if the returned value is below the threshold, we consider that the two person are not linked. All the methods described in this part are implemented in `analysis_module/metrics_analysis.py`

The first method followed by the paper written by Cunche, Kaafar, and Boreli (2012) is to separate the database into two sets: a first set where every person is really socially

linked to another one and another set where there is no couple of linked people. Then, we run the metric on the whole set and count the number of true positives, true negatives, false positives and false negatives. The closer we get from reality, the most accurate is the metric. As we didn't had the time and ressources to build such databases based on verified testimonies, we made the following simplifying assumption: *two people know each other if and only if they ran in the same club at least one time.* Running the metrics analysis based on this assumption gave erratic results so we realised that this assumption is false and could not replace real data on social links.

As we could not properly conduct this analysis, the choice of the threshold and metric in the final application is left to the user.

3.3 Analysis Python Module

The final application is provided in `analysis_module/main_link.py`. To work it needs to be connected with the database filled with the data scraped by the scraping module. This data is stored in `Data/backup_[date]` and can be loaded in a postgreSQL database with the command `psql runners < backup_[date]`. It may be required to tweak the file `analysis_module/database/connection.py` to match each specific configuration of the database. All the queries needed to perform the analysis are implemented in `analysis_module/database/manager.py`. The final application asks for a name and compute the metrics for all potentially linked runners from the database. We start by fetching all runners that has at least one race in common with the targetted runner. Then we compute the metrics matrix that stores the value returned for each metric and for each runner. Finally we provide a way for the user to have a comprehensive view of this matrix by displaying the most probable linked runners based on each metric or the mean of all them. This application was designed for runners to check if they suffer from a privacy breach on this particular website and to see if they are vulnerable to social engineering attacks.

4 Conclusion

blablabla

References

Cunche, Mathieu, Mohamed Ali Kaafar, and Roksana Boreli. 2012. "I know who you will meet this evening! Linking wireless devices using Wi-Fi probe requests." In *WoWMoM - 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks - 2012.* San Francisco, United States. https://hal.inria.fr/hal-00747825.