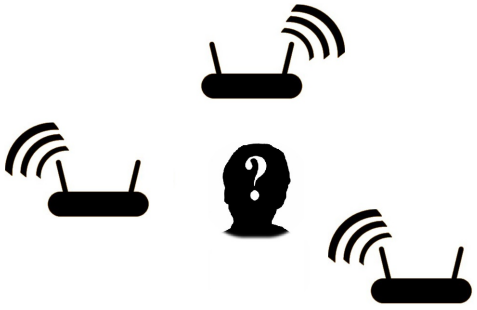
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de
Eletrónica e Telecomunicações e de Computadores



Mining Social Individuals Movements and Friends based on mobile devices

Pedro de Brito e Nunes
(Licenciado em Engenharia Informática e de Computadores)

Trabalho de projeto realizado para obtenção do grau
de Mestre em Engenharia Informática e de Computadores

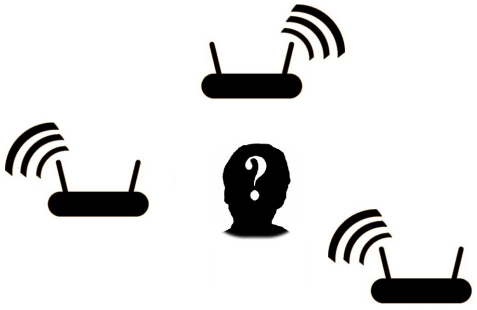
Orientadores:
Doutor Nuno Cruz
Doutor João Ferreira

Júri:
Presidente: Doutor Nuno Miguel Soares Datia
Vogais:
Doutor Hugo Alexandre Tavares Miranda
Doutor Nuno Miguel Machado Cruz

Outubro, 2015

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de
Eletrónica e Telecomunicações e de Computadores



Mining Social Individuals Movements and Friends based on mobile devices

Pedro de Brito e Nunes
(Licenciado em Engenharia Informática e de Computadores)

Trabalho de projeto realizado para obtenção do grau
de Mestre em Engenharia Informática e de Computadores

Orientadores:
Doutor Nuno Cruz
Doutor João Ferreira

Júri:
Presidente: Doutor Nuno Miguel Soares Datia
Vogais:
Doutor Hugo Alexandre Tavares Miranda
Doutor Nuno Miguel Machado Cruz

Outubro, 2015

“We learn something every day, and lots of times it’s that what we learned the day before was wrong.”

Bill Vaughan

“Software is a great combination between artistry and engineering.”

Bill Gates

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Abstract

Área Departamental de Engenharia Eletrónica e de Telecomunicações e de Computadores

Master's degree in Computer Science and Computer Engineering

Mining Social Individuals Movements and Friends based on mobile devices

by Pedro Brito Nunes

This thesis aims to provide context-awareness through identification of anonymized users in network trace records by crossing data gathered from social network websites.

Most people nowadays carry a mobile device with wireless functionality. This allows recording users' internet connection related metrics, such as incoming and outgoing traffic, IP address or location in a business wireless network that requires authentication, like university campus or company premises.

Constant internet access in mobile devices grants the use of social media everyday and everywhere, from a different vast of social networks and users, which can lead to data mining of information such users' activities and events. By combining information gathered from social networks and Eduroam's access records it's possible to identify a user's social network identity, even while using anonymized WiFi records.

The WiFi records are anonymous to ensure user privacy; however, most users leak some information by social events, such as their current geographic location, that allow us to identify them in the anonymous records.

An application was developed to obtain social networks users' activity and relate it with the WiFi connections obtained from the AP records at the Eduroam network available at ISEL. The combining of two sources of information lead to the presented results, which include identification of users on anonymized network access records.

The final results were satisfactory, as a specific user could be identified using only his network presence and a few social events, and then with geographic information through AP information extrapolation and even less social events.

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Resumo

Área Departamental de Engenharia Eletrónica e de Telecomunicações e de Computadores

Master's degree in Computer Science and Computer Engineering

Mining Social Individuals Movements and Friends based on mobile devices

by Pedro Brito Nunes

Esta tese tem como objetivo fornecer contexto a dados anonimizados através da identificação de utilizadores por informação recolhida de *websites* de redes sociais.

Maioria das pessoas possui e utiliza diariamente um telemóvel com funcionalidades *wireless*. Isso permite manter registo de métricas das ligações em redes empresariais com autenticação, como tráfego na rede, endereços *ip* or localização, no *campus* de uma universidade ou local de trabalho.

O constante acesso à internet em dispositivos móveis permite uma maior utilização de multimédia social, por parte de diferentes redes sociais e pessoas, o que pode originar a *data mining* de informação recuperando as atividades de utilizadores e eventos. Combinando essa informação de redes sociais e dos registos das ligações à rede Eduroam à internet é possível identificar utilizadores nos registos anónimos através da sua identificação na rede social.

Os registos WiFi são anónimos para garantir a privacidade do utilizador; porém, a maior parte dos uitizadores disponibilizam involuntariamente alguma informação sua nas redes sociais, como a sua localização geográfica em determinado momento e hora, o que nos permite identificá-los nos registos.

Foi desenvolvida uma aplicação que obtém actividade de utilizadores na rede social e relaciona-a com os registos da rede WiFi Eduroam disponível no ISEL. A combinação das duas fontes de dados leva aos resultados apresentados neste documento, que inclui a identificação de utilizadores anónimos na rede.

Os resultados finais são satisfatórios, na medida em que se conseguiu identificar um utilizador específico utilizando simplesmente a sua presença na rede e alguns eventos sociais, e de seguida com informação geográfica e ainda menos eventos sociais.

Acknowledgements

I would like to express my gratitude to my supervisor Eng. Nuno Cruz, who most of all, pushed me continually to reach the most of my research, teached and helped through the several problems developed along this work. Work which wouldn't be possible without his support and encouragement.

I want to thank Eng. Nuno Leite and Dr. Artur Ferreira, who answered all my desperate calls for help when in most need. Their patience, understanding and advice guided me to make the most important breakthroughs on my work which allowed me to get better results than expected. My sincere thanks to them.

I would also like to thank my girlfriend Joana Vaz and my friends André Jerónimo, João Vaz, Daniel Albuquerque and André Ferreira for their support and understanding for when this thesis had to come before company. Special thanks to João Belbut who helped on this work by creating all social events used on the research.

Thanks to my family, who supported me along all of this path, since this work's beginning until the end. Without their persistance for me to go on this wouldn't be possible.

Finally, my appreciation goes to the university Instituto Superior de Engenharia de Lisboa by providing me with the knowledge and the best teaching possible in the field.

Contents

List of Figures

List of Tables

Abbreviations

ISEL	**I**nstituto **S**uperior **E**ngenharia **L**isboa
IPL	**I**nstituto **P**olitécnico de **L**isboa
AP	**A**ccess **P**oint
REST	**RE**presentational **S**ate **T**ransfer
GPS	**G**lobal **P**ositional **S**ystem
MAC	**M**edia **A**ccess **C**ontrol
WiFi	**Wi-Fi**, defined as any wireless local area network by Wi-Fi Alliance
API	**A**pplication **P**rogramming **I**nterface
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
LSN	**L**ocation-based **S**ocial **N**etwork
SNE	**S**tochastic **N**eighbor **E**mbedding

Chapter 1

Introduction

The increasing sales growth rate of smartphones and laptops[1–3] make wireless networks a useful source of data for the study of human mobility. Wireless networks are a common utility found today at a larger number of institutions, from corporate premises to academic campuses

The widespread availability of wireless network access records, required for law or user policy enforcement, makes it possible to extract a raw location of the device, by observing the access point at which a connection was established. Knowing at what time a device was connected to each AP it's possible to extract the location and time for that device. Considering mobility as a primary source of context information for several applications, the user's movement history can be used in many ways such as prediction of user behaviour or identification of a user on anonymous dataset such as AP wireless records.

With mobile devices market growth came not only mobility research, but also an increase of the usage of social networks[4, 5] - now people use them on the go. In some specific circumstances or applications, it is useful to establish a relation between a user on a publicly available anonymized dataset of access records from a WiFi network and another user of a social network. If an user X on the wireless access records produces data at the same time and place that user Y from the social network does, then there's a probability that they're both the same person, or have some kind of relation between them; combining that with repetion of events, it is possible to associate the identities of a person on the wireless and social networks. Combining datasets from anonymized AP wireless records and social networks geo-referenced data enables the identification of a user, proving that anonymity on a single dataset doesn't always protect an user's identity.

The anonymity on the wireless records ensures user privacy on datasets that are available publicly, being it for research or challenge purposes[1]

The remainder of this Chapter addresses the goals, objectives as well as the obstacles to overcome, followed by a small approach on how this work could branch out; Chap. 2 goes through some related work on the subject; Chap. 3 goes through the implementation process used to address the mentioned goals; Chap. 4 presents the obtained results and on Chap. 5 the conclusions are presented.

1.1 Goals

The aim of this thesis is to present a proof-of-concept case of contextualization of anonymized data from wireless records to a user of social networking sites such as Twitter, by crossing data gathered from both sources.

To create a proof of concept we used two different data sources. The first are the anonymized AP wireless records of May 2015 from the Eduroam[2] network available at Instituto Superior de Engenharia de Lisboa (ISEL)[3]. The second, is are the Twitter posts made by a single user at ISEL premises for the same month. Our aim is to identify that user on the wireless records.

1.2 Problem and methodology

This thesis answers the question:

"How many social network check-ins are needed for anonymity to be broken?"

The answer comes from two major points - Identifying a user on the network only from his presence in there, then identifying the same user using also geo-referenced information. The first subsection addresses the problem that is identifying a user on the network from social network information, while the second subsection focuses on how is possible to use geographic location on anonymous data. **It is intended to not have or use any real information on the**

[1] Example - TIM Big Data Challenge - http://www.telecomitalia.com/tit/en/innovazione/big-data-challenge-2015.html

[2] https://eduroam.pt/suporteUtilizadores/faq

[3] https://www.isel.pt/

AP traces, such as real login usernames, device MAC addresses or APs real locations. All results come from anonymous data or publicly available information.

1.2.1 Identifying a user on anonymous data

Since the AP record files contain the timespan for each access of an user's WiFi device, it is known where the user has been. To achieve the goal of this thesis, identifying a social network user on a dataset of anonymized wireless access records, the following requirements have to be addressed:

1. Which social networks provide Application Programming Interfaces (APIs) that meet the requirements for identification?
2. Which of those social networks are frequently used by the type of population in question?
3. How is a web profile tied to an anonymized user on the local network?
4. How can geographic information help?

The APIs of some of the most used social networks have to be used and some experiments had to be performed to infer how much data can be gathered from API calls. Three important aspects have to be taken into consideration when finding a social network API useful for our context:

i) the response has to contain geographic location coordinates – from those coordinates it's possible to know where the user was when the event occured, so data from the wireless records can be cross related with the user's position;

ii) it needs to provide date of creation – without a date it's impossible to cross relate with the data from the wireless records at the same time of occurence, and;

iii) the response also needs to contain personal identification – the real name is optimal to add context to the anonymized data from the AP traces.

The population in which the experiment is conducted includes all users of the Eduroam network, inside the premises of ISEL, Lisbon, Portugal. Eduroam is an worldwide available WiFi network, that is available to ISEL users. Eduroam network is set inside institutions and provides access to the network through a username/password combination, allowing users - students,

teachers and staff - to be identified. Most of internet users in Portugal are also users of social networks with great emphasis on using them on mobile devices.

Disregarding the bottlenecks that may surge with using social networks APIs (i.e. not complying with the three requirements described above), three initial social networks were selected for testing, being Facebook, Instagram and Twitter. The main reason for those choices were popularity and usage amount among the studied population[6] as more usage means more data and information to be of use for this study. Starting the testing with Facebook, the most used social network in Portugal, it couldn't be used due to the API having privacy enforcing settings which make it impossible to get events by geographic location. That leaves Facebook out in that regard. Since Instagram and Twitter don't have nearly the same portuguese population on them, despite passing the API response requirements, the analysis on them had to be done in a proof-of-concept test case. The test case was done on a single user from Twitter. More techincal info about the APIs and their particularities is described on Chap.3.

A single person has to be narrowed down from a list that has thousands of users. The first approach consists of cross-relating directly the Twitter posts of an account to the AP wireless records: users that have a single connection opened on the network at the same time that the tweet was posted, are candidates to be associated to that profile. The more tweets that are used, the less candidates are on the final list. The second approach consists of not only cross-relating Twitter posts and connection times, but also the geoghraphic location from where the tweet was posted and the AP ID from where the users were connected at the time. With that, the narrowing process eliminates both people not connected to the network at the time of the tweet and not connected near that geographic location, giving a much more short and precise list with less tweets needed. Since AP locations aren't disclosed in the wireless records, they have to be extrapolated from the available data using distancing and clustering algorithms. The Chapter 3 extends this and applies this methodology to detail.

1.2.2 Mapping access points to geographic locations

Each entry of the wireless records dataset is represented by a username, connection start time, connection end time, device MAC address and the AP to which the connection was made. The usernames, MAC addresses and APs are just integers so the real values remain anonymous, impossible to tie them in to the real subjects. The APs real location would help to narrow down quickly to where the user is, instead of using only its network presence. So a question arises:

"How can each AP be connected to its real location?". In order to address the question we started by calculating the average disconnect time from when a device hands-off the connection from one AP to the next. The hand-off can occur in two distinct cases: when the user's device leaves the AP's radius coverage and then enters the next one, triggering the hand-off; and when the device is on an overlapping radius of two or more APs and the user's device driver *decides* which AP to hand-off the connection to, or keep the current one[7]. Not all combinations of pair-wise temporal average distance between APs exist, due to inexistent of such hand offs, due to the existence of other pairs on the path between two distant APs. Through distance between points, it is now possible to display an undirected graph where the nodes represent the APs, while the edges represent their neighbourly distance. The graph can then be used to stand out APs groups and their "bridges", which is key to identify their real locations and which buildings they belong to using clustering algorithms.

Chapter 2

Related Work

This chapter addresses the state of the art on social behaviour analysis, user mobility and user identification through use of mobile devices. The related work is divided by its different types of approaches to mobile mobile device detection, such as usage of external applications, AP associations and geographic location coordinates.

A number of related research is focused on mobile device location by a omniscient observer prespective, using this information to predict human mobility. The related work is split according to its source of location information, which can be from the own device protection using an application to retrieve GPS coordinates [8, 9] or retrieving nearby access points [10–13], or using access records from a cellular or WiFi network in order to infer the devices' location. Table 2.1 shows a timeline of the following articles and publications.

2.1 App approaches

Tom Nicolai, Eiko Yoneki, Nils Behrens, and Holger Kenn, 2006[10], conducted a study where the prime objective was to detect social interactions between groups of people, at a conference in Japan. Their project, Wireless Rope, aimed to take a social approach and perspective to a controlled crowd and obtain results from that social context. The project uses an app installed on the phone of each participant recording events such as arrival and departure from the conference, passing by others or group interactions, coffe and lunch breaks, or others.

Despite sharing the same base and somewhat related goals, the results are attained through very different means. Also, the data is obtained from bluetooth traces, and an approach like that is

not feasible at all in a context where we want to know the user's location, not to mention very outdated by today's standards. Same applies for the intrusive application on the phone which is impossible to do on a passive scan and analysis.

2.2 Omniscient observer approaches

Wei-jen Hsu and Ahmed Helmy, 2007[11], published an article about user's behaviour on two university campuses through wireless network records. Hsu, Helmy and Dutta's goal was to present new ways to manage networks, network protocols (e.g. behaviour-aware protocols) and applications. They intended to capture trends on a user's behaviour such as user association to a access point, using a TRACE (Trace, Representation, Analysis, Clustering, Employment) approach, to then classify them into groups.

Hsu and Helmy's work has an interesting point - movement behaviuor patterning. This work shares similar points with this thesis, since that both use WLAN 802.11 AP records, in a passive way, to provide results from user movement. While this thesis focuses on identifying users on wireless records, the article focus on creating patterns with the same type of dataset.

In 2008, the authors of [14] identify users on an anonymized dataset of Netflix subscribers and their preferences in an attemp for privacy breaching. The dataset contains attributes for ratings, movies and shows watched for each anonymous user. The final results for user identification starts by defining what privacy breaching is in the context – what is the user identification, what is the objective and what scenarios compose a breach; then the de-anonymization algorithm is applied where the inputs are the large dataset and – as in our own work for the social events – baseline auxiliary information that is known to be true. The algorithm applies a score to each record based on correlated data between the large dataset and the auxiliary information; then a criterion is applied to determine if there is a match or not; finally the record selection selects the "best-guess" matched record by its highest hit probability.

Our work relates to this analysis, as both use a public anonymized dataset along with auxiliary information as input for an algorithm that ties anonymous users to their real identity. Both works show that privacy can be compromised and sensitive information can be known from publicly available data.

Marangaze Munhepe Mulhanga, Solange Rito Lima, and Paulo Carvalho, 2011[12], published an article in which presents a study they did on identifying and characterising patterns of user mobility behaviour, between April and June of the same year, at both Univesity of Minho (Portugal) and University of Vigo (Spain). Analyzing the network traces of Eduroam, the results contained AP coverage, traffic volume, total users connected, number of session and their average duration.

In this article no social interactions between people and locations were correlated, making only use of a social network's data to provide solely results on mobility and activeness on the LAN. Our work differs by establishing a relation between the information gathered from social networks and information from access records on a LAN, which can be used for research on human mobility patterns.

In A.B.M. Musa and Jakob Eriksson, 2012[13], the authors described a system that tracks unmodified smartphones based on WiFi detections. They performed their trial on a busy road with several Access Points (APs), which lead to results such as estimation of trajectories, analysis of number of devices per vendor; number of unique MAC addresses per day and hour.

Musa and Eriksson's approach shares some characteristics with the one presented on this thesis, but without the social networks data correlation. Using this approach users can not be identified by their social profile, friends or individual presence on campus.

In [15], 2013, the authors identify the uniqueness of each user's movement "fingerprint" by analyzing an extensive dataset of call records. Spatio-temporal points recorded by receiving or making calls create a sub-trajectory for a specific user that is analyzed for their number of traces to represent uniqueness on the set. Unique trajectories are specific to their user, so the number of points is directly correlated to their uniqueness factor – more points make bigger trajectories, thus more unique traces are found to identify the users. Spatial and temporal resolution also has a correlation to the uniqueness factor, as for the same number of points a resolution of more hours an antennas originate a greater uniqueness factor.

2.3 Geographic coordinates approaches

In [8] the authors present a study on Location-based Social Network (LSN), where GPS positions were obtained by a number of smartphones during two months. They analyzed data from

Brightkite, a LSN, from a two month window which included hundreds of thousands of location updates. The extracted mobility patterns contributed to classify the users into groups of mobility and clustering to and activity classified users into groups of user type.

Despite the mechanisms to maximize detection, this approach covers several points addressed by this thesis, such as trajectory estimation. However this estimation is made by walking or driving on a road and has no contact whatsoever with social network events such as posts or comments, so those devices' owners remain anonymous to the collector.

The authors of [16] show that anonymized datasets with real location values can't be made publicly available as it offers low security measures for the user's privacy. Analyzing a call detail record (CDR) dataset from a cellular provider in the US for three months the authors study the impact and risks of using different granularity on locations, locations distance and social information on public datasets based on how easier it provides user identification. In the end they offer possible solutions such for publishing data without compromising privacy, such as coarse location granularity (e.g. city scale) so the top locations for each user are harder to pinpoint; however, the price to pay is possibly butchering the utility of the published data.

In [9] the authors used the publicly available social network data to identify anonymous users on a CDR dataset which included 335 million calls made by 3 million different users during 47 days.

This approach from Cecaj, Mamei and Bicocchi has the same goal as this thesis does - extract a location from an anonymized dataset by using data from social networks and try to breach user privacy despite anonymization. My main difference is the source of data, Cecaj, et al. used CDRs from a cellular network and on this work we use access records from wireless networks AP.

TABLE 2.1 Timeline

2006	Tom Nicolai, Eiko Yoneki, Nils Behrens, and Holger Kenn – detection of social interactions through cellphone's bluetooth.
2007	Wei-jen Hsu and Ahmed Helmy – study of user's behaviour through wireless AP records.
2008	A. Narayanan and V. Shmatikov – Robust de-anonymization of large sparse datasets.
2009	Nan Li and Guanling Chen - study of location-based social networks.
2011	Marangaze Munhepe Mulhanga, Solange Rito Lima, and Paulo Carvalho – identification and characterization of user behavior in a wireless network. Hui Zang and Jean Bolot – Anonymization of location data does not work: a large-scale measurement study.
2012	A.B.M. Musa and Jakob Eriksson – tracking of unmodified smartphones through Wireless detections.
2013	Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen and Vincent D. Blondel - Unique in the Crowd: The privacy bounds of human mobility.
2014	Alket Cecaj, Marco Mamei, Nicola Bicocchi – identification of anonymous users from call records through social network information.

Chapter 3

Implementation

On this chapter we will present the implementation, addressing the already defined objectives by presenting the code developed, goals and gaps filled by each project and the structure of the entire solution.

We chose C# as the main language for the most part of the work on this thesis. The reasons for this choice were mainly ease to manage, having automatic garbage collection and for being object-oriented, which provides more confort and abstraction. The IDE and platform was Microsoft's Visual Studio 2013. Almost all data analysis, username identifications and algorithms to achieve that goal were developed in a C# solution with various projects, each one addressing a certain objective on the pipeline. One can observe that it follows a standard three-tier architecture, visible on Fig. 3.1, where the presentation layer only depends on the business layer. The business layer contains all the business logic associated to this solution and to all calls made by the presentation layer. The business layer has a dependency on the data access layer, in which the latter contains access to the database with the anonymized network records. It's arguable if the API projects should or not be part of the business layer, but in this context they are for simplification.

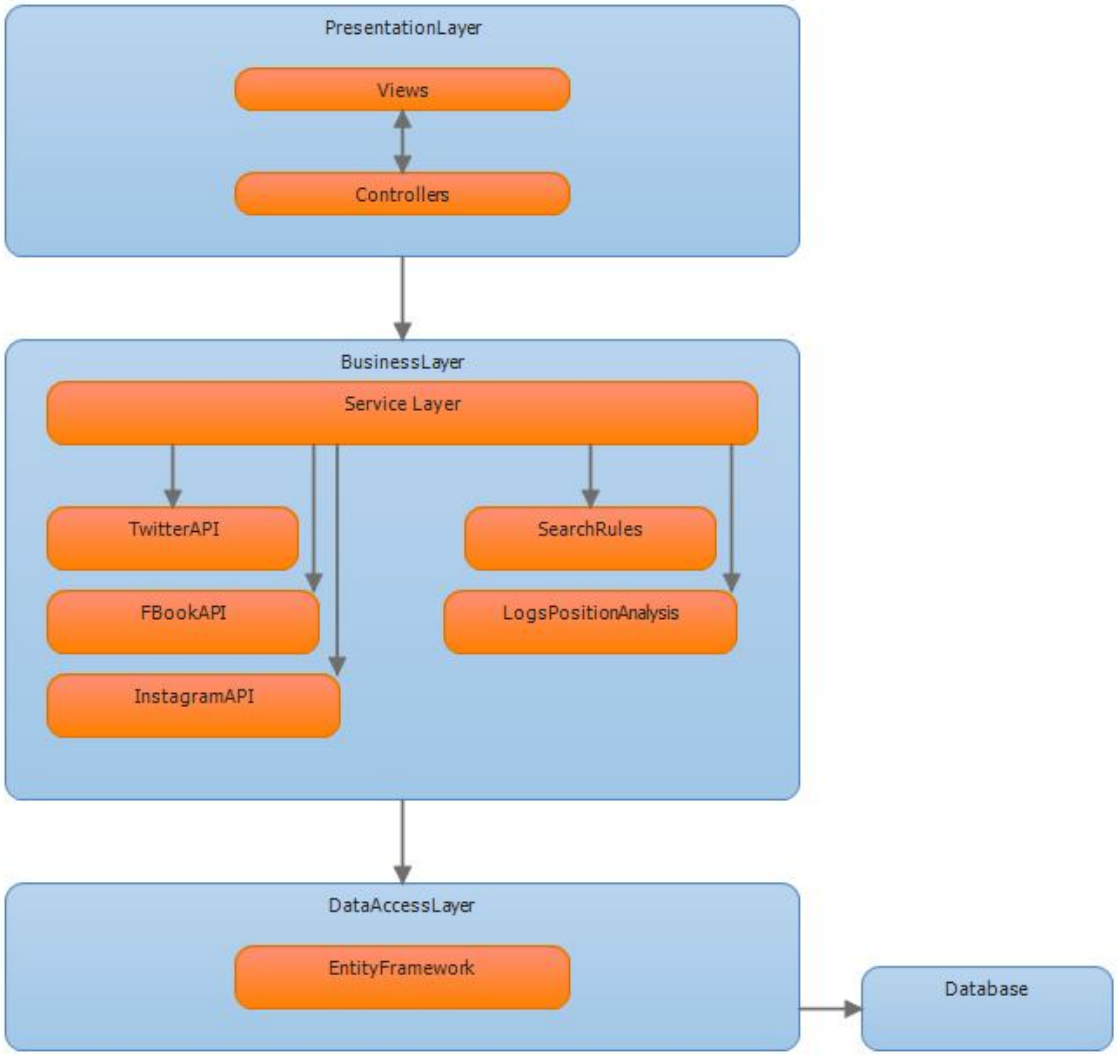


FIGURE 3.1: Solution's layer diagram

The layers are ordered from the more independent projects to more complex and coupled ones, which is the order they were developed, and are composed as follows:

1. Data access layer and access points (APs) records composition;
2. Social network application programming interfaces (APIs);
3. Business layer and main services;
4. Analyzing AP positions.
5. Searching rules and results from crossing AP records with social network data;
6. Presentation layer and graphic interface;

3.1 Data access layer and wireless records composition

The wireless records used on this thesis were obtained from the RADIUS server, which include all wireless access records generated by all the access points of the Eduroam network present at ISEL, from 1 May 2015 to 31 May 2015. The records were anonymized to the point where usernames, MAC addresses and access points cannot be identified or be tied to their real counterparts. The access records include 953462 connections, in total, made from the users' devices to ISEL APs.

3.1.1 Wireless records composition

Each access record is defined by a connection made from a device to a single AP, meaning that everytime a device establishes a new connection to a new AP, a new access record is created. The table has the following schema:

- "id" as a bigint. The identification on the database, the primary key;
- "UserName" as an int. The user on the network - its real value is the e-mail account from the user.
- "AcctStartTime" as a datetime. The time the connection started with this AP;
- "AcctUpdateTime" as a datetime. The time the connection ended with this AP;

- "AcctSessionTime" as an int. The duration of the connection with this AP, in seconds;
- "AcctInputOctets" as a bigint. Number of bytes received in the AP, during the connection, for the user;
- "AcctOutputOctets" as a bigint. Number of bytes sent from the AP, during the connection, for the user;
- "CalledStationId" as an int. The AP identifier - its real value is a MAC address, which is here anonymized as int;
- "CallingStationId" as an int. The user's device identifier - its real value is a MAC address, which is here anonymized as int;
- "WISPrLocationName" as an int. The AP location name - its real value is a descriptive string, which is here anonymized as int;
- "ClientIPAddress" as an int. The AP controller or the AP address.

Below is presented an example of some rows on the database, which are a part of the AP wireless records used on this project:

	id	UserName	AcctStartTime	AcctUpdateTime	AcctSessionTime	AcctInputOctets	AcctOutputOctets	CalledStationId	CallingStationId	WISPrLocationName	ClientIPAddress
1	1131820	1314	2015-05-01 09:47:31.000	2015-05-01 09:49:50.000	139	32366	55654	4	4288	2	41
2	1131821	1080	2015-05-01 11:12:19.000	2015-05-01 11:14:44.000	145	1528	1247	4	4282	2	41
3	1131822	1346	2015-05-01 12:16:35.000	2015-05-01 12:16:36.000	1	48	90	4	4890	2	41
4	1131823	1629	2015-05-01 13:21:06.000	2015-05-01 13:21:21.000	15	49000	176486	4	5043	2	41

FIGURE 3.2: Example of the first rows of the anonymized AP records

As can be seen, the first row contains a connection made by the Username 1314 on the AP 4 by the device 4288. It started at 09:47:31 and remained connected at that AP for two minutes and nineteen seconds, before disconnecting or getting closer to another AP on the network.

3.1.2 Data access layer structure

The server chosen to hold this database was Microsoft's SQL Server 2014. This choice is justified by it's simplicity to use with the chosen framework to communicate with the database, as well as with the chosen IDE since the entire .NET ecosystem is built to work together. A table named "anon_radius2015" was created, which would have the thousands of hundreds of rows.

The data access layer consists of a project, contained in the solution, that uses Entity Framework 4 with ADO.NET Entity Data Model to recreate the table "anon_radius2015" in a C# object. This object, is automatically created, along with a connection to the database server, allowing other projects that reference it to make use of the Entity Model for accessing the access records. Figure 3.3 shows the basic object arquiteture of the data access layer project.

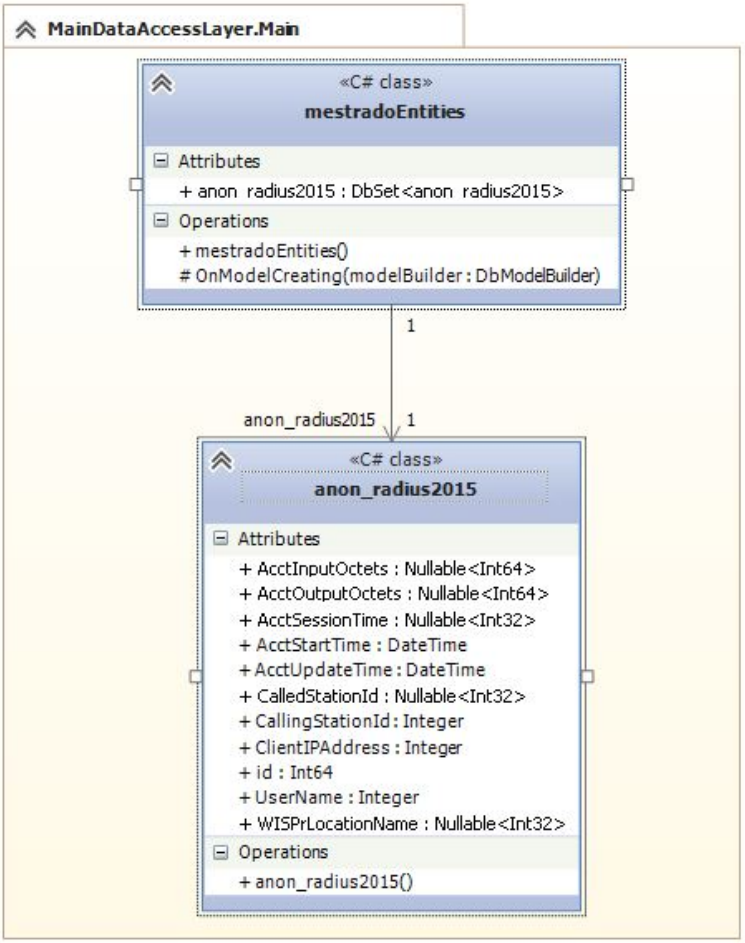


FIGURE 3.3: Data access layer - UML

The choice to use Entity Framework instead of any open source solution was simply because it synergies really well with both Visual Studio and SQL Server, since they're all from the same provider.

3.2 Social network application programming interfaces (APIs)

In the beginning of this thesis there was a relevant question that had to be answered before the implementation itself began: *"What social networks are going to be useful?"*. The first approach was to use Instagram, Faceebook and Twitter, since all provide check-in posts with GPS coordinates as a location attached to that post, and that would be key to an easier identification of a user on the anonymous records.

This section discusses all three APIs, developed on the business layer, and why was Twitter the chosen one and others discarded.

3.2.1 Facebook API

Facebook is the most widely used social network in Portugal and was the first social network that we used. Its REST API[1] facilitates greatly development of solutions that make use of it.

The Graph API is called through HTTP Requests, both GET and POST, to a variety of searches like groups, pages or users, depending on the HTTP URL. Figure 3.4presents the project structure for the API developed to make Facebook Graph calls.

Both classes from the `Searches` package classes are responsible for searches - one for Facebook groups, other for Facebook pages - doing so by making HTTP Requests to the respective URL. The response comes in JSON[2] format, which is parseable by each of these classes and contains the posts on the page, their comments and users associated. The search objects, visible in Fig. 3.4, can both return a collection of posts or events, from the package Entities. An event in this context is not directly associated with Facebook, being instead an occurence of a comment or post - it contains only the user and date, which despite being important, still miss the GPS coordinates rendering this method unusable.

However, this API ended up unused on this thesis because of Facebook privacy reasons. The current Facebook Graph API version 2.2 doesn't support obtaining the feed of user profiles without those users accepting the application beforehand and that's unconvenient in scenarios where a large population is under study. Before dropping this source a test was made on ISEL public groups, but without GPS coordinates associated to the posts or comments and lack of

[1] https://developers.facebook.com/tools/explorer
[2] http://json.org/

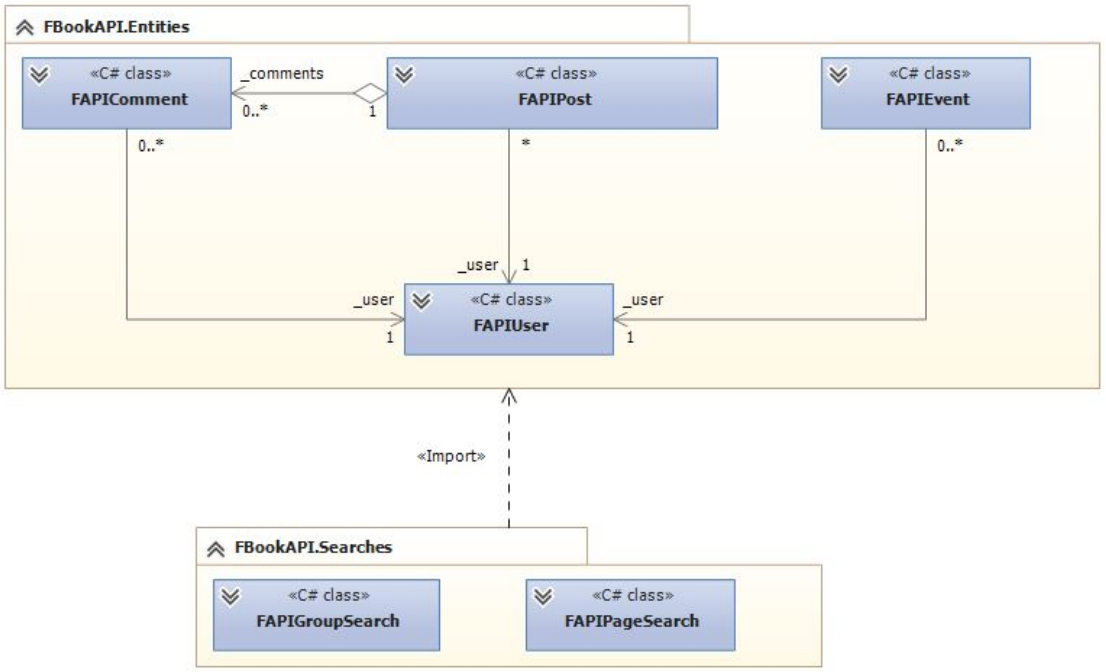


FIGURE 3.4: Facebook API - UML

activity in those groups there wasn't sufficient information to connect at least a Facebook user to a user at ISEL.

3.2.2 Instagram API

Instagram, despite having a lesser penetration in Portugal than Facebook, has less restrictions on search usage. It has a REST API [3] that can provide user's feed, posts within a radius of GPS coordinates and posts in a location name. The location name is the most interesting part, because Instagram uses Facebook's registred locations[4], therefore searches with locations "ISEL - Instituto Superior de Engenharia de Lisboa" , "ISEL" or "Biblioteca Do Isel" work and deliver posts from those locations. The API works the same as Facebook's regarding HTTP GET and POST: requests for a certain URL with some parameters leads to a response in JSON with the requested data. However the collected data is richer, containing information such as Instagram users tagged in the photo which hints at their location at the time.

An API wrapper project was developed and its structure can be seen in Fig. 3.5, presenting the search methods and the Instagram entities.

[3]Instagram API Documentation - https://instagram.com/developer/endpoints/locations/
[4]https://instagram.com/developer/endpoints/locations/

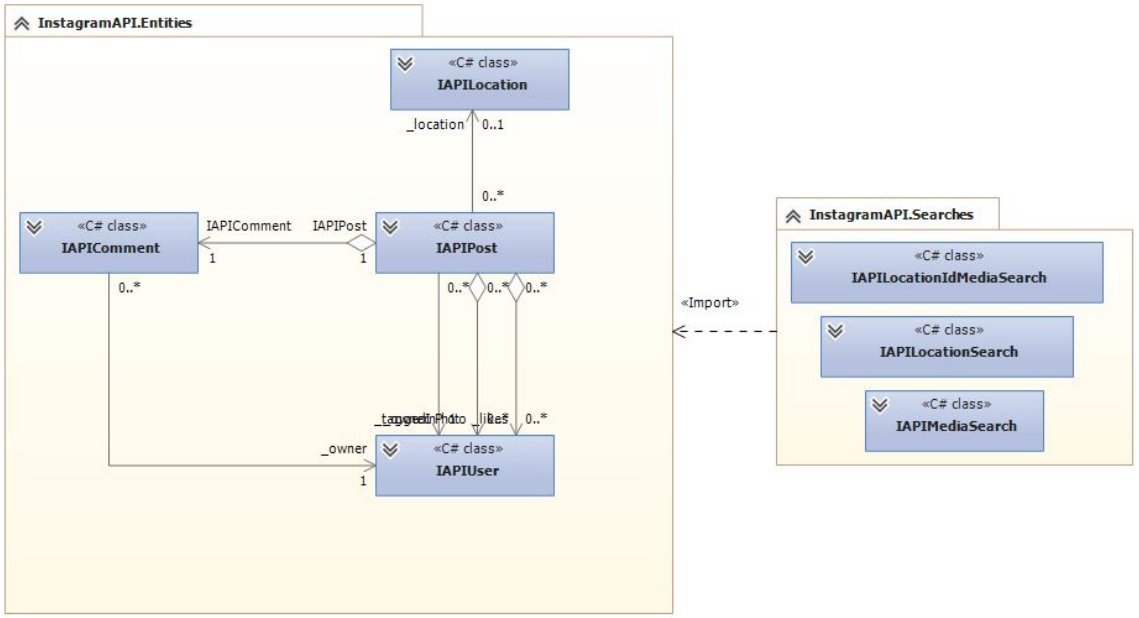


FIGURE 3.5: Instagram API - UML

It provides searches for location names and ID from GPS coordinates - for example, the coordinate representing the center of the ISEL campus and its radius -, used for the search query that returns posts from the desired location. The response is returned in the format of posts, that include users who liked the photo, users tagged in the photo, comments and more.

This API is not used on this thesis due to the lower usage when compared with other social networks that could benefit more this work.

3.2.3 Twitter API

Twitter was the first option after Facebook got opted out, based on the fact that it has the most user penetration after Facebook and has great GPS integration into posts from mobile devices. It has an API[5] implemented using REST and just like Instagram's, that can provide posts for given GPS coordinates and radius, as well as an entire feed for a given user.

Its API was adapted to be used by the solution, and the arquitecture is presented in Fig. 3.6.

In constrast to the previous social network APIs, this one was not made from the ground up, but simply adapted to fit the needs through the utilization of a wrapper pattern on linvi's Tweetinvi library[6]. It contains two possible search objets: one for timelines searches from a specific user

[5]https://dev.twitter.com/rest/tools/console
[6]https://tweetinvi.codeplex.com/

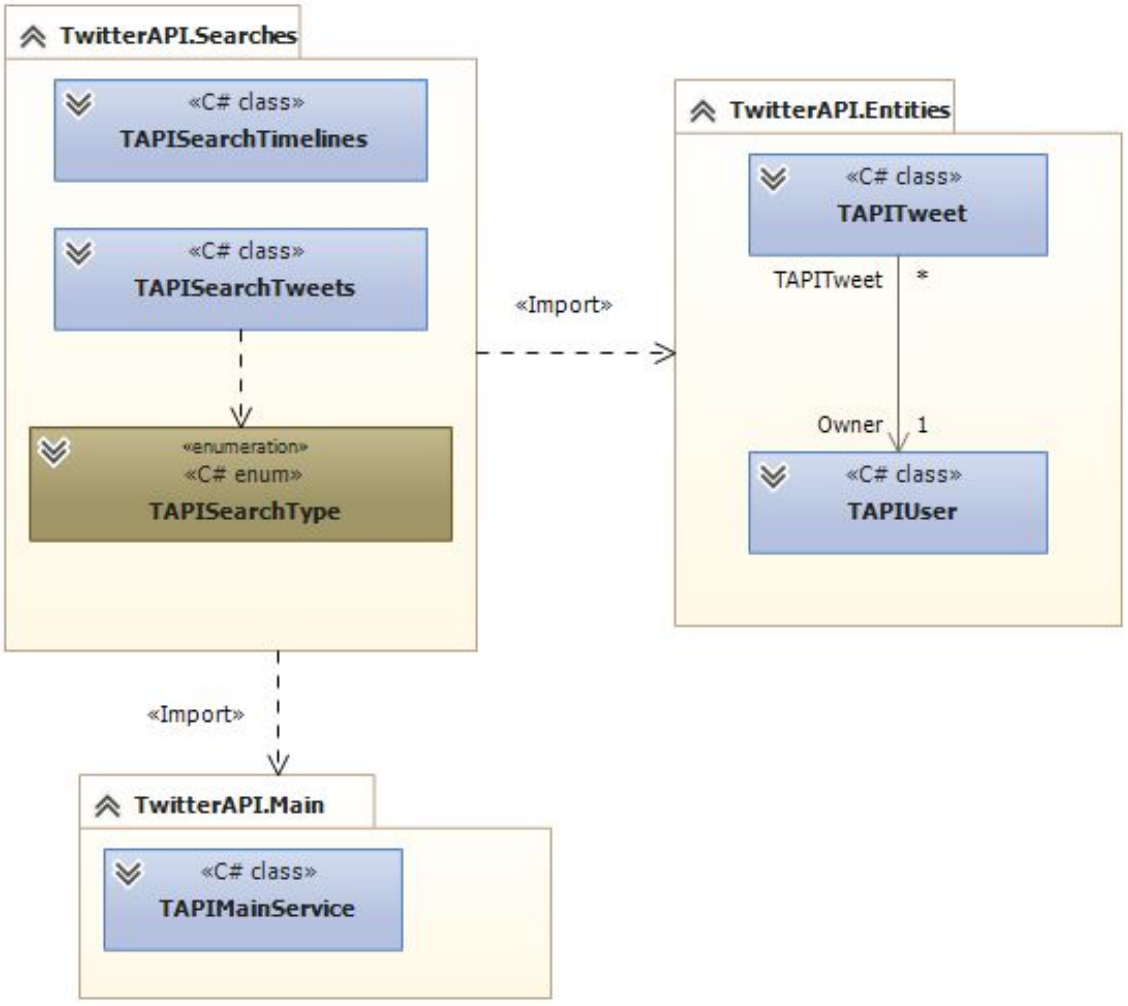


FIGURE 3.6: Twitter API - UML

ID or user name; and another for post searches using a set of parameters such as GPS coordinates and radius, specific time window and search type - recent or popular posts. Results include a collection of tweets that contain the text, the owner and the GPS coordinates when available.

Twitter ended up being the chosen social network to gather info and achieve the goals set before, for popularity and privacy reasons.

3.3 Business layer and main services

The business layer was developed to serve as a middleware between the presentation layer and the social network APIs. While theoretically both social network APIs and services belong on the same business layer, they are in fact separate code projects.

Services are what the presentation layer calls and work as wrappers for the APIs, exporting only the intended methods calls. Fig. 3.7 shows the UML for this project, presenting the different services that can be called from any external binary.

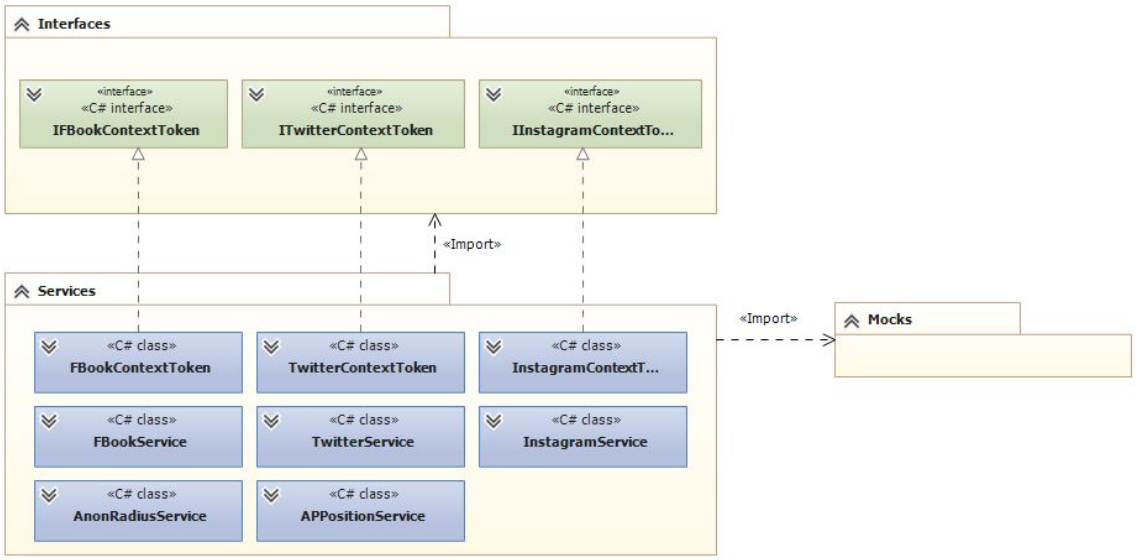


FIGURE 3.7: Business layer - UML

This layer exports five main services: Facebook, Instagram, Twitter, AnonRadius and APPosition services. All services work as a singleton instance of themselves, keeping context for each different caller. The mocks work like adapters to classes from other assemblies - it mocks the other classes so this layer can return, for example, Facebook posts instances to avoid excess dependencies.

3.3.1 Social network services

The social network services are services that can be called by projects that import the presentation layer. The services are used as singleton and return a special token that is used for context; the token contains, in most cases, the URL for the requests. Each service contains different methods and they have to be called with the context as parameter. The Facebook service provides methods to get posts and comments from a public group or page; the Instagram service provides methods to get posts from geographic coordinates, get location IDs from geographic coordinates and get posts from location IDs; the Twitter service provides methods to get tweets from a user, tweets from a geographic location and radius and time window. Additionally to

fetching data from the social network, all services provide a method to identify users on the network.

All three social network services were implemented, but only the Twitter service was used due to reasons mentioned before.

3.3.2 AnonRadius service

The purpose of this service is to allow database interactions by the caller project. It provides methods for single tuple insertion, single tuple read and read all from the table anon_radius2015, where the AP records are.

This service is used for the insertion of all the rows on the anon_radius2015 table - which is done only once - and for further tuple reads. The tuples are inserted via the service since the raw access logs were in MySQL format and the service handles the parsing and data conversion.

3.3.3 AP Position service

This service works as a wrapper to the component that is responsible for calculating the distances between APs, position them in 2D space and then clustering and mapping them accordingly to their position. This is the bridge that provides distinction between being able to identify users based on only their presence on the network and using geographic positioning.

The component this service calls is detailed in the section 3.4.

3.4 Analyzing AP positions

The association of a social network user to an eduroam user is possible by using multiple approaches. The simplest one is by crossing only time references, associating a social network user to Eduroam users that were connected at the same time. The correct association is thus dependent on the number of users at both data sources. The difficulty on finding a user increases accordingly with the number of users present on both datasets.

To improve the abililty of associating a user from the social network to one on the eduroam network, we needed to reduce the number of possible associations. To do so, we opted by crafting

an algorithm to infer the position of the APs and associate them to their real position. With geographic location from both events and APs it's possible to narrow down the list for each event presence to buildings or even rooms, instead of the whole network. That means, theoretically, that anonymity can be broken with even less events on the social network if geographic information is taken into account.

This project provides an interface for locating APs in space. Fig. 3.8 shows the architecture used to develop this component and presents the main functions and the different distance approaches.

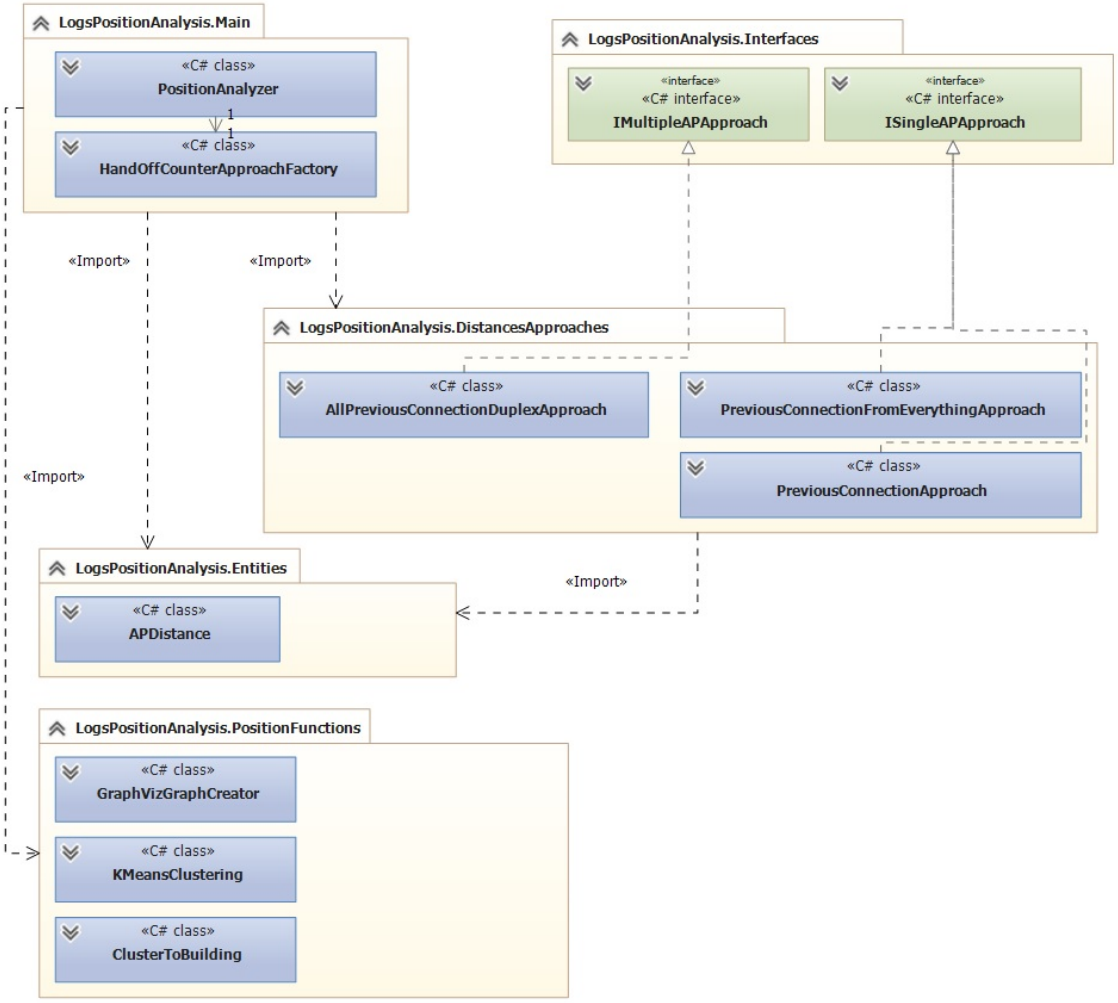


FIGURE 3.8: AP positions analysis - UML

The functions exported by this binary component work as a three-phase pipeline:

1. The entire Eduroam wireless access records are used to calculate the distance between each AP, creating an undirected graph with all APs in a relative 2D space;

2. Isolate AP groups using their graph distance and proximity;

3. Assign a building name to each group based on the real relative buildings distance.

3.4.1 First phase – Creating the graph and extracting space coordinates

The first function provided by this component consists of using all AP records to calculate the probable distance between each pair of APs and create an undirected graph using the APs as nodes and distances as weighted edges.

To determine the distance between nodes, we used the number of handofs between APs. "Handoff" or "Handover" occur when a device on the network switches its wireless connection to a new AP. This is usually justified when the device determines that the second AP is a better candidate for providing the WiFi connection. This typically occurs when a user moves in range of the second. The handoffs are used to detect that a user is moving. To achieve this, we determine the time difference between the ending of a connection and the start of a new one. We consider that this time difference corresponds to the distance between those two APs. We consider that all users move at about the same speed, and that the connections from users travelling on cars around the campus are residual.

We implemented multiple approaches to infer the distance between two APs, and while all calculate the distance, they have some differences between them.

- `PreviousConnectionApproach` – to calculate the time distance between AP X and Y, this approach gets from the database all rows where the connection is with AP Y. It runs through all of them in a cycle, and on each iteration it gets all connections that ended before the start of this specific connection, from that username with that device MAC address, ordered descending by starting time. The first connection from the query is then determined if it was on AP X - if it was, then the handover went X to Y, and the difference time is calculated and added to the total time. The algorithm is described in pseudocode in Alg. 1.

- `PreviousConnectionFromEverythingApproach` – this approach gets from the database all rows where the connection is with AP X or Y and orders them by username, device MAC address and connection start time. So, cycling through all of these resulting connections, if that connection is to AP Y and the connection on the list was to AP X, to the same username and device address, then the difference time is calculated and added.

This approach is considered the weakest because, since it only lists the connections to APs X and Y, it doesn't assure that the same user and device haven't gone through other APs in between them. The algorithm is described in pseudocode in Alg. 2.

- `AllPreviousConnectionsDuplexApproach` – this approach is the quickest to calculate, because it lists all connections from the database, ordered by username, device address and connection start time. While iterating through all the previously ordered dataset, this approach checks if the previous connection was from the same username, device address. The handoff time is calculated from the previous connection AP X to the current connnection to AP Y, and stored in an object that contains all X-Y times added and number of handoffs between them. In the end, all pairs X-Y and Y-X are combined, times summed and number of handoffs, so there's not distinction between going from X to Y and Y to X. This approach is the strongest because it calculates the time in duplex mode and it's the quickest, with only one database request.

Algorithm 1 PreviousConnectionApproach

1: $timeSum \leftarrow 0$
2: $sampleCount \leftarrow 0$
3: $tableY \leftarrow$ from $database$ where $row.AP$ = AP_Y
4: **for all** $rowY$ in $tableY$ **do**
5: $previousConnection \leftarrow$ (from $database$ where $row.Username = rowY.Username$ **and** $row.AP = AP\_Y$ **and** $row.ConnectionEnd <= rowY.ConnectionStart$ orderby $rows.AcctStartTime$ descending).First()
6: **if** $previousConnection.AP = AP\_X$ **then**
7: $timeDifference \leftarrow rowY.ConnectionStart - previousConnection.ConnectionEnd$
8: **if** $timeDifference <= threshold$ **then**
9: $timeSum \leftarrow timeSum + timeDifference$
10: $sampleCount++$
11: **end if**
12: **end if**
13: **end for**
14: **return** $timeSum/sampleCount, sampleCount$

All approaches end up with a time from AP X to AP Y as the average time between all handoffs from each pair of APs. Pairs with handoffs above a certain threshold of time distance are discarded, so scenarios like disconnecting a device and connecting it on the following day to another AP isn't considered in the results. This threshold was used as 240 seconds as it was considered to be the average time between the most distant APs on campus. Another filter is passed as pairs that have number of samples below a determined threshold are discarded too.

Algorithm 2 PreviousConnectionFromEverythingApproach

1: $timeSum \leftarrow 0$
2: $sampleCount \leftarrow 0$
3: $tableAllConnectionsXY \leftarrow$ from $database$ where $row.AP$ = AP_Y **or** $row.AP$ = AP_X orderby $row.UserName, row.CallingStationId, row.AcctStartTime$
4: $prevRow \leftarrow null$
5: **for all** $rowXY$ in $tableAllConnectionsXY$ **do**
6: **if** $prevRow <> null$ **and** $rowXY.Username = prevRow.Username$ **and** $prevRow.AP = AP\_X$ **and** $rowXY.AP = AP\_Y$ **and** $prevRow.ConnectionEnd <= rowXY.ConnectionStart$ **then**
7: $timeDifference \leftarrow rowXY.ConnectionStart - prevRow.ConnectionEnd$
8: **if** $timeDifference <= threshold$ **then**
9: $timeSum \leftarrow timeSum + timeDifference$
10: $sampleCount++$
11: **end if**
12: **end if**
13: $prevRow \leftarrow rowXY$
14: **end for**
15: **return** $timeSum/sampleCount, sampleCount$

This assures that too few samples don't make a solid average as more population means the average is not skewed by less common events such as two distant APs having low sample numbers with low time distance. This threshold was used as 170 minimum samples, as after several tests using both pair data and Graphviz (as discussed ahead) it was determined that a number lower than that would allow for weak and not viable pair time distances and a number much higher than that would break pair connections that actually had really close distance between them, but not that high number of samples.

The return from each approach is a collection of pairwise distances between APs.

A Graphviz[7] file was created, where nodes are the APs and the edges are their previously calculated pairwise distances. That file is opened in Graphviz and the graph is created by Neato, a layout engine that interprets weight parameters for the edges. The graph is exported to a plain-text format, where each node and edge contain their own 2D space coordinates, which is used as input for the following phase.

Figure 3.9 shows the created graph for the APs in a 2D space in relative distance.

[7] Graphviz - http://www.graphviz.org/

FIGURE 3.9: AP spatial graph

3.4.2 Second phase – Grouping APs

On this phase APs are grouped into clusters based on the space coordinates previously calculated.

A K-Means clustering algorithm[17, 18] is used for this phase as it is one of the simplest and robust way to cluster points in a *n*-dimensional space since it's only needed to specify the number of clusters to form (the *k* in the name) and the spatial graph. It's efficient iterations allow to have a $O(kodi)$ cost, where *k* is clusters, *o* is nodes, *d* is spatial dimensions and *i* is number of algorithm iterations until ideal return. Various other cluster models were pondered for use

such as connectivity or distribution models, but the centroid model was determined as the most straightforward and easy to understand. More clustering approaches were experimented, as described in 3.4.4.3.

This algorithm returns a list of integer lists – a list of all final clusters. The algorithm groups all nodes (APs) into k sets of points, where k is an input parameter. It starts by selecting k random nodes from the graph that work as seeds. It then assigns every other node to the closest seed, based on a simple two dimension euclidian distance. Spatial mean nodes are calculated for each node group and the assignment is repeated using the new means as if they were the initial seeds. The process is repeated a set number of times, converging each iteration to the optimal node clustering.

Optimally, the clusters passed as parameter represent the number of buildings to map the APs to. For ISEL Campus 8 clusters was the chosen parameter.

3.4.3 Third phase – Mapping clusters to real buildings

After getting the clusters from the previous phase, it is necessary to assign a cluster to a building, identifying where the APs really are. For this we need to create a set of rules that are specific for each solution.

This project implements the solution for the ISEL Campus only as a method, whereas other environments would need another specific set of rules. The clusters and AP pairwise distance collection work as input for this method, as it compares clusters with each others to differentiate what buildings are assigned to the clusters. Fig. 3.10 shows the different buildings, where each color means an expected cluster. Each color will be treated as a single building as small buildings were conjoined with the biggest closest building. This algorithm assigns a name to each different building, and the following legend represents the colors on Fig. 3.10: dark blue is building “F”, purple is building “C”, red is building “G”, orange is building “A”, yellow is building “P”, light blue is building “M”, dark green is building “E” and light green is building “RMB”.

Our implementation for ISEL starts by calculating and mapping the percentage of handoffs that occur between each cluster based on the AP pairwise handoff list, e.g. how many APs from cluster 1 have handoffs to APs in cluster 2 - that roughly says how “close” the clusters (that we can map to buildings) really are. Table 3.1 presents the handoff percentage between APs from

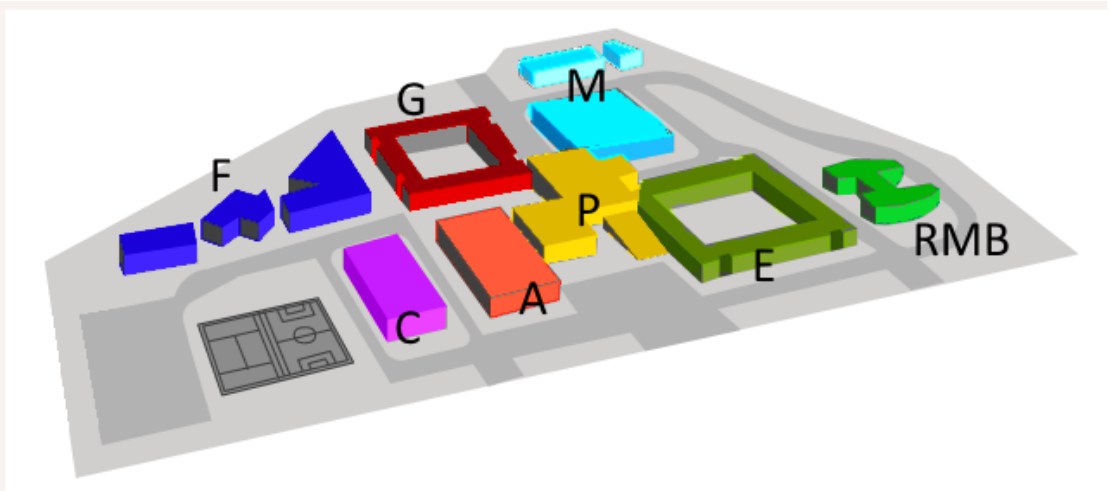


FIGURE 3.10: ISEL Campus buildings

TABLE 3.1: Handoff percentage between APs from each cluster

	cluster 1	**cluster 2**	**cluster 3**	**cluster 4**	**cluster 5**	**cluster 6**	**cluster 7**	**cluster 8**
cluster 1		29%	53%	47%	0%	20%	0%	0%
cluster 2	38%		80%	23%	0%	13%	71%	66%
cluster 3	38%	88%		47%	0%	73%	42%	11%
cluster 4	23%	23%	60%		0%	40%	0%	11%
cluster 5	0%	0%	0%	0%		20%	0%	0%
cluster 6	15%	23%	66%	29%	16%		14%	0%
cluster 7	0%	47%	26%	0%	0%	6%		66%
cluster 8	0%	47%	6%	5%	0%	0%	21%	

each cluster. The representation is as follows: 38% of APs from cluster 1 contain handoffs to APs from cluster 2; this is unidirectional since only 29% of cluster 2 APs have handoffs to cluster 1. The percentage thresholds used on the algorithm described further down were determined by analyzing this table, the probable cluster locations and real buildings locations, so they're a product of study beforehand and not applicable to a general solution for other campuses or contexts.

The next stage consists of identifying what we call the "fuzzy clusters", which are clusters that have a high percentage of handoffs to all other clusters thus being hard to pinpoint the exact building and remaining unidentified as the error margin token. Those clusters were identified for having 5 or more instances of more than 20% handoff, which implies being either on the center of the graph or being composed by APs from lots of different buildings. The clusters 2 and 3 were identified as being fuzzy clusters. Figure 3.11 presents the AP graph with the APs belonging to the fuzzy clusters as grey.

FIGURE 3.11: AP spatial graph - marked fuzzy clusters

We try to identify the RMB building next, for being the most isolated, thus easier to identify. A quick glance at the handoffs table would tell us that it's probably cluster 5, so the algorithm searches for the cluster with low (<20%) handoff average or no handoffs at all for other clusters. The cluster was identified as being the RMB building. Figure 3.12 presents the AP graph with the RMB APs shown with light green coloring.

Since we'll need to know where the other buildings are before identifying M or E, the following procedure identifies A, C, F, P and G. It starts by trying to identify the cencer group of neighbours, where those 5 clusters are – neighbours of X are clusters that X has more than 15% handoff to. When a cluster respects that rule, it is checked if all those neighbours are neighbours

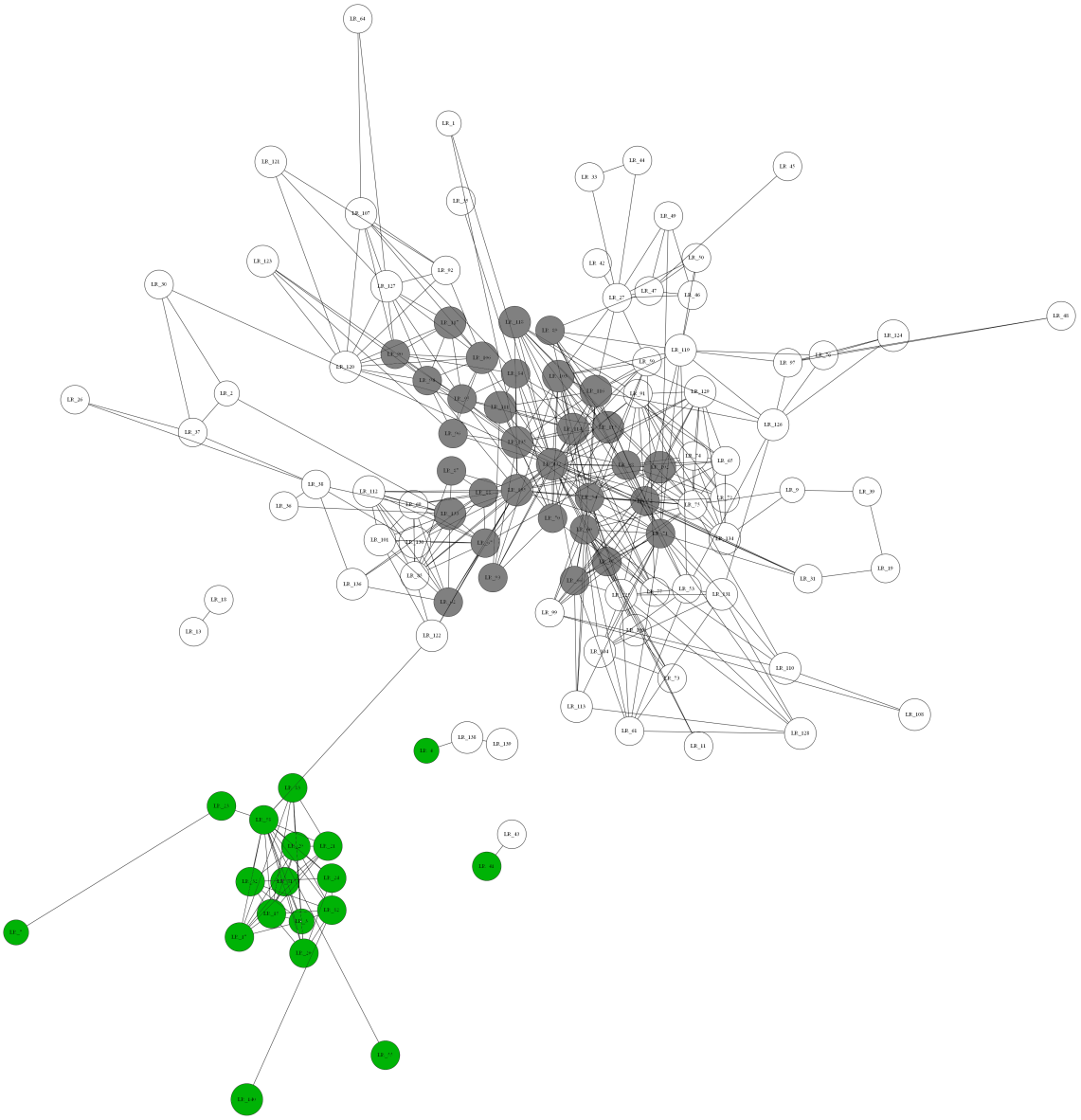


FIGURE 3.12: AP spatial graph - marked fuzzy clusters and RMB cluster

of each other's. If the cluster's neighbours are each other's neighbours, then it is assumed that group is the center of campus and those clusters are A, C, F, P and G. Iterating only the group, we'll try to identify from the F's perspective. Knowing that geographically G and C are the closest to F, we start by searching which cluster – in the neighbours list – has more than 40% handoff to a neighbour cluster and less than 20% to another neighbour. The cluster which respects that is F, while the neighbour with better handoff we consider to be G, the lower to be C since the F's entrance facilitates much more the handoffs to the G building. However, the neighbour list only consisted of 3 clusters, instead of 5. The other 2 were the fuzzy clusters, so it's assumed most of P and A APs are inside those fuzzy clusters, so no identification for those two buildings is

made. Cluster 1 was identified as being C, cluster 4 as G and cluster 6 as F. Figure 3.13 presents the AP graph with the C APs colored as pink, G APs colored as red and F APs colored blue.

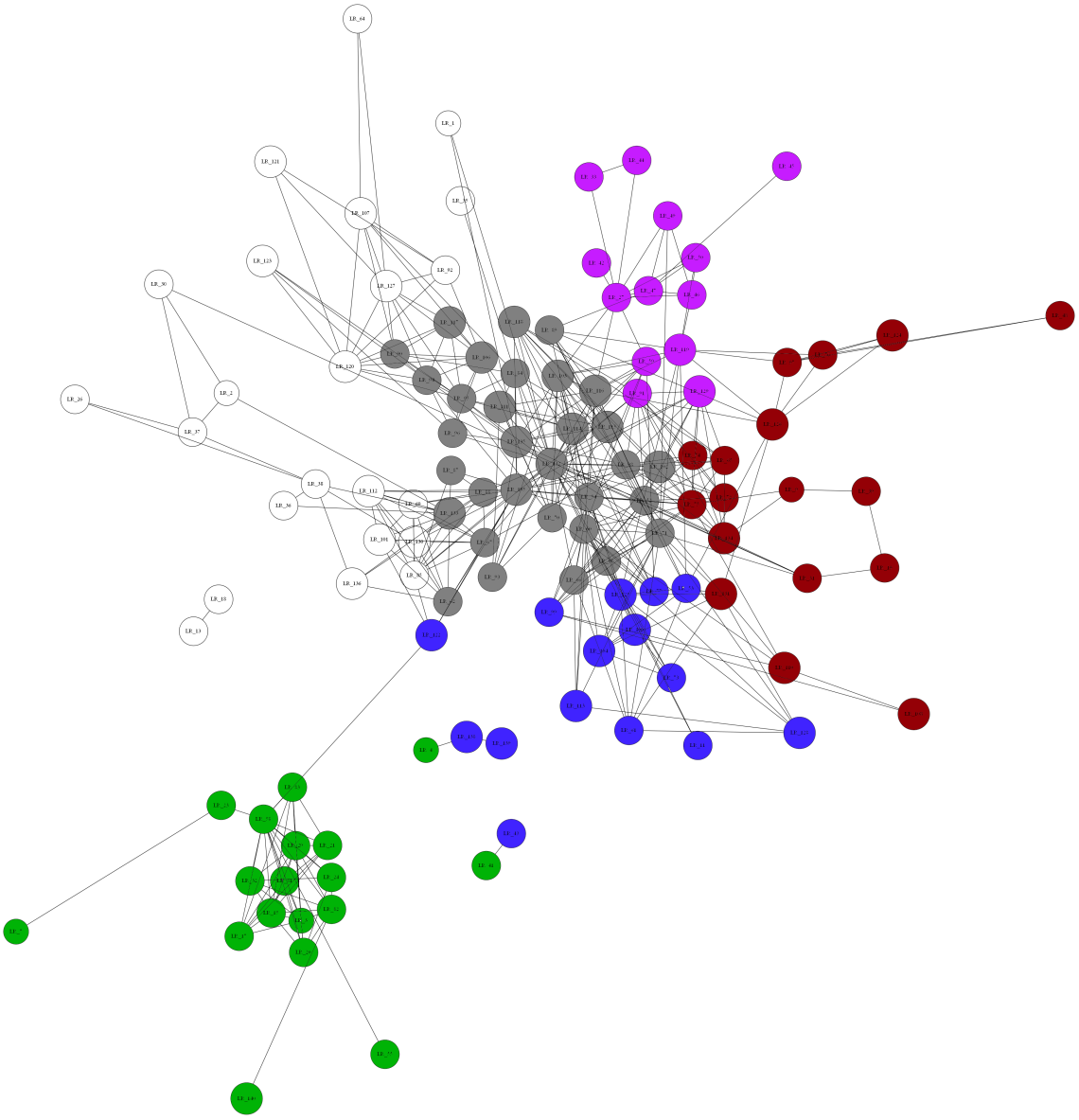


FIGURE 3.13: AP spatial graph - marked fuzzy clusters, RMB, C, F and G clusters

In the end only 2 clusters are left identified. M and E are distinguished by their surroundings – M is closer to G and F, while E is closer to P or A. The algorithm searches for those specific handoffs knowing already which buildings belong to each clusters. Knowing also that the fuzzy clusters are mostly A and P, the algorithm checks which one of those clusters is closer to F and G. Then it searches for one of the two clusters that is closer to that fuzzy cluster neighbour of F and G. That cluster is M, while the remainder is E. Figure 3.14 presents the AP graph with all

clusters now properly identified, with the M cluster showing as light blue coloring and E cluster as dark green.

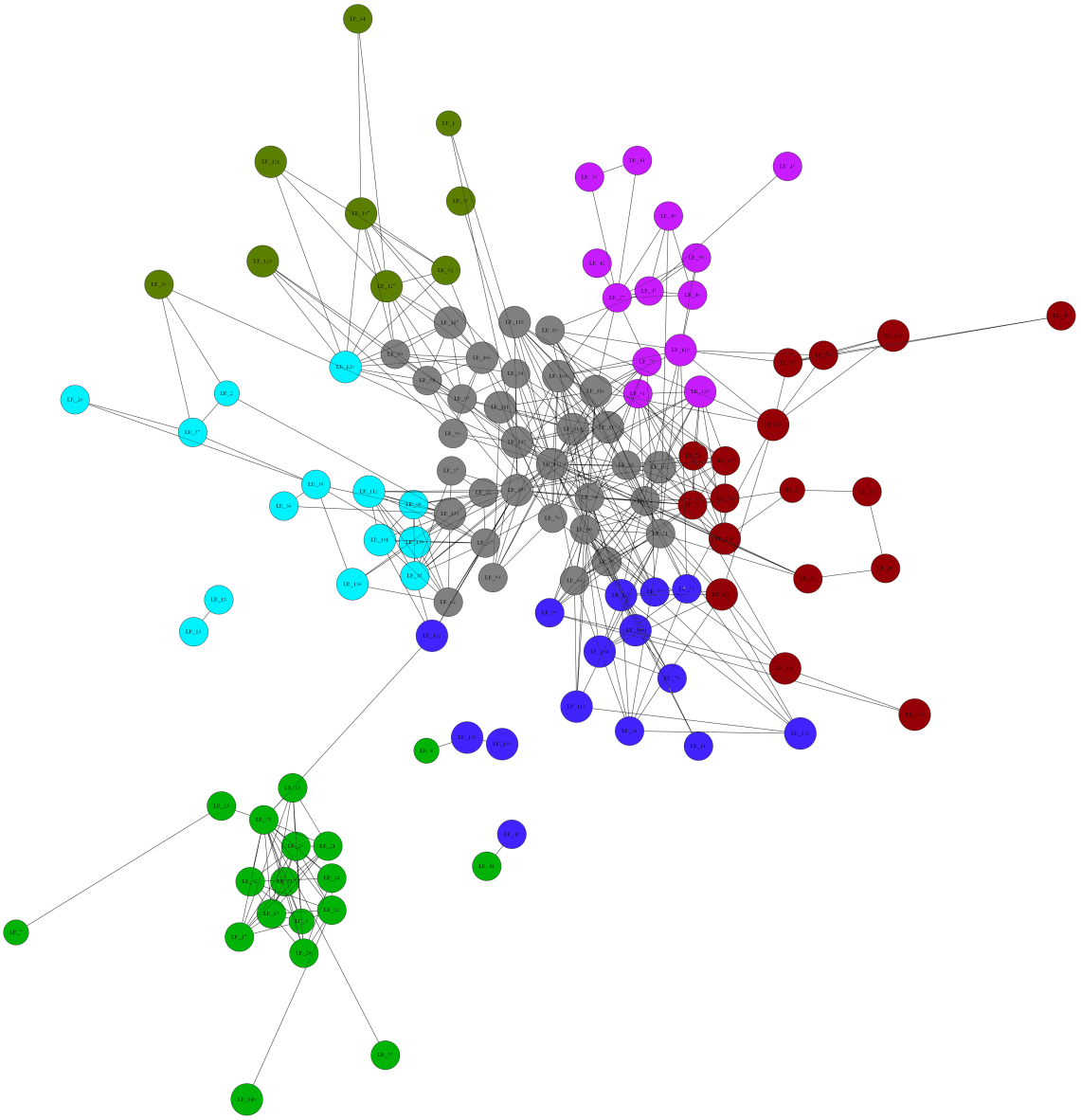


FIGURE 3.14: AP spatial graph - all clusters marked

This function returns a collection containing association between clusters and buildings, or if it's a fuzzy cluster. That information is then used on the user identification process.

3.4.4 Different approaches to clustering APs

Before settling with K-Means doing the spatial clustering based on 2D coordinates, a multiple number of algorithms were evaluated. After having the distances between the APs, we were

able to determine the spread and relative positions between APs. However, two issues arrised:

1. Not all possible APs pair combination have a listed distance between them;
2. The distances between each APs pair are not mathematically correct.

Many approaches were taken into consideration and developed, although all proved to be incapable of determining the correct AP positions due to these issues. Some of the taken approaches will be explained as follows.

3.4.4.1 Mathematical approaches

The first approach was to use Accord.NET C# Framework[8] to, with a mathematical solution, get a distance matrix D_{ij} where i and j represent the APs and D the distance between them. This is a symmetrical square matrix whose diagonal elements are all equal to zero. From the distance matrix a Gramian matrix M can be obtained, whose vectors are of an Euclidian space, with the following definition:

$$M_{ij} = \frac{D_{1j}^2 + D_{i1}^2 - D_{ij}^2}{2}$$

Now, using the framework it's possible to get the eigenvalue decomposition of M, namely eigenvectors U and eigenvalues S, and knowing that every symmetrix matrix can be written $M = USU^{\text{T}}$, then the matrix $X = U\sqrt{S}$ gives the of points in each of its rows. However the solution ended up with nulled values because the first and second problem mentioned above. The distances were not perfect euclidian distances, just estimates and there were missing pair values, which were randomly filled. This solution was discarded.

The second approach was using multidimensional scaling-type function t-SNE[9] (t-distributed Stochastic Neighbor Embedding). For this approach, the distance matrix was actually exported to a text file and then imported on matlab, where the library was developed. That matrix would be used as a parameter to the t-SNE function. The fact that t-SNE is actually a probability distribution function with a random seed and added the fact that both problems still reflect here,

[8] http://accord-framework.net
[9] http://lvdmaaten.github.io/tsne/#implementations

the results would vary in every execution and the final coordinates didn't even respect the passed distances, which made this approach unusable.

The third approach was using the multidimensional scaling function (cmdscale) from matlab[10]. It uses the exported distance matrix, as t-SNE, and produces a matrix that contains the reconstructed points. However, given the two problems mentioned before, filling the missing pair distances differently seemed to lead to different results.

3.4.4.2 Force directed graph drawing approaches

Algebra solutions failed on the previously mentioned issued missing distance values and not having acceptable euclidian distances. The next step was to use force directed graph drawing, which act by having a repulsion between each pair of nodes. Through machine learning algorithms, the result should converge for the optimal graph based on input.

The first approach consisted of using Satsuma[11], a .NET graph library. This library contains various classes for graph drawing, including force directed graphs. A node was created for every AP and arcs between those who have a distance calculated. However, upon testing and exploring the framework with the nodes as the APs, the results were not satisfactory, as the arcs couldn't be assigned a force and. That means that, through simulated annealing, the spring and electric force only act on the actual distance between the nodes, so the attraction and repulsion, respectively, cannot be measured by their real distances. This approach was discarded because of these reasons.

The second approach was made from a 2D/3D force directed graph algorithm demo called Ep Force Directed Graph.cs[12]. This library offered a graphic user interface that allowed to add nodes and create lengthened arcs on-the-fly, also adjustable damping, repulsion and stiffness that applied to all nodes equally. Added nodes can be seen drifting away from each other respecting the arc's length until a converging point is achieved. However, different damping and repulsion lead to different results, so the inconsistency based on input parameters wasn't really reliable for a solution that tends to be optimal.

[10] http://www.mathworks.com/help/stats/multidimensional-scaling.html
[11] http://satsumagraph.sourceforge.net/
[12] https://github.com/juhgiyo/EpForceDirectedGraph.cs

3.4.4.3 Other clustering approaches

Before settling with K-Means, other clustering algorithms were tested. The Java JUNG Framework[13] was used and tested, being a framework that deals with graph oriented problems with clustering algorithms:

The first used algorithm was a Edge Betweenness clustering algorithm (also called Girvan–Newman clustering) that iteratively removes edges on the graph based on their "betweenness" in communities - meaning that it is most likely to remove edges that connect different clusters or groups of connected nodes. The number of edges to remove is the main input for the algorithm, which isn't known without experimentation and guessing.

The second clustering algorithm tested was Voltage clusterer, based off Fang Wu and Bernardo A. Huberman's physics approach to clustering[19] which is also based on K-Means. Basically this algorithm mimics what K-Means does, but uses a voltage scorer as the means to group nodes together into clusters.

The third and final clustering algorithm tested on this framework was Weak Component Clustering, which finds the weak components (nodes) of the graph and isolates the remaining nodes. It defines weak component as a maximal subgraph in which all pairs of nodes in the subgraph are reachable from one another in the underlying undirected subgraph, i.e., the bridge between clusters.

All of the above, being part of the same framework, don't offer weight edges on graphs when clustering the nodes, so only "having handoffs" count, not the actual calculated distance between the nodes, which makes the distance irrelevant.

3.5 Searching rules and results from crossing AP records with social network data

This project from the business layer identifies anonymous users on the network. Fig. 3.15 shows the project architecture, presenting the different search rules and how their interfaces are exported.

[13] http://jung.sourceforge.net/

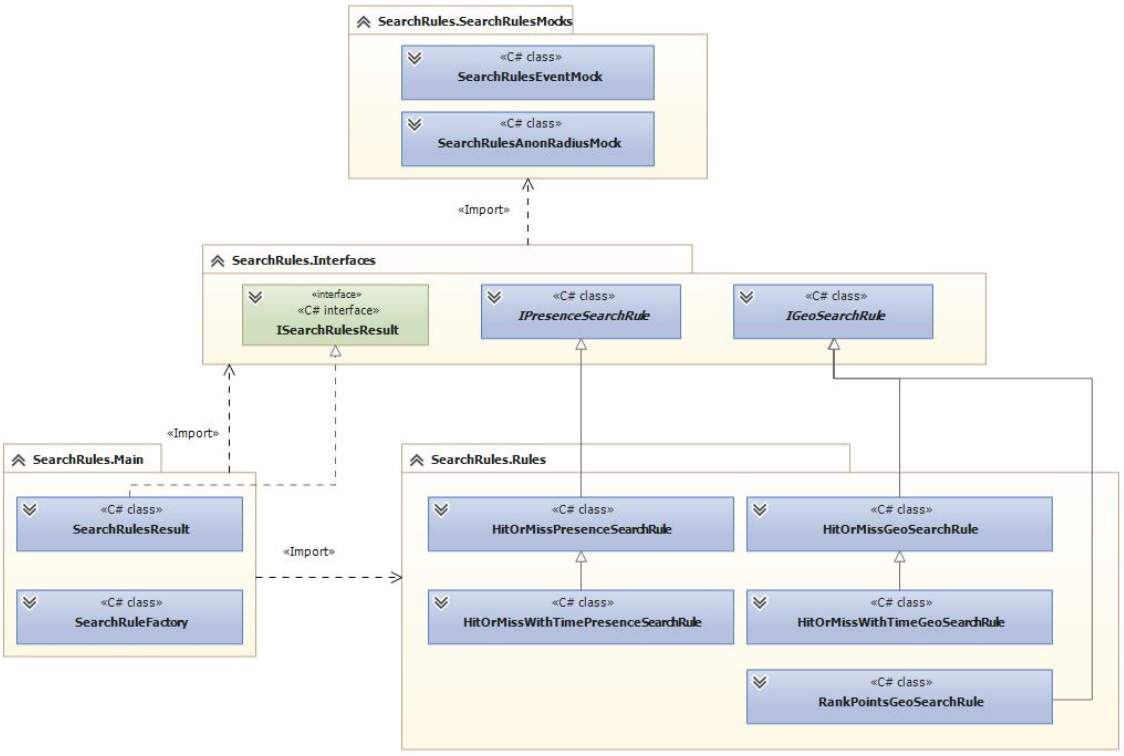


FIGURE 3.15: Search Rules - UML

It is composed by a set of rules that are used to cross AP wireless records and social network posts. Search rules are internal classes, therefore returned by the static factory, as an interface that is public. Only one rule is used at a time, since each one has a different search methodology.

3.5.1 Mocks

To allow the previous fetch of user events from any social network, this component can be used by any social network events, since it uses its own mock classes - events and access records mocks.

In this context, events can be Instagram posts or comments, Twitter tweets and Facebook posts or comments. Events only have three properties: the screen name, which represents the user's unique name on the social network; the GPS coordinates, which are optional because it depends on the search rule being used and there are rules that don't use geographic positioning; and the date of that post or comments creation, to then correlate with the connections at the same time on the Eduroam's network.

The mocks for the access records are useful for this project to have dependencies for the data access layer, since the access records can be parameterized by the calling project as access records mocks.

3.5.2 Search rules for user identification

A search rule only has one public method that can be called publicly – the `Execute`, that receives as parameters the social events and anon radius information and returns a collection of results.

A search rule allows inheritance and easy extension by having protected methods that the `Execute` method calls in sequence, that are `BeforeExecute`, `MidExecute`, `AfterExecute`. The names are self-explanatory: `BeforeExecute` uses the anon radius and events collections and tampers with them, allowing a pre-processing block to modify the data to that specific search rule; `MidExecute` uses that possibly-modified collections to perform the main processing and cross the data to return the results and possible user identifications; the `AfterExecute` allows some kind of post-processing where additional rules and filters can be applied to the results before coming out of the class.

We've implemented five rules, that can be either rules that act on network presence or geographical presence:

- `HitOrMissPresence` – Being strictly presential, this search doesn't use GPS coordinates from the social events, nor from the AP information on the records. It filters the network users that have **at least one connection opened on the network at the time of all social events for that particular social user**, with some margin of error. If the user X isn't connected to Eduroam at the time of one or more social network events for that social network user Y, the possibility X being Y is discarded and the final list narrows down. This process is done through all combinations of Eduroam's user - social network user pairs. This rule returns a collection of results that fit those characteristics. Was the first approach to the problem and the one that later incentivized the three-way division of the execution. Only implements the `MidExecute`, so derived rules can extend this by implementing the `BeforeExecute` and/or `AfterExecute`. The core algorithm for this rule is described in pseudocode in Alg.3.

Algorithm 3 HitOrMissPresence Rule

```
1: resultsList ←empty list
2: for all anonRadiusUsername in allAnonRadius.names do
3:     for all eventUsername in allEvents.names do
4:         canBeThisUsername ← true
5:         for all event in allEvents[eventUsername] do
6:             isInThisEvent ← false
7:             for all anonRadiusRow in allAnonRadius[anonRadiusUsername] do
8:                 if isInsideTimeFrame(anonRadiusRow, event) then
9:                     isInThisEvent ← true
10:                    break
11:                end if
12:            end for
13:            if isInThisEvent == false then
14:                canBeThisUsername ← true
15:                break
16:            end if
17:        end for
18:        if canBeThisUsername == true then
19:            resultsList.add(anonRadiusUsername, eventUsername)
20:        end if
21:    end for
22: end for
23: return resultsList
```

- `HitOrMissWithTimePresence` – This rule is inherited directly from `HitOrMiss` and adds functionality to the post processing method. Using the results from the `MidExecute` method from the `HitOrMiss` rule, a new filter is applied that removes all Eduroam's users that have more average hours connected per day than a specified threshold value from the results. This works by cycling through all network users that are present in the results and counting their total hours connected throughout the month, and on how many days that they have at least a connection established. The final value for average daily hours per month, for a user, is determined by:

$$averageDailyHours = \frac{totalMonthlyHoursConnected}{daysConnected}$$

. This post-processing filter is explained in pseudocode in Alg.4. If $averageDailyHours$ is above a specified threshold, then it is considered a school's computer that remained powered on and connected to the network through most days and nights, therefore not being part of this social study. The threshold used was 17 hours and that value was decided after several tests to effectively eliminate the devices that had high probability of being those outliers.

Algorithm 4 HitOrMissWithTimePresence Rule (post-filter)

```
1: newResultsList ←empty list
2: hoursForEachAnonRadiusList ←empty list
3: daysForEachAnonRadiusList ←empty list
4: for all result in resultsList do
5:     for all eventUsername in allEvents.names do
6:         if daysForEachAnonRadiusList.contains(result.anonRadiusUsername) then
7:             continue
8:         end if
9:         hoursForEachAnonRadiusList.add(result.anonRadiusUsername, 0)
10:        daysForEachAnonRadiusList.add(result.anonRadiusUsername, 0)
11:        for all anonRadiusRow in allAnonRadius[result.anonRadiusUsername] do
12:            dateBeg ← anonRadiusRow.connectionStart.hour
13:            dateEnd ← anonRadiusRow.connectionEnd.hour
14:            while dateEnd >= dateBeg do
15:                if !hoursForEachAnonRadiusList[result.anonRadiusUsername].contains(dateEnd) then
16:                    hoursForEachAnonRadiusList[result.anonRadiusUsername].add(dateEnd)
17:                end if
18:                if !daysForEachAnonRadiusList[result.anonRadiusUsername].contains(dateEnd) then
19:                    daysForEachAnonRadiusList[result.anonRadiusUsername].add(dateEnd)
20:                end if
21:                dateEnd ← dateEnd.addHours(-1)
22:            end while
23:        end for
24:        if hoursForEachAnonRadiusList[result.anonRadiusUsername].count / hoursForEachAnonRadiusList[result.anonRadiusUsername].count < HOUR_AVERAGE_THRESHOLD then
25:            newResultsList.add(result)
26:        end if
27:    end for
28: end for
29: return newResultsList
```

- `HitOrMissGeo` – This rule serves as basis for other rules that identify users based on network presence and geographical position. Only implements the `MidExecute` and it starts by assigning the social network name to their events, and then the events to the buildings where they happened based on their geographic coordinates as explained in pseudocode in Alg.5. Like the HitOrMissWithPresence rule, it filters the network users that have at least one connection opened on the network at the time of all social events for that particular social user, but all those connections have to be with an AP that belongs to the fuzzy cluster or the cluster on the building the social event happened. The fuzzy cluster works as a "blind-spot" for this algorithm since if a user is always connected to an

AP that ended up on the fuzzy cluster then he is always wrongly filtered by this rule. This rule, not only enforces network presence, but also geographical presence.

Algorithm 5 HitOrMissGeo Rule (assigning social network names to events and events to buildings)

```
1: resultsList ←empty list
2: screenNameToEventsToBuildings ← empty list ▷ Nested dictionaries: social network name to his events. Events to buildings where happened.
3: for all eventUsername in allEvents.names do
4:     screenNameToEventsToBuildings.add(eventUsername, empty dictionary)
5:     for all event in allEvents[eventUsername] do
6:         screenNameToEventsToBuildings[eventUsername].add(event, empty list)
7:         for all buildingName in buildingsLocations do
8:             for all hotSpot in buildingsLocations[buildingName] do
9:                 euclidianDistance ← harvesineInMeters(event.coordinates, hotSpot.coordinates)
10:                if hotSpot.radiusInMeters >= euclidianDistance then
11:                    screenNameToEventsToBuildings[eventUsername][event].add(buildingName)
12:                    break
13:                end if
14:            end for
15:        end for
16:    end for
17: end for
```

- `HitOrMissWithTimeGeo` – Is an extension to the `HitOrMissGeo` rule, implementing only the post processing method, similar to what `HitOrMissWithTimePresence` does, filtering the users that have more than a defined threshold of hours logged a day, in average.

- `RankPointsGeo` – This rule was implemented to address the issues raised for using the fuzzy cluster. The use of a fuzzy cluster makes all users connected to an AP in that cluster ignored by the rule, since it's impossible to know if that AP really belongs or not to the building where the social event occurred. A rank for each user was created while identifying the user, in which points are awarded for:

  1. Having a connection opened at the time of the social event;
  2. Having a connection opened at the time of the social event to an AP that belongs to the fuzzy cluster;
  3. Having a connection opened at the time of the social event to an AP that belongs to the building where the event happened.

The further down the list, the more points awarded. The filtering works like `HitOrMissGeo`, starting by assigning the social network name to their events, and then the events to the buildings where they happened based on their geographic coordinates (Alg.5). However, with this rule, being on the correct building (cluster) provides a higher reward than while existing on the fuzzy cluster's APs, thus obtaining more correct results. Alg.6 shows the pseudocode that cycles through all anonymous users on the network and then all social events; if he has been present on those events he gets rank points – more if on the correct building, less if on the fuzzy cluster; if not present on either clusters for at least one event, he's not part of the final results. The users with the most ranked points get returned as matches.

On this algorithm the network presence reward was determined to be of zero points, because if a network user is required to be present in all events to be considered a final result, then all final results would have the same amount of point gain from that reward. Since a user can be on the final results by being either on the fuzzy cluster or specific building cluster for any amount of events, then different rewards had to be applied to two different occurences – the specific location is more rewarding with five points, to one point from the fuzzy cluster presence.

3.5.3 Geographical presence validation method

A component was developed to determine the distance between two geographic points, given their latitudes and longitudes, through the harvesine formula. It was used to determine which building the social event occured in.

To know where the actual buildings are, an input parameter must be passed to the search rules that include the location of buildings and their key-name according to the clustering and search rules algorithms. That parameter includes the perimeter points for all buildings in ISEL, where for each building multiple geographic coordinates are taken (corners, preferably) and a radius of action is assigned to them; Every social event inside that radius for that specific geographic location, is considered to have happened in that specific building. For a different campus or solution, there must be a different manually coded perimeter collection for input.

In Fig. 3.16 it is shown the example used on this solution, applied for ISEL and this thesis' results, where different colored points indicate different buildings (radius for each point ommited).

Algorithm 6 RankPointsGeo Rule

```
1: resultsList ←empty list
2: for all anonRadiusUsername in allAnonRadius.names do
3:     for all eventUsername in allEvents.names do
4:         thisUsernameRankPoints ← 0
5:         canBeThisUsername ← true
6:         for all event in allEvents[eventUsername] do
7:             isInThisEvent ← false
8:             for all anonRadiusRow in allAnonRadius[anonRadiusUsername] do
9:                 if isInsideTimeFrame(anonRadiusRow, event) then
10:                    buildingsForThisAp ←empty list
11:                    for all buildingName in apsLocations do
12:                        for all ap in apsLocations[buildingName] do
13:                            if ap == anonRadiusRow.ap then
14:                                buildingsForThisAp.add(buildingName)
15:                                break
16:                            end if
17:                        end for
18:                    end for
19:                    thisUsernameRankPoints ← thisUsernameRankPoints + PRESENCE_REWARD
20:                    if screenNameToEventsToBuildings[eventUsername][event].Intersect(buildingsForThis
then
21:                        thisUsernameRankPoints ← thisUsernameRankPoints + GEO_PRESENCE_REWARD
22:                        isInThisEvent ← true
23:                        break
24:                    end if
25:                    if buildingsForThisAp.contains("Fuzzycluster") then
26:                        thisUsernameRankPoints ← thisUsernameRankPoints + FUZZYCLUSTER_REWARD
27:                        isInThisEvent ← true
28:                        break
29:                    end if
30:                end if
31:            end for
32:            if ! isInThisEvent == false then
33:                canBeThisUsername ← true
34:                break
35:            end if
36:        end for
37:        if canBeThisUsername == true then
38:            resultsList.add(anonRadiusUsername, eventUsername, thisUsernameRankPoints)
39:        end if
40:    end for
41: end for
42: return resultsList
```

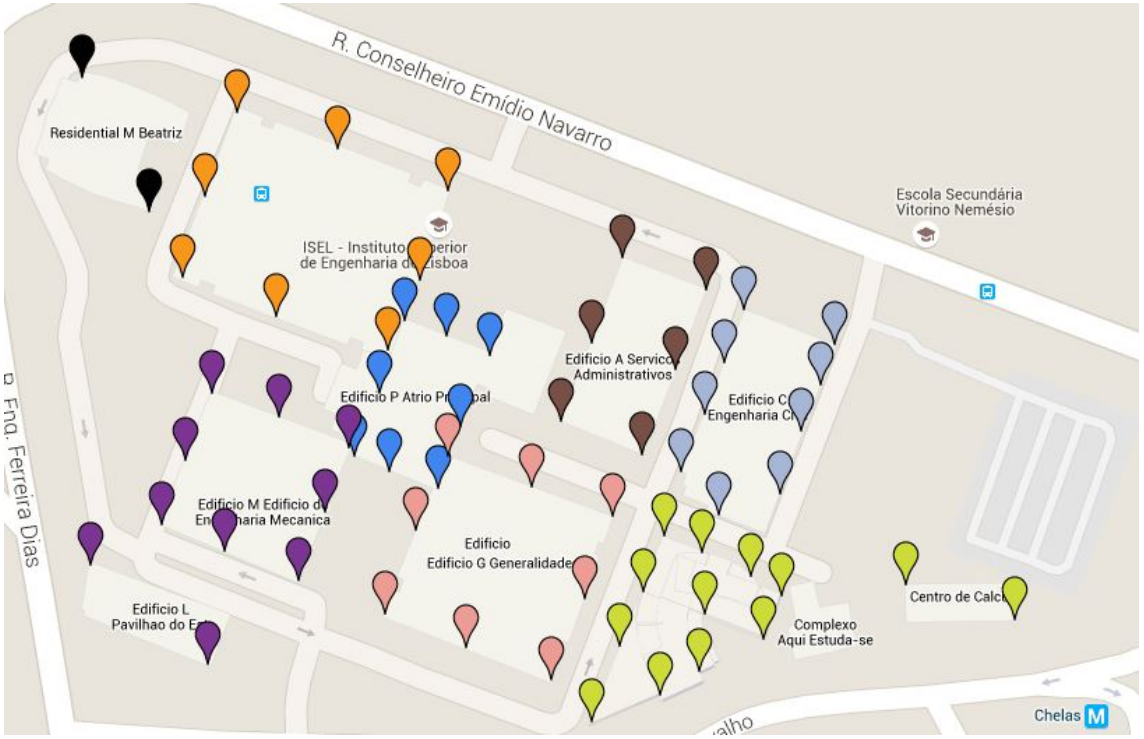


FIGURE 3.16: Perimeter points for each ISEL building

3.5.4 The search results

The search result is an object that contains two properties: the wireless access records' username and the social network name. Each property identifies uniquely a person on its own dataset. That object says that, after the rule's filtering and searching, there's a high a chance that username X from the Eduroam's network is actually the user Y from the social network.

3.6 Presentation layer and graphic interface

This layer exists mainly for scaling purposes, since the main program course runs in a linear pipeline in search of present social information and user identification. This is the solution's entry point for execution, which consists of only calling the service methods on the business layer since it's the only dependency.

The current state includes only console interface without any input, only processing with service calls hardcoded on the entry point class. The workflow goes from calling the Twitter service with the context for a user, then obtaining his feed; then it calls AnonRadius service to get all the

AP records entries; finally the Twitter service is called again to execute the user identification, passing both database and social network info as arguments.

Future work for this layer includes view-controller architecture for every option selectable by the user, that will include which social network and user to gather events from and how is the user identification going to proceed.

Chapter 4

Results

This chapter addresses the results and tests performed to answer the main question and thesis' goal: *"How many social network check-ins are needed for anonymity to be broken?"*. Before the results, we will start by presenting the environment where the evaluation was performed. The presented results are split into results obtained while using just the presence information, and results with AP location information.

4.1 Context

In both cases the test is ran with 14 tweets made between the dates 21/05/2015 and 29/05/2015 from the account screen-named "@PedroBritoNunes" with the real name "Pedro Nunes". The tweets were made from the same device (a smartphone), with GPS coordinates included in most of them, while connected to the Eduroam's network.

All tests were made to identify the user with the name "Pedro Nunes" on the network, not assuming that all tweets were made on ISEL's Eduroam network. Before each rule is applied, the tweets that dont' fit into any building points were removed from the events list.

4.2 Results without location information

These tests were ran without making use of either tweets GPS information or called stations (APs) locations. Since they made use of the search rule `HitOrMiss` and `HitOrMissWithTime`,

only the network presence is relevant for this kind of search.

The tests without geographic location were divided into two separate tests: one to try and identify the user on the network, the other to do the same but with minimum possible tweet amount to identify the same user.

4.2.1 With all tweets

This test uses all 14 tweets from "@PedroBritoNunes" to try and identify him on the network. Using the `HitOrMiss` search rule the output was:

```
For real name = PedroBritoNunes and 14 events:
-AnonUserName = 70
-AnonUserName = 127
-AnonUserName = 642
-AnonUserName = 939
-AnonUserName = 1065
-AnonUserName = 1236
-AnonUserName = 1618
-AnonUserName = 2127
-AnonUserName = 2185
-AnonUserName = 6288
```

This means that there were 10 different users that were connected to ISEL's Eduroam network during the time that all tweets were posted by this social netwrok account. This is inconclusive since no user can be identified with certainty, which leads to the use of the derived rule - `HitOrMissWithTime`. This rule was created to exclude outliers from the result pool. Analyzing the daily average hours connected to the network, all of those users except one had more than the threshold of 15 daily hours logged. This threshold was set to eliminate the devices that remained connected to the network through the night, therefore not considered fit for this analysis.

So, still using all 14 tweets from "@PedroBritoNunes" and using `HitOrMissWithTime` search rule, the output was:

```
For real name = PedroBritoNunes and 14 events:
-AnonUserName = 939
```

Now a single user is isolated and it can be said with a high probability of certainty that the anonymous user 939 is actually the owner of the Twitter account "@PedroBritoNunes", with the real name "Pedro Nunes".

4.2.2 With minimum tweets possible

This test addresses directly the main question. Knowing beforehand that `HitOrMissWithTime` rule successfully identifies a user with all 14 tweets, the challenge now is trying to use the minimum amount of tweets to identify the same anonymous user.

The same process used with all tweets was set in motion, but starting by using only 1 random tweet from the pool, consequently adding 1 more random each time the results come with more than a single user. When the results list come with a single user, the process stops and outputs the results. The randomness using tweets is important, because each tweet timestamp crosses with different sets of users on the network records, thus different tweet sets with the same size can achieve different results. Nevertheless, the goal is to discover the minimum number of tweets that can isolate a single user. Still using `HitOrMissWithTime` on a loop, adding tweets to a number of different combinations, the output was:

```
For real name = PedroBritoNunes and 5 events:
-AnonUserName = 939
```

This concludes that 5 tweets is the minimum number of social events required to identify an anonymous user on the network without any use of geographic information, given the context.

4.3 Results with location information

The same tests will be conducted, but now with geographic location obtained from the clustering algorithms and using the perimeter points from ISEL, using both `HitOrMissWithTimeGeo` and `RankPointsGeo` rules. Notice that only 9 events were used because 5 of them are filtered on the pre processing method of the search rules for not being geographically included in the radius for any ISEL's building perimeter points – in this case, those tweets didn't have geographic coordinates attached.

4.3.1 With all tweets

This test uses all 14 tweets from "@PedroBritoNunes" to try and identify him on the network. Using the `HitOrMissWithTimeGeo` search rule the output was:

```
For real name = PedroBritoNunes and 9 events:
-AnonUserName = 939
```

Now using not only network presence but also geographic positioning, the same user 939 is still solely identified as Pedro Nunes. It means the user 939 was, during all reliable tweets, at the correct building (cluster) or fuzzy cluster. This can be conclusive, but as said above the fuzzy cluster can wrongfully retrieve false results because the user 939 could have very well been connected to any AP that belongs to the fuzzy cluster during all social events, ending up on the results.

To avoid being in doubt, the `RankPointsGeo` rule was used for all tweets too:

```
For real name = PedroBritoNunes and 9 events:
-AnonUserName = 939
```

This concludes that during all tweets, the user 939 is most likely to be Pedro Nunes, based on the fact that there is not any user that has been more times in any combination of fuzzy cluster or correct building. And since being on the correct building is much more rewarding than being on the fuzzy cluster, the user 939 stands out even more, despite existing the really low probability of user 939 *still* being on the fuzzy cluster during all tweets and all other users missing the correct building or fuzzy cluster.

4.3.2 With minimum tweets possible

In this subsection we present the results for identification of a user possibly being "@PedroBritoNunes", but with the minimum tweets possible among all different combinations from tweets from that user.

Using the `HitOrMissWithTimeGeo` rule, the output was:

```
For real name = PedroBritoNunes and 3 events:
-AnonUserName = 939
```

With this result is possible to assert that the user 939 could be identified with the minimum of 3 tweets and that he was always on the correct building, being the sole user always on the fuzzy cluster or in any combination of both. But present only on either nonetheless.

Now using the ranking rule `RankPointsGeo` the results obtained are:

For real name = PedroBritoNunes and 2 events:
-AnonUserName = 939

Identifying now with 2 tweets instead of 3 means that there was at least one more user on the network that was connected to the fuzzy cluster, and an extra tweet was necessary to correctly pinpoint the actual unique user to be "@PedroBritoNunes". This happens because, as said before, the `HitOrMissWithTimeGeo` rule treats the fuzzy cluster as also being the correct building because we don't know where the APs included in that cluster really belong to. Since the `RankPointsGeo` rule rewards more the correct cluster than the fuzzy cluster, the user 939 gets more points than the others and is isolated with only 2 tweets, meaning that in those two events he was at least once in a correct cluster.

This concludes that 2 tweets is the minimum number of social events required to identify an anonymous user on the network using geographic information calculated solely by clustering and working information from the anonymous AP records, without any prior knowledge of the real APs location or users real identity.

Chapter 5

Conclusion and future work

This thesis has aimed to develop an application that identifies a student on anonymous WiFi access points records in an academic environment, based solely on social network information. The solution was experimented and tested on a single social network user, managing to identify that user on the wireless network records, in a proof-of-concept methodology. However, since it's divided in very different programming components, it manages to be flexible, modular and scalable to the point where different locations and social networks can be used, as well as even different search rules for as many number of users as desired.

When developing an application that uses a lot of external data as input and manages 2D geographic locations, several points have to be taken into consideration:

1. Social network information isn't always available, as it requires internet connection or can be pulled from fetching possibilities for time constraints;
2. Most prominent social networks don't provide the API calls to fetch the required information, such as GPS coordinates from social events;
3. AP records must have consistent, coherent and precise information - connection beginning and ending time, preferably to the second, and AP ID connected must always be the same physical wireless AP;
4. The results are less precise the taller the building - the undirected graph and coordinates work in two-dimensional basis, where the calculated distance is a metric for a X-Y axis, completely disregarding a third dimension.

Using data that doesn't follow these points may result in a bigger margin of error when clustering the APs, thus ending in more identification misses.

The concluded work allows to overcome the false sense of privacy and security based on providing publicly anonymous data such as access points records from a complex wireless network. It helps understand how that data can be used to sketch a 2D "map" of a university, based only on handoff time differences, as a key to identify and study anonymous users on that network.

5.1 Future work

The work done and code developed on this thesis has many possibilities for improvements and branching out, being in this same context or another one related to both social network and/or wireless network data analysis. The following list is composed of possible future work that can be developed and extended from this one as base:

- Making use of other highly location-reliable social network, such as Instagram, for the same context;
- Scale the solution for better 3D usage;
- Better clustering algorithm, with less margin of error;
- Crossing social network events and profiles for the same purpose and better user identification;
- Applying the same work on a larger scale, or community and population with much more Twitter penetration for better user identification;
- Performing human behaviour analysis based on the wireless network records;
- Performing human movement patterning and prediction based on the wireless network records;
- User classification based on traffic volume, visited buildings and movement routes through the wireless network records;
- Predict future user presence on the network based on analysis of past and future external events.

Bibliography

[1] Danyl Bosomworth. Statistics on mobile usage and adoption to inform your mobile marketing strategy, July 2015.

[2] eMarketer. 2 billion consumers worldwide to get smart(phones) by 2016, December 2014.

[3] eMarketer. Smartphone users worldwide will total 1.75 billion in 2014, January 2014.

[4] Kimberlee Morrison. The growth of social media: From passing trend to international obsession, January 2014.

[5] Edison Research. The infinite dial 2015, March 2015.

[6] Group marktest. Social media statistics for portugal, February 2014.

[7] N. Cruz, H. Miranda, and P. Ribeiro. O impacto dos smartphones nos modelos de mobilidade tradicionais. In *INForum 2014*, pages 195–210, September 2014.

[8] Nan Li and Guanling Chen. Analysis of a location-based social network. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 4, pages 263–270, August 2009.

[9] A. Cecaj, M. Mamei, and N. Bicocchi. Re-identification of anonymized cdr datasets using social network data. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 237–242, March 2014.

[10] Tom Nicolai, Eiko Yoneki, Nils Behrens, and Holger Kenn. Exploring social context with the wireless rope. In *Proceedings of the 2006 International Conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET - Volume Part I*, OTM'06, pages 874–883, Berlin, Heidelberg, 2006. Springer-Verlag.

[11] Wei-jen Hsu, Debojyoti Dutta, and Ahmed Helmy. Mining behavioral groups in large wireless lans. *CoRR*, abs/cs/0606002, 2006.

[12] MarangazeMunhepe Mulhanga, SolangeRito Lima, and Paulo Carvalho. Characterising university wlans within eduroam context. In Sergey Balandin, Yevgeni Koucheryavy, and Honglin Hu, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, volume 6869 of *Lecture Notes in Computer Science*, pages 382–394. Springer Berlin Heidelberg, 2011.

[13] A. B. M. Musa and Jakob Eriksson. Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pages 281–294, New York, NY, USA, 2012. ACM.

[14] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125, May 2008.

[15] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Nature srep.*, 3, 2013.

[16] Hui Zang and Jean Bolot. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 145–156, New York, NY, USA, 2011. ACM.

[17] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[18] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 100–108, 1979.

[19] F. Wu and B.A. Huberman. Finding communities in linear time: a physics approach. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):331–338, 2004.