# Modeling Mobility Patterns

by

Adele M. Hedrick

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Computer Science

University of Ontario Institute of Technology

Supervisors: Dr. Ken Pu, Dr. Ying Zhu

November 2017

# Abstract

This work focuses on analysis and model generation for user mobility patterns given a sequence of observed WiFi signals. Built on the Android platform, the data collection mobile application gathers WiFi sensor readings (BSSID and SSID). The implemented pipeline performs location identification using an online hierarchical timeline clustering algorithm and segmentation algorithm. The segmentation algorithm constructs a tree of location candidates which are then aggregated by a similarity measure based on their BSSID and SSID features. The generated locations are processed to extract mobility patterns. A pattern is a sequence of location transitions which have high information content, high activity over time, and high degree of predictability. Each of these aspects are described by a numerical measure based on statistical properties of the location observations in a feature space.

# Acknowledgements

I would first like to thank my supervisors Dr. Ken Pu and Dr. Ying Zhu. The numerous meetings in Dr. Pu's office helped guide me through not only this research, but my career as well. The support I found in Dr. Zhu has been instrumental in completing this research, and I believe I have made a lifelong friend in her.

I will forever be grateful to my parents, Bruce and Inamae, for their support as parents. More importantly, I am grateful for their support as grandparents. I would not have been able to focus on my research without knowing my children were in the best care possible.

My children, Jaxon and Tiberius, have motivated me to become the best version of me possible. Forcing me to take breaks to read a book or play games. They are the best kids a mother could ask for, and I am honored to be theirs.

Lastly, I wish to express my love and gratitude to my husband, Ryan. His support as a husband is what gave me the freedom to achieve all my goals and dreams.

# Contents

**Abstract** i

**Acknowledgements** ii

**Contents** iii

**List of Figures** v

**List of Tables** vii

**List of Symbols** ix

**1 Motivation and Problem Definition** 1
- 1.1 Motivation . . . 1
- 1.2 Localization and mobility patterns . . . 3
- 1.3 Problem definition . . . 3

**2 Related Work** 6

**3 Algorithms and Analytical Techniques** 10
- 3.1 Hierarchical time segmentation . . . 10
  - 3.1.1 Traditional hierarchical clustering . . . 11
  - 3.1.2 New hierarchical clustering approach: ht-append . . . 12
  - 3.1.3 Segmenting the hierarchy timeline: ht-segment . . . 22
- 3.2 Locations and semantic grouping . . . 24
  - 3.2.1 Physical locations . . . 24
  - 3.2.2 Semantic locations . . . 25
- 3.3 Mobility patterns . . . 30
  - 3.3.1 Evaluation . . . 32

**4 System Implementation** 40
- 4.1 An app for mobile data collection . . . 40
- 4.2 A pipeline for mobility pattern analysis . . . 42
  - 4.2.1 Hierarchical clustering . . . 43

4.2.2 Segmentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.2.3 Localization . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

**5 Case Study** **46**
5.1 Description of the case study . . . . . . . . . . . . . . . . . . . . . . 46
5.2 Discussion and visualization . . . . . . . . . . . . . . . . . . . . . . 47
5.3 Evaluation of localizations . . . . . . . . . . . . . . . . . . . . . . . 53

**6 Conclusion** **64**
6.1 Contribution and summary . . . . . . . . . . . . . . . . . . . . . . . 64
6.2 Lessons learned . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
6.2.1 Mobile application development . . . . . . . . . . . . . . . . . 65
6.2.2 Data analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
6.2.3 Data visualization . . . . . . . . . . . . . . . . . . . . . . . . . 66
6.3 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

**Bibliography** **68**

# List of Figures

1.1 Movement of mobile device . . . . . . . . . . . . . . . . . . . . . . . . 4
1.2 Timeline of readings . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.1 Traditional Hierarchical Clustering . . . . . . . . . . . . . . . . . . . . 12
3.2 Proposed hierarchical time clustering using ht-append . . . . . . . . 13
3.3 Clustering the first 4 readings of the timeline . . . . . . . . . . . . . . 15
3.4 Clustering the 6th reading of the timeline . . . . . . . . . . . . . . . . 16
3.5 Clustering the 6th reading of the timeline . . . . . . . . . . . . . . . . 17
3.6 Clustering the 7th reading of the timeline . . . . . . . . . . . . . . . . 17
3.7 Clustering the 8th reading of the timeline . . . . . . . . . . . . . . . . 18
3.8 Clustering the 9th reading of the timeline . . . . . . . . . . . . . . . . 19
3.9 Clustering the 10th reading of the timeline . . . . . . . . . . . . . . . 20
3.10 Clustering the 11th reading of the timeline . . . . . . . . . . . . . . . 20
3.11 Clustering the 12th reading of the timeline . . . . . . . . . . . . . . . 21
3.12 Clustering the 13th to 16th reading of the timeline . . . . . . . . . . . 22
3.13 Physical location identification example . . . . . . . . . . . . . . . . . 24
3.14 Physical to semantic location conversion . . . . . . . . . . . . . . . . . 27
3.15 A list of singleton keys . . . . . . . . . . . . . . . . . . . . . . . . . . 28
3.16 A list of paired keys . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
3.17 Reducing semantic locations by paired-keys . . . . . . . . . . . . . . . 31
3.18 Reducing semantic locations by singleton-keys . . . . . . . . . . . . . 32
3.19 Generating a semantic timeline . . . . . . . . . . . . . . . . . . . . . . 35
3.20 Activity of semantic location . . . . . . . . . . . . . . . . . . . . . . . 36
3.21 Transition diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
3.22 Training and test set creation . . . . . . . . . . . . . . . . . . . . . . . 37
3.23 Sequential Neural Network . . . . . . . . . . . . . . . . . . . . . . . . 38

4.1 WiFi Scanner Mobile Application . . . . . . . . . . . . . . . . . . . . . 42
4.2 Performance of hierarchical append function . . . . . . . . . . . . . . 43

5.1 BSSID count for top 20 SSIDs . . . . . . . . . . . . . . . . . . . . . . . 48
5.2 Reading count for top 20 SSIDs . . . . . . . . . . . . . . . . . . . . . . 49
5.3 Total number of generated locations in localizations . . . . . . . . . . 51

5.4 L2 Week of Activity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
5.5 Transition diagrams with different levels of localization . . . . . . . . 53
5.6 Transition diagram of L2 . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
5.7 Transition diagram of L2 marking activity $> 1$ . . . . . . . . . . . . . . . 56
5.8 Transition diagram of L2 marking information $< 1$ . . . . . . . . . . . 57
5.9 Transition diagram of L2 marking regularity $> 0$ . . . . . . . . . . . . 58
5.10 L0 evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.11 L1 evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . . . 59
5.12 L2 evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . . . 59
5.13 L0 transition diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
5.14 L1 transition diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.15 L2 transition diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.16 L0 2D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 60
5.17 L0 3D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 61
5.18 L1 2D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 61
5.19 L1 3D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 62
5.20 L2 2D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 62
5.21 L2 3D evaluation comparison . . . . . . . . . . . . . . . . . . . . . . . 63

# List of Tables

3.1 Transition count matrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

5.1 Total number of BSSID vs SSID observed in case data set . . . . . . . . 47
5.2 L2 Semantics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
5.3 L2 Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
5.4 Localization comparisons . . . . . . . . . . . . . . . . . . . . . . . . . 55

# Listings

3.1 Tensor Flow with Keras implementation . . . . . . . . . . . . . . . . . . . 37

# List of Symbols

$bssid$ Single MAC address of a WiFi hotspot.

$\mathcal{B}$ A set of $bssid$values.

$c$ A cluster of readings. Contains a left child and a right child, which may be other clusters or readings, $r$.

$\mathcal{H}$ A collection of clusters, $c$.

$\mathcal{H}_T$ A collection of clusters, $c$, that maintain the order of the timeline of readings, $\mathcal{T}_r$, that they represent at the leaf clusters.

$l$ A location consisting of groups of hotspots.

$l_p$ A location consisting of clusters grouped by BSSID values.

$\mathcal{L}_p$ A collection or list of physical locations, $l_p$.

$l_s$ A semantic location consisting of a set of SSID values.

$\mathcal{L}_s$ A collection or list of semantic locations, $l_s$.

$r$ A tuple of $\langle t, set\left(\langle x_0, s_0\rangle, \langle x_1, s_1\rangle .. \langle x_n, s_n\rangle\right)\rangle$ where $n$ is the number of hotspots, $x$, observed in the reading.

$\rho$ Physical location similarity threshold used in loc-physical.

$\sigma$ Segmenting similarity threshold used in ht-segment.

$sim$ Jaccard similarity measure that compares sets of BSSID.

$ssid$ Single user generated name of a WiFi hotspot. These may not be unique.

$\mathcal{SSID}$ A set of $ssid$values.

$s$ A signal strength of the respective hotspot, $x$, in a reading, $r$.

$\mathcal{T}_l$ A timeline of semantic locations.

$\mathcal{T}_r$ A timeline of readings.

$\mathcal{T}_s$ A timeline of cluster segments.

$t$ A date-time value in the form of a string.

$x$ A hotspot in the form of a tuple: $\langle ssid, bssid \rangle$.

# Chapter 1

# Motivation and Problem Definition

## 1.1 Motivation

Obtaining a personal movement history and mobility pattern allows for the development of mobility analysis on an individual level. Traditionally, data samples are collected from a large population and can be used to generate insights [42]. However, this accumulation may not accurately reflect any one individual sampled within the population. By collecting large amounts of data from a single individual, we are able to obtain a dataset that is reflective of that person, so any insights gained would be tailored and specific to that person. Personalized insights that are accurate come at the risk of loss of privacy due to the nature of the task when utilizing cloud storage. Data needs to be collected and transfered to a server that is capable of doing the analysis, and therefore privacy is at risk.

The goal of this project was to collect personalized mobility data, provide a means to analyze it without putting privacy at risk, generate a database of locations defined by hotspots that are in close proximity to one another, and generate a timeline of those locations for further analysis to discover mobility patterns.

As a person's movement patterns is the focus, and prior knowledge of the environment the person frequents is not necessary. Only locations based upon detected hotspots needed to be defined. This sets the stage for the development of an ad-hoc system.

Two challenges, which were mentioned in [19], that this project aims to overcome are: privacy, and complexity of analysis. To maintain privacy, creation of physically unaware localization and movement analysis that can be achieved within the confines of the data collection device is needed. Traditional methods [2, 10, 11, 14, 22, 24, 27, 28, 30–32, 36, 36, 38–41] were not conducive, as their complexities were beyond the scope of this project.

The uses of both GPS and WiFi are inherently private, but this privacy is lost with the transfer of the location data being sent to cloud storage, and is at the mercy of nefarious individuals. Privacy is maintained with GPS so long as the data is locally stored and not sent to cloud storage. Typically applications that utilize GPS location positioning send the location data to cloud storage (e.g. Google Maps Timeline [33]). In order to create a history of location, privacy is currently at risk, and therefore mobility analysis can not be achieved without putting privacy at risk as well.

When Android devices scan for WiFi hotspots, they do so passively; they listen for beacons emitted by hotspots. By doing passive scans, an Android device does not provide any of its own information. The risk of privacy being lost is again caused by sending data to cloud storage.

The decision to use WiFi signals over GPS, was twofold: WiFi scanning is a more power efficient means of data collection than the use of GPS on a regular basis [6], and WiFi will provide a lower level of resolution (indoor and outdoor) than GPS will allow.

Another challenge was the nature of the data. WiFi signals are subject to many forms of interference, including the ability of individuals to create their own ad-hoc WiFi hotspots. The creation of their own temporary, moving hotspots would create noise in the data collected.

## 1.2 Localization and mobility patterns

To generate a mobility pattern, localization is required. By definition, localization is a means to find previously unknown locations of a device by means of using devices with known locations and signal strengths to triangulate the unknown positions. Since the goal of this work is to have an ad-hoc system that maintains privacy, the scope of localization needs to be modified. In this work, clustering of a timeline of WiFi signals is used to achieve localization. This version of localization doesn't take into account the position of the WiFi hotspots in the real world, only what WiFi hotspots are observed within the same readings. Once a localization has been established, the timeline can then be mapped to a timeline of the locations that exist in the localization, and with this simplified timeline, mobility patterns can be extrapolated.

## 1.3 Problem definition

WiFi hotspots, $x$, have a single BSSID, $bssid$, known as a MAC address and a single user generated SSID, $ssid$; $x = \langle bssid, ssid \rangle$. A device scanning for WiFi signals would obtain a set of hotspots with their respective signal strength, $s$, in decibels (dB). A reading, $r$, was defined to be a timestamp, $t$, with a set of tuples containing a hotspot and the signal strength; $r = \left\langle t, set(\langle x_0, s_0 \rangle, \langle x_1, s_1 \rangle, \langle x_2, s_2 \rangle ... \langle x_n, s_n \rangle) \right\rangle$

where $n$ is the number of hotspots discovered at the scan recorded in the reading.

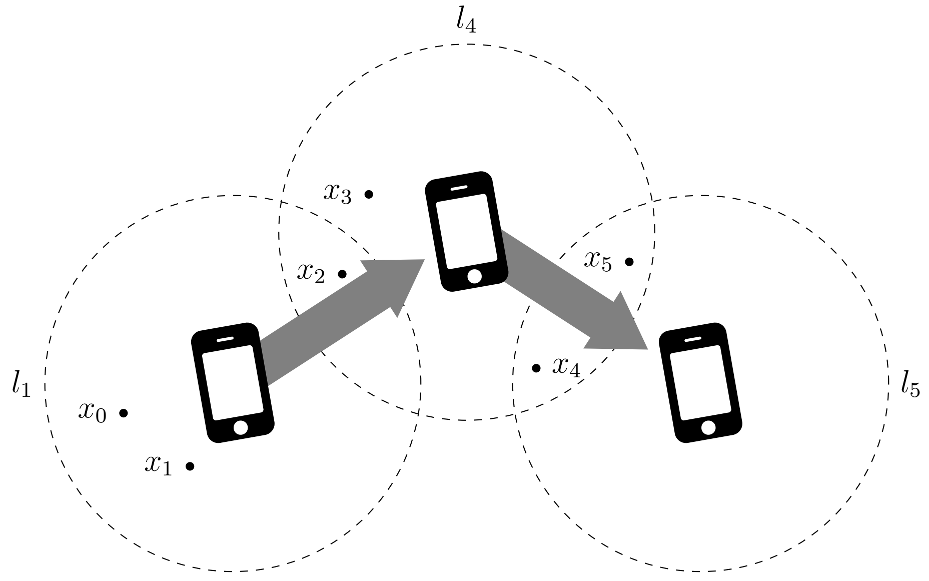


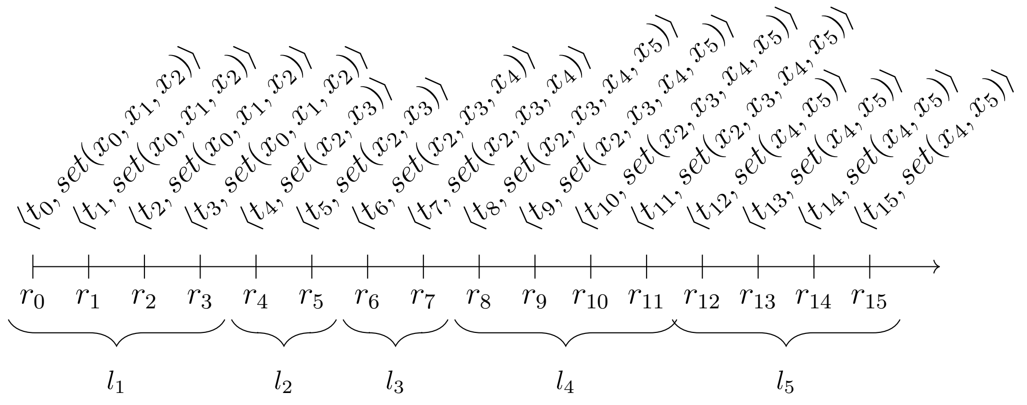
Figure 1.1: Movement of mobile device

A timeline of readings, $\mathcal{T}_r$, is a sequence of readings, where the $t$ in $r_i$ would always be less than the $t$ in $r_{i+1}$. A location, $l$, was defined as being a set of hotspots frequently seen together. A timeline of locations would be a mobility pattern and give us a view of a person's movement at a higher level than the timeline of readings.

Figure 1.1 illustrates the movement of a mobile device in a physical space, with Figure 1.2 illustrating the timeline of readings that would represent the physical movement. The mobile device has a limited distance in which it can detect WiFi hotspots, and as the mobile device moves, the radius in which it can detect hotspots moves along with it.

By grouping similar readings together, a set of hotspots then defines a location, $l$. Some locations were more significant than others, so two types of locations were defined; transient and persistent. A transient location is one that has few readings or a small length of time spent there; an example being $l_2$, and $l_3$ in Figure 1.2.

A persistent location is one that is significant either by the duration of time spent there, or the frequency of visits. In Figures 1.1 and 1.2, the persistent locations would be $l_1$, $l_4$ and $l_5$.



$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$l_1$
$l_2$
$l_3$
$l_4$
$l_5$

Figure 1.2: Timeline of readings

As a result of this localization, locations were defined as groups of readings that have the same set of hotspots. If locations were allowed to be created from readings that were within a similarity measure, then locations would have spanned larger physical locations.

As the main goal of this research was to achieve a pattern of movement, there were a few necessary capabilities. The ability to track the transitions from one location to another was paramount, in addition to monitoring when an individual returned to a location that had already visited. The combination of these two enabled the creation of a pattern of movement to describe a person's timeline.

# Chapter 2

# Related Work

Signal strengths collected from various sources (cellular, WiFi, GPS, Bluetooth) have been used in positioning and localization, detecting movements, and generating mobility patterns.

Localization of the estimate position of mobile devices in a physical space, through the use of radio frequency and ultrasonic signals, was achieved in [38]. By using beacon devices installed throughout a building, that have known locations, triangulating the position of mobile devices was achieved.

Clustering and regression of anonymized cellular data was implemented in [18], to identify ”important places”, and discern locations that could be semantically labeled as ”home” and ”work”.

Using mobile phone traces, [8] added semantics by inferring individuals daily activities, using a model that was created using travel diary surveys.

While [44] and [18] were interested in mobility patterns to improve policies for communities and calculating carbon footprint, [3] is interested in the use of human mobility patterns to predict the direction and velocity of infectious diseases.

Cellular towers were also used in [44] to obtain mobility patterns of a population within a defined area. Triangulating their locations from cellular towers allowed for the establishment of mobility patterns that were based from an assumed ”home” position. The goal of this work was to discover if there were significant differences on a population’s ”home-based” mobility pattern in different communities.

Applications, such as Google Maps, generate a mobility history of a device, which is realized through the use of cloud web services and other cellular network based approaches [33], but privacy is not maintained.

Mobility patterns of anonymized mobile phone users across a user base was explored in [12, 42]. The analysis in [12], discovered that individuals follow simple mobility patterns. In [20], a new approach was suggested to investigate non-trivial behavior on smaller timer scales than in [42].

Algorithms combining an extended Kalman filter, approximate pattern matching and velocity vectors to predict future movements of users across cell boundaries was used in [31].

Cellular signals are only suitable for outdoor environments, as a mobile device would experience interference when indoors. Even when outdoors, this method is subject to noise from the signals being reflected by buildings. The issue of unpredictable effect of physical factors on signals was investigated by a propagation model for the relationship between signal and location by [11] and [2]. Cellular and WiFi interference was examined by [11], and [2] focused on WiFi signals.

RSSI from different sources was investigated for indoor localization in [30], and [10, 14, 24, 36, 45] focused on WiFi signals for indoor localization.

In both [40] and [27], user locations were modeled as states of a dynamic system with noisy observations of RSSI data from WiFi signals. An implementation of Bayesian filters called particle filters, estimated the system state—the user location—using probability. The initial training phase uses samples from predefined points to build a wireless sensor map of an environment divided into cells. Location is then estimated on a spatial connectivity graph. With each motion update step of the Bayesian filter, the user moves along an edge of the graph. As powerful as these probabilistic methods are, they have the potential of a high computational complexity, which is the drawback of particle filter methods. The worst-case complexity grows exponentially in the dimensions of the state space.

A physical area was divided into a grid in [39] and [41]. WiFi signal data was recorded from a set of fixed known points which was then used as training data to generate a probability distribution of signal strengths given the location values. With new signal observations, posterior distribution was computed and the highest probability is used to select a location. A grid-like manner was also used with WiFi signals in [29], with the data collection and off-line analysis phase being described. The off-line analysis phase required 22 minutes to process a 120 meter by 21 meter physical space containing multiple WiFi hotspots, which expresses the challenges in achieving localization of signals.

In [22,28,32], offline readings were collected first and then online readings were compared to them using a fingerprinting mode to find the physical position of mobile devices.

The WiFi and GPS data collected in [23] was analyzed for detecting location changes based upon signal strengths, and using diaries from the users in the case study, semantics were added.

The work in [6] focused on the power consumption of collecting GPS data vs WiFi data, and discovered a significant improvement on energy efficiency when using WiFi for data collection.

In [13], a ″origin-destination″ mobility pattern of a population was studied through a dataset containing taxi drivers GPS timeline. This work studies the general flow of a population of taxi passengers, but doesn't provide an individual's pattern which is the focus of the work in this project. The work in [21] also uses GPS data of taxi drivers, but also includes bus and subway data to analyze human mobility of a population.

In addition to cellular, WiFi and GPS, Bluetooth has also been used to achieve insights on a population's movement. Bluetooth signals were collected at a large event in [43] and [7] from various receivers, and from the data collected, offline analysis was executed to gain insights into visitors high level movement at the event. While [43] and [7] focuses on creating different profiles of visitors at the event given their movements, the work in this project focus on a significantly longer timeline that spans many locations that are discovered by an individual.

All of the related works mentioned require offline analysis and prior knowledge, while also putting privacy at risk or causing the need for anonymous data being used. The work in this thesis requires no prior knowledge and is achieved in an ad-hoc fashion. The algorithms presented in Chapter 3 have been published in [15], and the methods of evaluation and visualization of mobility patterns in Chapter 3 have been published in [16].

# Chapter 3

# Algorithms and Analytical Techniques

## 3.1 Hierarchical time segmentation

Clustering the timeline of readings, $\mathcal{T}_r$, via a binary tree hierarchy, $\mathcal{H}$, allowed for the maintenance of clusters of readings, which were then used to achieve localization. This further allowed for the generation of a mobility pattern. A hierarchy also provided a multi-resolution view of the locations discovered by the device.

The hierarchy, $\mathcal{H}$, was defined as a binary tree like structure made up of clusters, $c$, where the leaf clusters are the readings, $r$, from the timeline of readings, $\mathcal{T}_r$.

Hierarchical clustering requires a similarity measure for choosing optimal pairs to form new clusters. The similarity measure, $sim$, used for comparing clusters of readings was given in the definition of 1. $sim$ made use of Jaccard similarity [26], comparing the sets of BSSIDs, $\mathcal{B}$. Those were sourced from the creation of a super set of all BSSID observed in the readings. A similarity result of 0 indicates that the two clusters share no BSSIDs, and are two disjointed sets; whereas a similarity of 1 indicates that the two clusters are exactly the same.

**Definition 1** *Similarity function using Jaccard [26]*

$c_0, c_1$ *- clusters of readings.*

*Returns a number between 0 and 1, where 0 indicates the two clusters are completely dissimilar, and 1 being the two clusters are the same.*

$$sim(c_0, c_1) = \text{Jaccard}\left((\mathcal{B}(c_0), \mathcal{B}(c_1)\right) = \frac{|\mathcal{B}(c_0) \cap \mathcal{B}(c_1)|}{|\mathcal{B}(c_0) \cup \mathcal{B}(c_1)|}$$

### 3.1.1 Traditional hierarchical clustering

The traditional hierarchical clustering approach provides the localization desired, and it is succinctly described by [26] in Algorithm 1. This approach continuously clusters the most similar pair of clusters until there is only one cluster left, which would then be the root of the hierarchy. The clusters obtained provide groups of hotspots that have been frequently observed together, and therefore are in close proximity to one another in their physical environment. The traditional approach falls victim to the complexity of $O(n^2)$ with $n$ being the number of readings in the timeline.

```
while it is not time to stop do
    pick the best two clusters to merge
    combine those two clusters into one cluster
end
```

**Algorithm 1:** Traditional hierarchical approach [26]

In this usage case, the number of readings increases with time. So, for example, if one reading were taken each minute, there would be 1440 readings per day, and 10 080 readings per week. For this reason alone, the traditional approach is not suitable for this project.

Figure 3.1 illustrates that localization was achieved, but segmenting the timeline in an efficient manner still remains to be done.

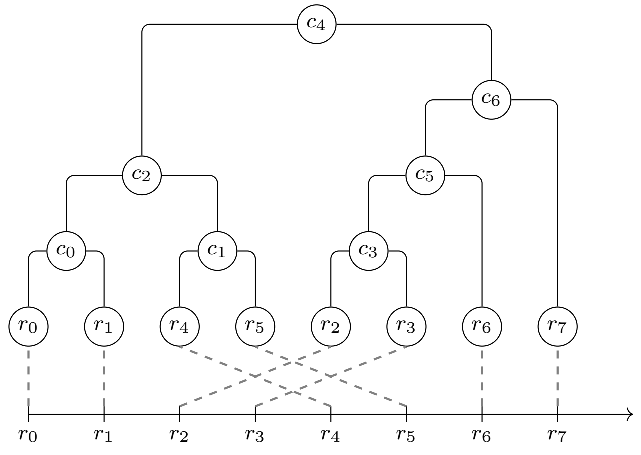

$c_4$
$c_6$
$c_2$
$c_5$
$c_0$
$c_1$
$c_3$
$r_0$ $r_1$ $r_4$ $r_5$ $r_2$ $r_3$ $r_6$ $r_7$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$

Figure 3.1: Traditional Hierarchical Clustering

### 3.1.2 New hierarchical clustering approach: *ht-append*

A new hierarchical clustering algorithm, *ht-append*, is proposed here, shown as Algorithm 2. This algorithm incrementally builds up the hierarchy as readings are being taken, maintains the original order of readings, and has the potential of a very low runtime on real data as it has an average case complexity of $O(log(n))$. This hierarchy that sits on the timeline, is denoted as $\mathcal{H}_T$. Figure 3.2 illustrates what the proposed algorithm would achieve on the same set of data as Figure 3.1. Figure 3.1 shows that readings $r_0$, $r_1$, $r_4$, $r_5$ belong to a single location, and readings $r_2$, $r_3$, $r_6$, $r_7$ belong to another location, but the movement from one location to another, and the return to the previous location is lost in the traditional method of clustering. In this approach, readings $r_0$ and $r_1$ get clustered into $c_0$, $r_2$ and $r_3$ into cluster $c_1$, $r_4$ and $r_5$ into cluster $c_3$, and $r_6$ and $r_7$ into cluster $c_5$. Further analysis would demonstrate that $c_0 \approx c_3$ and $c_1 \approx c_5$, which would allow obtaining a mobility pattern describing the movement from one location to another and back again.

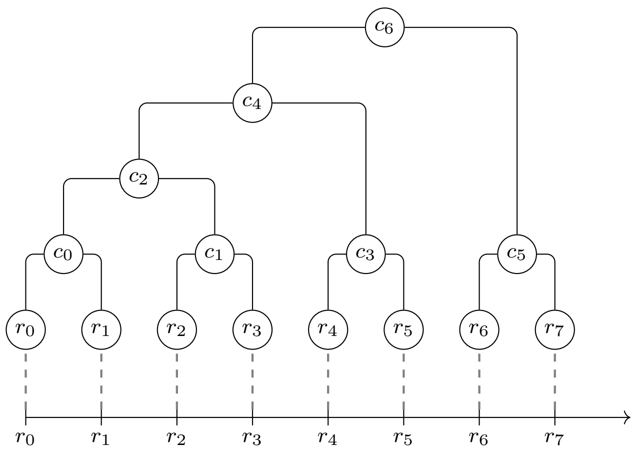


Figure 3.2: Proposed hierarchical time clustering using *ht-append*

The *ht-append*(Algorithm 2) has two restrictions in addition to the traditional hierarchical clustering approach:

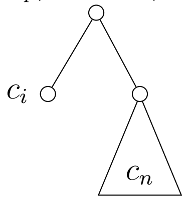
1. Clusters can only be made from consecutive clusters
2. New clusters can only be added along the right-most branch of the hierarchy

The first restriction is to maintain order of the readings in the timeline, and the second restriction is to ensure the hierarchy is incrementally created in an online fashion.

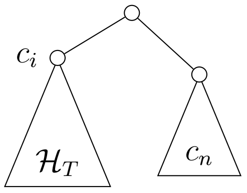
When a new reading is being appended to the hierarchy using *ht-append*(Algorithm 2), a new cluster is created, $c_n$, containing the information from the reading. The initial insertion point, $c_i$, is initialized to the rightmost cluster in the hierarchy which is found by traversing the hierarchy from the root, always returning the right child cluster, until it reaches a leaf cluster, which then gets returned. $c_p$ was set to the cluster previous to $c_i$, which happens to be $c_i$'s sibling cluster.

If the $sim(c_i, c_n) > sim(c_p, c_i)$, then a new cluster was created $c'_n$, which had the left child of $c_i$, and the right child being $c_n$. An updated hierarchy, $\mathcal{H}^*_T$, is created with the original parent to $c_p$ and $c_i$, replaced with $c_p$, and then new cluster $c'_n$ is

```
where: $\mathcal{H}_T$ - existing hierarchy
       $c_i$ - insertion point (a cluster in $\mathcal{H}_T$)
       $c_n$ - a cluster to be inserted after $c_i$
result : $\mathcal{H}'_T$ - the updated hierarchy
ht-append($\mathcal{H}_T$, $c_i$, $c_n$)
    if $c_i <> \mathbf{root}(\mathcal{H}_T)$ then
        $c_p = prev(c_i|\mathcal{H}_T)$
        if $sim(c_i, c_p) < sim(c_i, c_n)$ then
            $c'_n =$
```



```
            $\mathcal{H}^*_T = \mathbf{replace}(\mathcal{H}_T, \mathbf{parent}(c_i), c_p)$
            $\mathcal{H}'_T = ht\text{-}append(\mathcal{H}^*_T, c_p, c'_n)$
        else
            $\mathcal{H}'_T = ht\text{-}append(\mathcal{H}_T, \mathbf{parent}(c_i), c_n)$
        end
    else
        $\mathcal{H}'_T =$
```



```
    end
```

**Algorithm 2:** Hierarchy time append function: *ht-append*

then attempted to be inserted at $c_p$, with a recursive call to the ht-append function; $ht\text{-}append(\mathcal{H}^*_T, c_p, c'_n)$.

If the similarity of the new tree to the insertion point cluster, $sim(c_i, c_n) <= sim(c_p, c_i)$, then $c_n$ is attempted to be inserted at the parent of $c_i$, through a recursive call to the ht-append function; $ht\text{-}append(\mathcal{H}_T, \mathbf{parent}(c_i), c_n)$.

In this fashion, the hierarchy is traversed along the rightmost path towards the root. When the insertion point $c_i$ becomes the root, a new root cluster must be created, with the original root cluster $c_i$ to be the left child, and the new cluster $c_n$, to be inserted as the right child.

The hierarchy, $\mathcal{H}_T$, sits on the timeline of readings, $\mathcal{T}_r$, and if a person returned to a location they had frequented before, there would be separated clusters in the hierarchy. In a traditional hierarchical clustering algorithm, these clusters would have been clustered together. As they were not created from consecutive readings, they are separated by other branches of clusters.

With the hierarchy in place, a multi-resolution representation of clusters, and their place on the timeline, would be observable.

Each cluster of the hierarchy represents a segment of the timeline which is obtained by extracting the leaves of a cluster. The start and end time of the segment can be achieved by traversing towards the leaves returning the left-most and right-most leaves respectively.

### Visual example of *ht-append*

The movement example from Figure 1.2 can supply the readings to the *ht-append* function (Algorithm 2). Figure 3.3 shows that a new root is being created for every new reading, since $sim(r_i, r_{i-1}) < sim(r_{i-1}, prev(r_{i-1}))$.

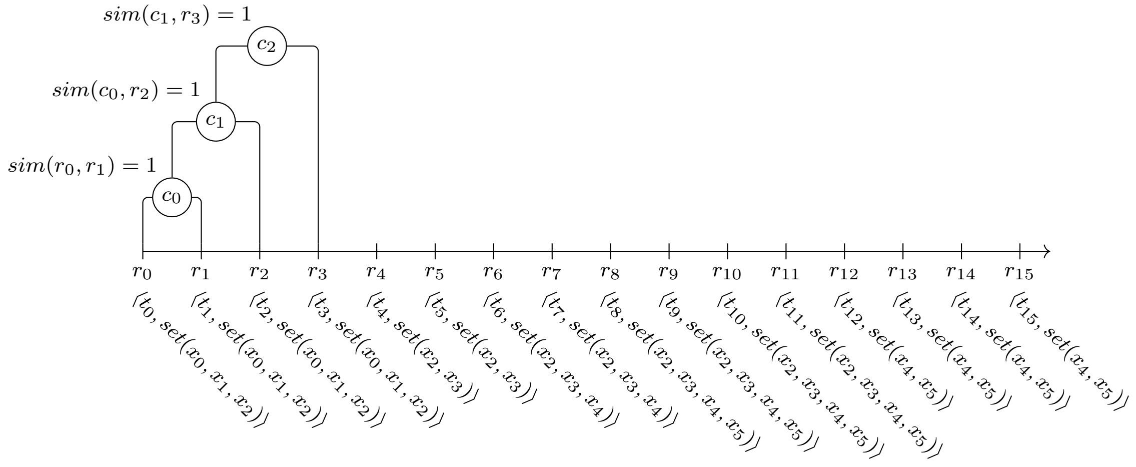

$sim(c_1, r_3) = 1$
$sim(c_0, r_2) = 1$
$sim(r_0, r_1) = 1$
$c_2$
$c_1$
$c_0$
$r_0$
$r_1$
$r_2$
$r_3$
$r_4$
$r_5$
$r_6$
$r_7$
$r_8$
$r_9$
$r_{10}$
$r_{11}$
$r_{12}$
$r_{13}$
$r_{14}$
$r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.3: Clustering the first 4 readings of the timeline

The first four readings in Figure 3.3, all have the same set of hotspots, and therefore a new root is created in every case. When $r_4$ is appended to the hierarchy in Figure 3.4, a new root $c_3$ is created, even though $sim(r_4, c_2) = 0.25$, because $c_2$ is the root of the hierarchy, a new root is created.

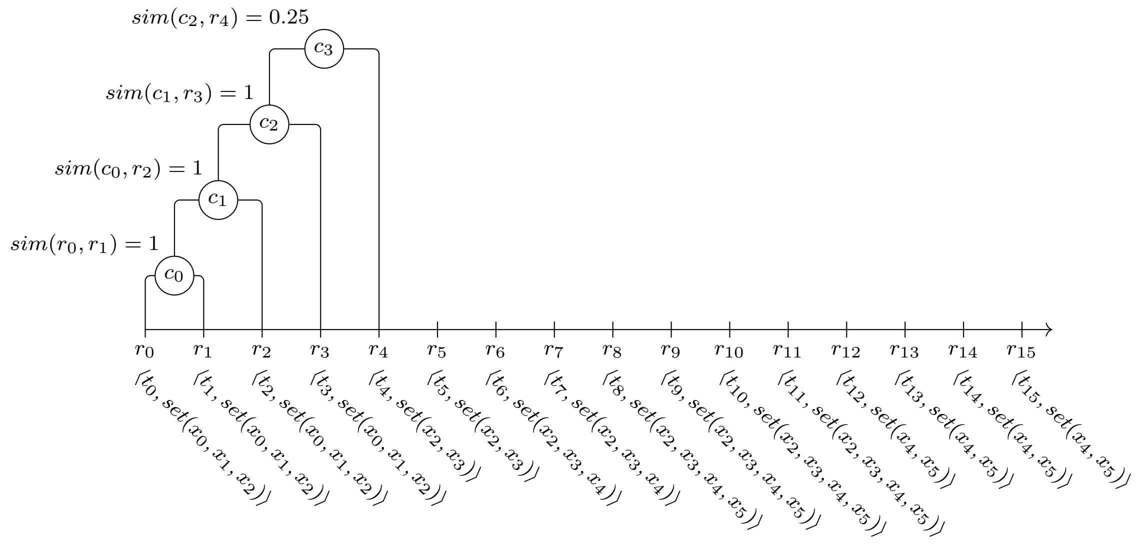

$sim(c_2, r_4) = 0.25$
$sim(c_1, r_3) = 1$
$sim(c_0, r_2) = 1$
$sim(r_0, r_1) = 1$
$c_3$
$c_2$
$c_1$
$c_0$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.4: Clustering the 6th reading of the timeline

When $r_5$ is appended to the hierarchy, the first case of having to break a cluster from the main hierarchy appears (Figure 3.5). Since $sim(r_4, r_5) = 1$, and $sim(r_4, c_2) = 0.25$, $r_4$ is detached from the hierarchy, and replace $c_3$ with $c_2$. A new cluster $c_4$ is created with the left child being $r_4$ and the right being $r_5$, and then $c_4$ is then attempted to be inserted at $c_2$ (currently the root) which then causes a new root cluster $c_5$ to be created.

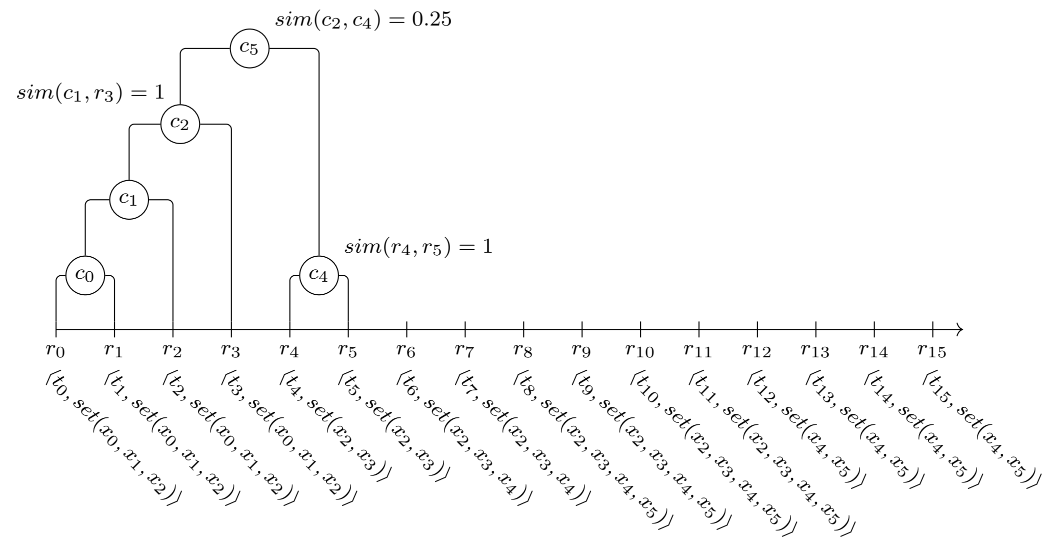

$sim(c_2, c_4) = 0.25$
$sim(c_1, r_3) = 1$
$sim(r_4, r_5) = 1$
$c_5$
$c_2$
$c_1$
$c_0$
$c_4$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.5: Clustering the 6th reading of the timeline

When $r_6$ is attempted to be inserted at $r_5$, as seen in figure 3.6, $sim(r_5, r_6) = 0.67$ is smaller than $sim(r_4, r_5) = 1$. As a result, $r_6$ is then attempted to be inserted at the parent of $r_5$ which is $c_4$. The value of $sim(c_4, r_6)$ is also 0.67, which is greater than the $sim(c_2, c_4) = 0.25$, which causes $c_4$ to be broken from the hierarchy and added as the left child of a new cluster $c_6$. Cluster $c_2$ is elevated to the root once again, and when $c_6$ is attempted to be inserted at $c_2$, a new root $c_7$ is created.

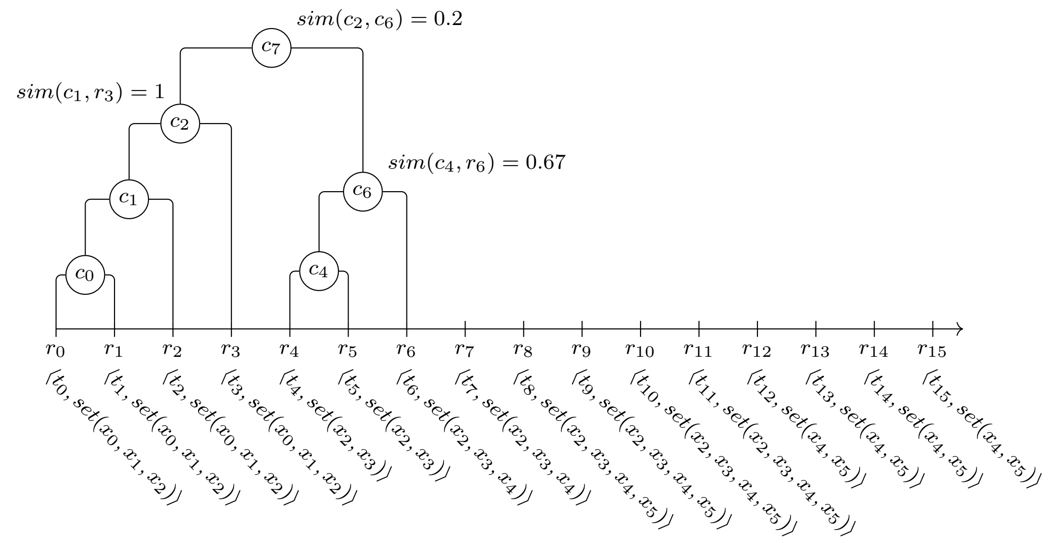
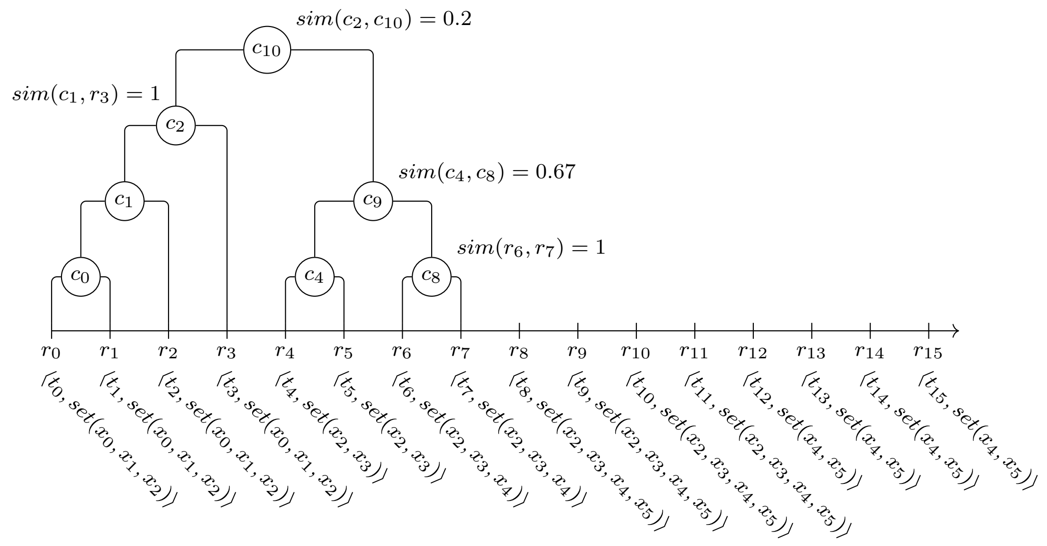

$sim(c_2, c_6) = 0.2$
$sim(c_1, r_3) = 1$
$sim(c_4, r_6) = 0.67$
$c_7$
$c_2$
$c_1$
$c_6$
$c_0$
$c_4$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.6: Clustering the 7th reading of the timeline

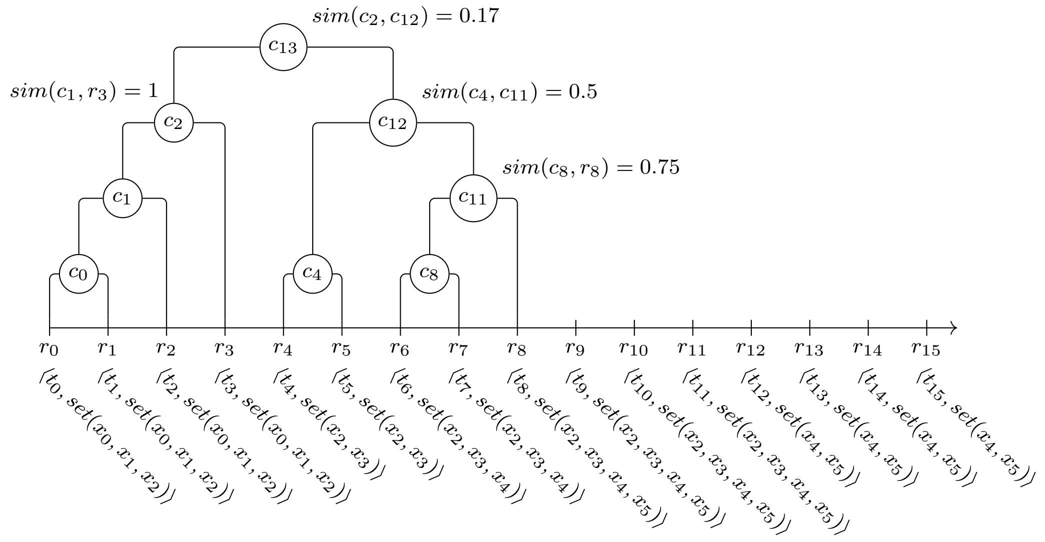
Inserting $r_7$ into the hierarchy is illustrated in figure 3.7, the previous reading must be disconnected from the hierarchy and create a new cluster $c_8$. The cluster $r_6$ was previously clustered with, $c_4$, replaces $c_6$, and the new $c_8$ cluster is attempted to be inserted at $c_4$. The similarity comparisons of $sim(c_2, c_4) = 0.25$ and $sim(c_4, c_8) = 0.67$, cause $c_4$ to be detached from the hierarchy and added as the left child of a new cluster, $c_9$, and the right child being $c_8$. The root, $c_7$, is then replaced with $c_2$ once again. When $c_9$ is attempted to be added at $c_2$, new root $c_{10}$ is compulsory.



$sim(c_2, c_{10}) = 0.2$
$sim(c_1, r_3) = 1$
$sim(c_4, c_8) = 0.67$
$sim(r_6, r_7) = 1$
$c_{10}$
$c_2$
$c_1$
$c_9$
$c_0$
$c_4$
$c_8$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.7: Clustering the 8th reading of the timeline

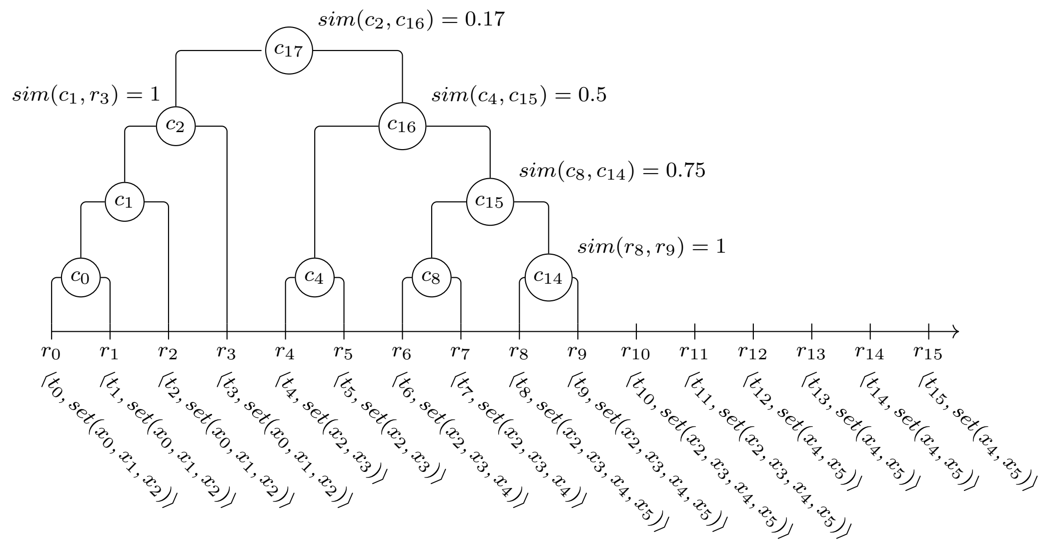
When appending the 9th reading, $r_8$ to the hierarchy, as illustrated in Figure 3.8, the new reading at $r_7$ was first attempted, and compared $sim(r_7, r_8) = 0.75$ with $sim(r_6, r_7) = 1$ which forced the attempt to insert at the parent of $r_7$, which is $c_8$. The $sim(c_8, r_8) = 0.75$ and $sim(c_4, c_8) = 0.67$ causes the detachment $c_8$ from the hierarchy and place it in the left child of a new cluster $c_{11}$ with the right child being $r_8$. The cluster, $c_4$, replaces $c_9$. The insertion of $c_{11}$ at $c_4$ was attempted, which succeeds and causes us to detach $c_4$ from the hierarchy, and elevate $c_2$ to become the root once again. A new cluster, $c_{12}$ is created with the children $c_4$ and $c_{11}$, and is attempted to be inserted at $c_2$, which being the root once again, forced the creation of another new root, $c_{13}$.



$sim(c_2, c_{12}) = 0.17$
$c_{13}$
$sim(c_1, r_3) = 1$
$c_2$
$c_{12}$
$sim(c_4, c_{11}) = 0.5$
$sim(c_8, r_8) = 0.75$
$c_1$
$c_{11}$
$c_0$
$c_4$
$c_8$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.8: Clustering the 9th reading of the timeline

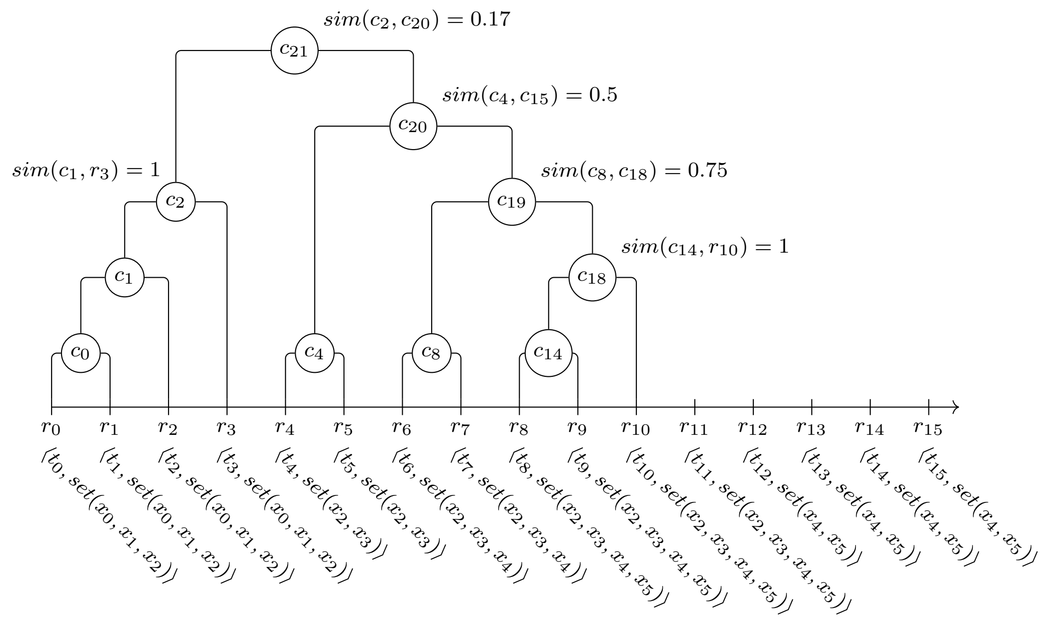
Inserting the 10th reading, $r_9$, in figure 3.9, causes $r_8$ to be detached from $c_{11}$ due to the fact that $sim(r_8, r_9) = 1$, causing $c_8$ to replace $c_{11}$, and the new cluster, $c_{14}$, being formed with the children $r_8$ and $r_9$. The new cluster $c_{14}$ is then attempted to be inserted at $c_8$. With the comparisons of $sim(c_4, c_8) = 0.67$ and $sim(c_8, c_{14}) = 0.75$, $c_8$ detaches from the hierarchy, and is added to a new cluster $c_{15}$ along with $c_{14}$. The cluster $c_4$ gets elevated to replace $c_{12}$. The new cluster $c_{15}$ is then attempted to be inserted at $c_4$, which is successful since $sim(c_4, c_{15}) > sim(c_2, c_4)$. Cluster $c_4$ is detached from the hierarchy and with $c_{15}$ create a new cluster $c_{16}$, causing $c_2$ to be elevated to the root. When $c_{16}$ is attempted to be inserted at $c_2$, a new root cluster, $c_{17}$ is created.



$sim(c_2, c_{16}) = 0.17$
$c_{17}$
$sim(c_1, r_3) = 1$
$c_2$
$c_{16}$
$sim(c_4, c_{15}) = 0.5$
$c_1$
$c_{15}$
$sim(c_8, c_{14}) = 0.75$
$sim(r_8, r_9) = 1$
$c_0$
$c_4$
$c_8$
$c_{14}$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.9: Clustering the 10th reading of the timeline

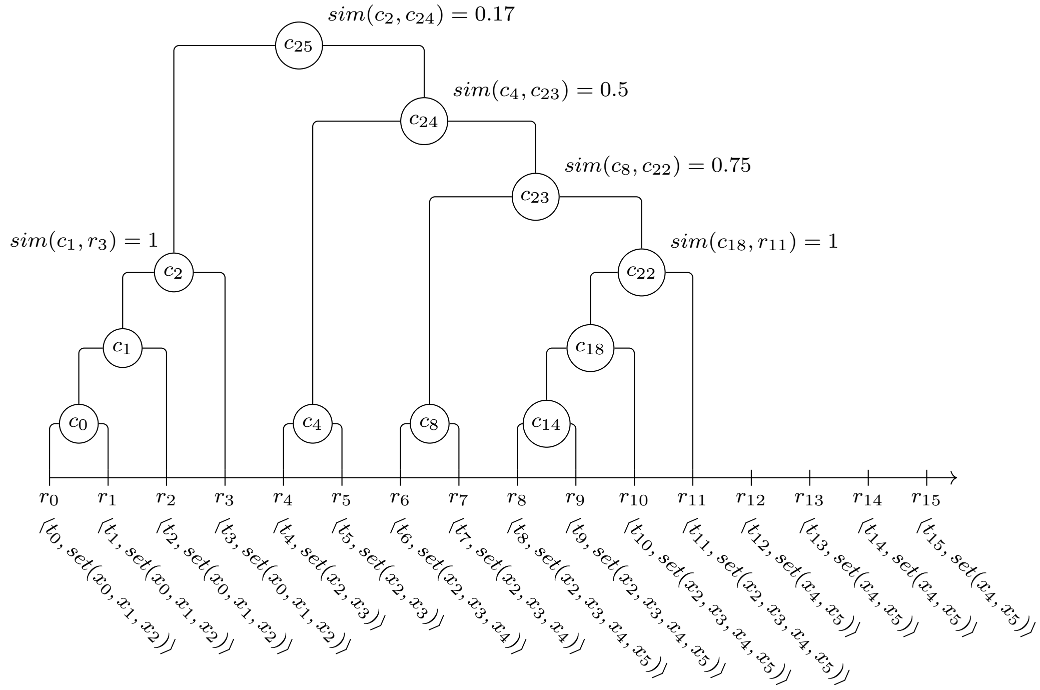
Inserting the 11th reading, $r_{10}$, in figure 3.10, will result in $r_{10}$ being combined with $c_{14}$ to create $c_{18}$, since it can't beat the similarity $sim(r_8, r_9) = 1$. Cluster $c_{19}$ is created by combining $c_8$ and $c_{18}$, $c_{20}$ is created by combining $c_4$ and $c_{19}$, and then $c_{21}$ is created by combining $c_2$ and $c_{20}$, and is the new root.



$sim(c_2, c_{20}) = 0.17$
$c_{21}$
$sim(c_4, c_{15}) = 0.5$
$c_{20}$
$sim(c_1, r_3) = 1$
$c_2$
$sim(c_8, c_{18}) = 0.75$
$c_{19}$
$sim(c_{14}, r_{10}) = 1$
$c_1$
$c_{18}$
$c_0$
$c_4$
$c_8$
$c_{14}$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.10: Clustering the 11th reading of the timeline

Figure 3.11, denotes the clustering of the 12th reading, $r_{11}$. Similar to the behavior of adding $r_{10}$, the top cluster that had a similarity of 1 among the children was broken off and combined it with $r_{11}$ to create a new cluster $c_{22}$. The insertion of $c_{22}$ at cluster $c_8$ was successful, which replaced $c_{19}$, since $sim(c_8, c_{22}) > sim(c_4, c_8)$. The new cluster $c_{23}$ is created from $c_8$ and $c_{22}$, and then attempted to be inserted at $c_4$, which is accepted since $sim(c_8, c_{23}) > sim(c_2, c_4)$. Cluster $c_{24}$ is created from $c_4$ and $c_{23}$, and $c_2$ becomes the root. When the attempt to insert $c_{24}$ at the root, $c_2$ was made, and a new root was generated, $c_{25}$.



$sim(c_2, c_{24}) = 0.17$
$c_{25}$
$sim(c_4, c_{23}) = 0.5$
$c_{24}$
$sim(c_8, c_{22}) = 0.75$
$c_{23}$
$sim(c_1, r_3) = 1$
$sim(c_{18}, r_{11}) = 1$
$c_2$
$c_{22}$
$c_1$
$c_{18}$
$c_0$
$c_4$
$c_8$
$c_{14}$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.11: Clustering the 12th reading of the timeline

Figure 3.12 shows the resulting hierarchy after readings $r_{12}$ to $r_{15}$ are appended to the hierarchy.

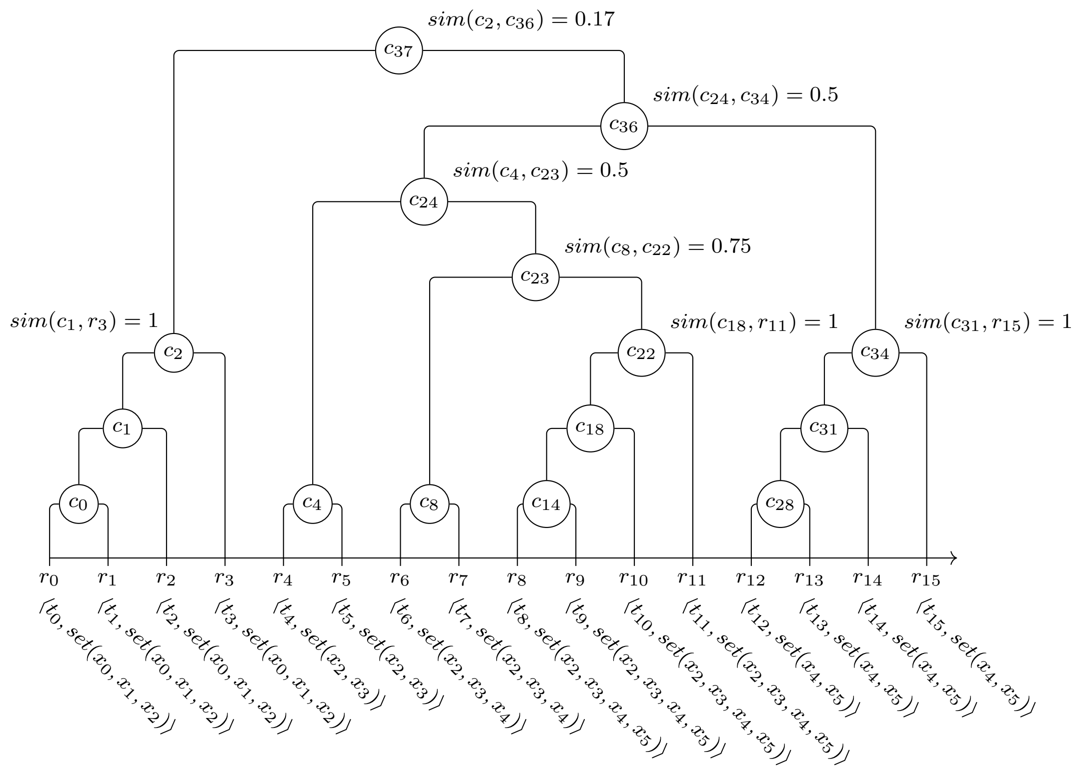

$sim(c_2, c_{36}) = 0.17$
$sim(c_{24}, c_{34}) = 0.5$
$sim(c_4, c_{23}) = 0.5$
$sim(c_8, c_{22}) = 0.75$
$sim(c_1, r_3) = 1$
$sim(c_{18}, r_{11}) = 1$
$sim(c_{31}, r_{15}) = 1$
$c_{37}$ $c_{36}$ $c_{24}$ $c_{23}$ $c_2$ $c_{22}$ $c_{34}$ $c_1$ $c_{18}$ $c_{31}$ $c_0$ $c_4$ $c_8$ $c_{14}$ $c_{28}$
$r_0$ $r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$ $r_7$ $r_8$ $r_9$ $r_{10}$ $r_{11}$ $r_{12}$ $r_{13}$ $r_{14}$ $r_{15}$
$\langle t_0, set(x_0, x_1, x_2)\rangle$
$\langle t_1, set(x_0, x_1, x_2)\rangle$
$\langle t_2, set(x_0, x_1, x_2)\rangle$
$\langle t_3, set(x_0, x_1, x_2)\rangle$
$\langle t_4, set(x_2, x_3)\rangle$
$\langle t_5, set(x_2, x_3)\rangle$
$\langle t_6, set(x_2, x_3, x_4)\rangle$
$\langle t_7, set(x_2, x_3, x_4)\rangle$
$\langle t_8, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_9, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{10}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{11}, set(x_2, x_3, x_4, x_5)\rangle$
$\langle t_{12}, set(x_4, x_5)\rangle$
$\langle t_{13}, set(x_4, x_5)\rangle$
$\langle t_{14}, set(x_4, x_5)\rangle$
$\langle t_{15}, set(x_4, x_5)\rangle$

Figure 3.12: Clustering the 13th to 16th reading of the timeline

### 3.1.3 Segmenting the hierarchy timeline: *ht-segment*

Using *ht-segment* (algorithm 3), the hierarchy of clustered readings on the timeline, $\mathcal{H}_T$ (the result of *ht-append*) was taken. this was turned it into a list of clusters that have the similarity of their children meet a minimum threshold, $\sigma$. The clusters then represent segments, and the list of cluster segments, is a timeline of segments,

$\mathcal{T}_s$.

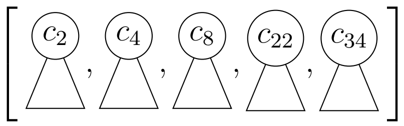
```
where: $c$ - cluster in the hierarchy
       $\sigma$ - similarity threshold
result : A sequence of clusters which represent segments of the timeline

ht-segment(c, σ)
    if $sim(left(c), right(c)) < \sigma$ then
        return [ ht-segment(left(c), σ) + ht-segment(right(c), σ) ]
    else
        return [ c ]
    end
```

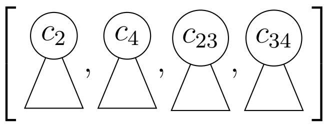
**Algorithm 3:** Segment function: *ht-segment*

If the resulting hierarchy from Figure 3.12 was able to be input in the *ht-segment* function with a similarity threshold $\sigma$, of 1, 0.75 and 0.5, the result would be:

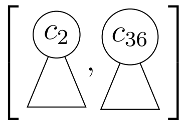
$$\mathcal{T}_s = \textit{ht-segment}(c_{37}, 1) =$$



$$\mathcal{T}_s = \textit{ht-segment}(c_{37}, 0.75) =$$



$$\mathcal{T}_s = \textit{ht-segment}(c_{37}, 0.5) =$$



Continuing to utilize the hierarchy structure of the clusters and readings, all of the information in the readings is able to be extracted, and new information was derived (start time, end time and duration), for each cluster segment. The start time is accessed through traversing from the root of a cluster to the left-most leaf node which would be the first reading in the cluster, and conversely, the end time is available by traversing to the right-most cluster. Obtaining the duration is then a matter of subtracting the start time from the end time. It is also still possible to obtain the set of hotspots that the cluster encompasses and even retrieve the signal strength for each hotspot in each reading.

## 3.2 Locations and semantic grouping

### 3.2.1 Physical locations

A physical location, $l_p$, is a list of clusters, where each cluster is obtained from the timeline of segments, $\mathcal{T}_s$, resulting from *ht-segment* (Algorithm 3). Each cluster within a physical location must meet the physical location similarity threshold, $\rho$, with at least one other cluster within the physical location as per algorithm *loc-physical* (Algorithm 4), which iterates through the segments and returns a list of physical locations, $\mathcal{L}_p$. The similarity measure only takes into account the set of BSSID values within a cluster, $\mathcal{B}$, which is why we refer to these as physical locations.

Figure 3.13 illustrates *loc-physical*taking a timeline of segments and grouping them based upon their BSSID sets.

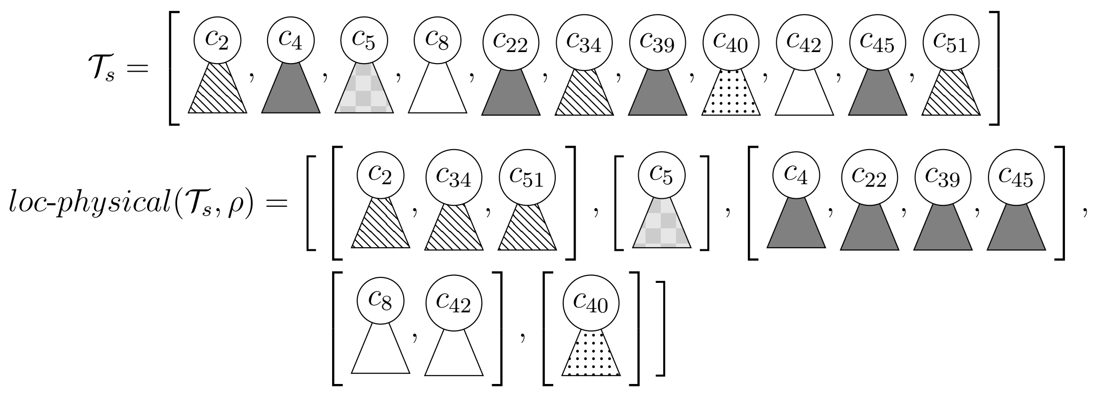

$\mathcal{T}_s =$
$c_2$, $c_4$, $c_5$, $c_8$, $c_{22}$, $c_{34}$, $c_{39}$, $c_{40}$, $c_{42}$, $c_{45}$, $c_{51}$
$loc\text{-}physical(\mathcal{T}_s, \rho) =$
$c_2$, $c_{34}$, $c_{51}$
$c_5$
$c_4$, $c_{22}$, $c_{39}$, $c_{45}$
$c_8$, $c_{42}$
$c_{40}$

Figure 3.13: Physical location identification example

```
where: $\mathcal{T}_s$ - list of clusters that represent segments of the timeline
       $\rho$ - physical location similarity threshold
result : $\mathcal{L}_p$ - A collection or list of physical locations, $l_p$
loc-physical($\mathcal{T}_s$, $\rho$)
    $\mathcal{L}_p$= empty list
    for $c_s$ in $\mathcal{T}_s$ do
        found = False
        for $l_p$ in $\mathcal{L}_p$ do
            for $c_l$ in $l_p$ do
                if $sim(c_s, c_l) >= \rho$ then
                    found = True
                    append $c_s$ to $l_p$ in $\mathcal{L}_p$
                end
            end
        end
        if found = False then
            $l_p$= [ $c_s$ ]
            append $l_p$ to $\mathcal{L}_p$
        end
    end
```
**Algorithm 4:** Physical location identification: *loc-physical*

### 3.2.2 Semantic locations

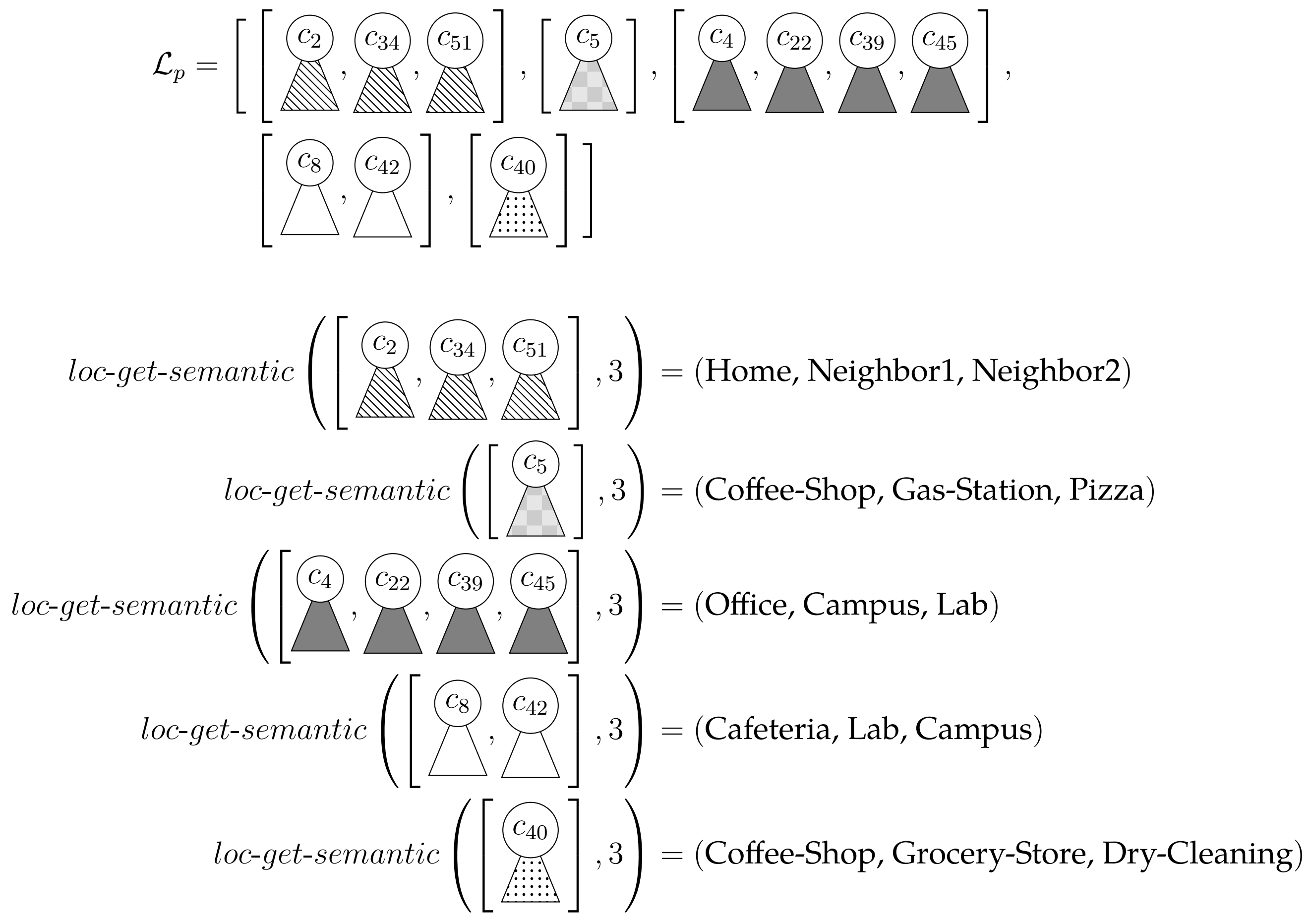
Semantic locations, $l_s$, are a set of $\mathcal{SSID}$ values that are generated from physical locations. A collection or list of semantic locations is denoted by $\mathcal{L}_s$. The set of the top $k$, $\mathcal{SSID}$ values with the highest signal strength sum over all the readings contained at the leaf clusters over all clusters that make up a physical location, becomes the semantic location. For example, if a semantic location for a physical location (a person's home) was generated, the top SSID value, based upon the sum of signal strengths, would be the SSID from their home WiFi hotspot. The next top SSID values may be that of the closest neighbors. This semantic location, made from the top 3 SSID values, would look something like $\langle ''MyHome'', ''Neighbor1'', ''Neighbor2''\rangle$. The generation of the semantic SSID set is described in *loc-get-semantic* (Algorithm 5).

```
where: $l_p$ - physical location comprised of a list of clusters
       $k$ - number of SSID values to make up each physical location
result : $l_s$ - semantic location which is a set of SSID values
loc-get-semantic($l_p$, k)
    $K$ = empty list of SSID-strength pairs
    for $c$ in $l_p$ do
        for $r$ in readings($c$) do
            for $\langle x, s \rangle$ in $r$ do
                $K[ssid(x)]$ += $s$
            end
        end
    end
    sort SSID-strength pairs in $K$ by their strength sum in descending order
    $l_s$= set($ssid$ of first $k$ elements in $K$)
```
**Algorithm 5:** Generate semantic location from physical location: *loc-get-semantic*

Algorithm *loc-semantic* (Algorithm 6) utilizes *loc-get-semantic* (Algorithm 5) to convert the physical locations to semantic locations, by iterating over the list of physical locations, $\mathcal{L}_p$, generating a set of the top SSID values, and then appending it to the list of semantic locations, $\mathcal{L}_s$.

```
where: $\mathcal{L}_p$ - list of physical locations
       $k$ - number of SSID values to make up each key
result : $\mathcal{L}_s$ - list of semantic locations
loc-semantic($\mathcal{L}_p$, k)
    $\mathcal{L}_s$= empty list of semantic locations
    for $l_p$ in $\mathcal{L}_p$ do
        $l_s$ = loc-get-semantic($l_p$, k)
        append $l_s$ to $\mathcal{L}_s$
    end
```
**Algorithm 6:** Generate list of semantic locations: *loc-semantic*

If the same example from Figure 3.13 were expanded upon, and use the output of *loc-physical* as the input variable $\mathcal{L}_p$, with the value of 3 for the size of the key set, then when using *loc-semantic*, Figure 3.14 is the result. In this example, some of the physical locations have similar SSID values, or there are hotspots that belong to more than one physical location discovered by the hierarchical segmenting algorithms.


$\mathcal{L}_p =$
$c_2$, $c_{34}$, $c_{51}$
$c_5$
$c_4$, $c_{22}$, $c_{39}$, $c_{45}$
$c_8$, $c_{42}$
$c_{40}$
loc-get-semantic
, 3
= (Home, Neighbor1, Neighbor2)
= (Coffee-Shop, Gas-Station, Pizza)
= (Office, Campus, Lab)
= (Cafeteria, Lab, Campus)
= (Coffee-Shop, Grocery-Store, Dry-Cleaning)

$$\begin{aligned}\textit{loc-semantic}(\mathcal{L}_p, 3) = \big[&(\text{Home, Neighbor1, Neighbor2}), (\text{Coffee-Shop, Gas-Station, Pizza}),\\ &(\text{Office, Campus, Lab}), (\text{Cafeteria, Lab, Campus}),\\ &(\text{Coffee-Shop, Grocery-Store, Dry-Cleaning})\big]\end{aligned}$$

Figure 3.14: Physical to semantic location conversion

This research aims to further group the semantic locations based upon SSID to ensure that a general area that may produce multiple physical locations due to multiple hotspots providing WiFi coverage to the large area, get grouped together to provide a more succinct collection of semantic locations.

Currently, the semantic locations are made up of $k$ number of the top SSID values, and merging semantic locations should be done. Semantic locations that have values that share $j$ number of SSID values, where $j$ is $0 < j <= k$, get merged. Either a list of all possible pair combinations of SSIDs, that have been seen together in the semantic location collection, is created, or simply a list of all unique SSIDs that appear in the semantic locations. These combinations are called *keys*. The singleton-keys for the example illustrated in 3.14 would be the list in Figure 3.15, and the paired-keys would be in Figure 3.16.

singleton-keys = [(Home), (Neighbor1), (Neighbor2), (Coffee-Shop), (Gas-Station), (Pizza), (Office), (Campus), (Lab), (Cafeteria), (Grocery-Store), (Dry-Cleaning) ]

Figure 3.15: A list of singleton keys

paired-keys = [(Home, Neighbor1), (Home, Neighbor2), (Neighbor1, Neighbor2), (Coffee-Shop, Gas-Station), (Coffee-Shop, Pizza), (Gas-Station, Pizza), (Office, Campus), (Office, Lab), (Campus, Lab), (Cafeteria, Lab), (Cafeteria, Campus), (Coffee-Shop, Grocery-Store), (Coffee-Shop, Dry-Cleaning), (Grocery-Store, Dry-Cleaning)]

Figure 3.16: A list of paired keys

Once the list of *keys* is generated, the collection of semantic locations are reduced with the *loc-reduce* function (Algorithm 7), to check for optimal keys to merge on using *optimal-key* (Algorithm 8), which also returns a *weight*. So long as the *weight* of the *key* returned is greater than 1, it means that the optimal key is a subset of two or more semantic locations. The *loc-merge* function that is used in *loc-reduce*, creates a new list of semantic locations, removing all locations that match the optimal key, and then adding the optimal key to the locations as the replacement.

```
where: $\mathcal{L}_s$ - list of semantic locations
       $K$ - list of key permutations
result : $\mathcal{L}_s$ - reduced list of semantic locations
loc-reduce($\mathcal{L}_s$, $K$)
    $\langle key, weight \rangle = optimal\text{-}key(\mathcal{L}_s, K)$
    while $weight > 1$ do
        $\mathcal{L}_s = loc\text{-}merge(\mathcal{L}_s, key)$
        $\langle key, weight \rangle = optimal\text{-}key(\mathcal{L}_s, K)$
    end
```

**Algorithm 7:** Aggregate semantic locations: *loc-reduce*

Carrying on with the resulting $\mathcal{L}_s$ from the example in Figure 3.14, the loc-reduce (algorithm 7) is demonstrated by using the paired-keys list as $K$ in Figure 3.17. In this example, only one merge occurred. The locations; (Office, Campus, Lab) and (Cafeteria, Lab, Campus), were merged into the location (Campus, Lab), because both locations shared the key pair (Campus, Lab) in the list of key pair permutations. This merge has made the semantic locations a little more generalized to reflect a group of semantically related locations.

Executing loc-reduce on the same, original set of semantic locations, but using the singleton-keys, we achieve the example in Figure 3.18. This example still has the two locations that have (Campus, Lab) as a subset, merged; however, the new key that represents it is (Campus), due to the fact that (Campus) appears before (Lab) in the list of keys in singleton-keys. In addition to the (Campus) merge, two distinct Coffee-Shops have been merged. These may very well have been two different physical locations on opposite ends of town, but because the Coffee-Shop SSIDs are very strong in both physical locations. The person can be considered as having been at those Coffee-Shops rather than at the Grocery-Store or Gas-Station. This is optimal behavior for our system, as the interest lies within a person's mobility pattern. If a person is frequently visiting a Coffee-Shop, it would be beneficial to express all Coffee-Shop locations as one location to gain a more succinct pattern of

```
where: $\mathcal{L}_s$ - list of semantic locations
       $K$ - list of key permutations
result : $key_0$ - key with the largest weight
       $weight_0 - weight$

optimal-key($\mathcal{L}_s$, K)
    $key_0 = empty$
    $weight_0 = 0$
    for $key\ in\ K$ do
        $weight = 0$
        for $l_s\ in\ \mathcal{L}_s$ do
            if $key \subset l_s$ then
                $weight = weight + 1$
            end
        end
        if $weight > weight_0$ then
            $key_0 = key$
            $weight_0 = weight$
        end
    end
```
**Algorithm 8:** Find the optimal key to merge semantic locations on: optimal-key

the person's habits.

## 3.3 Mobility patterns

With the segmented timeline, $\mathcal{T}_s$, obtained from $ht\text{-}segment$ (Algorithm 3), and the resulting collection of semantic locations, $\mathcal{L}_s$, from Algorithm 7, iterations over the segmented timeline, $\mathcal{T}_s$, can be made to create a timeline of semantic locations $\mathcal{T}_l$. We use $loc\text{-}timeline$(Algorithm 10) to iterate over the segments in $\mathcal{T}_s$, and resolve the semantic location using $loc\text{-}resolve$ (Algorithm 9) on each segment, $c_s$. The result of $loc\text{-}resolve$ is a tuple of $\langle l_s, t_{start}, t_{end} \rangle$. The timestamps, $t_{start}$ and $t_{end}$, are from the first and last readings within the segment. The Algorithm $loc\text{-}resolve$ uses maximal likelihood estimation to select the best suited semantic location, comparing the SSIDs that are encompassed in the segment.

$$
\begin{aligned}
K = \big[&(\text{Home, Neighbor1}), (\text{Home, Neighbor2}), (\text{Neighbor1, Neighbor2}),\\
&(\text{Coffee-Shop, Gas-Station}), (\text{Coffee-Shop, Pizza}), (\text{Gas-Station, Pizza}),\\
&(\text{Office, Campus}), (\text{Office, Lab}), (\text{Campus, Lab}), (\text{Cafeteria, Lab}),\\
&(\text{Cafeteria, Campus}), (\text{Coffee-Shop, Grocery-Store}),\\
&(\text{Coffee-Shop, Dry-Cleaning}), (\text{Grocery-Store, Dry-Cleaning})\big]
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}_s = \big[&(\text{Home, Neighbor1, Neighbor2}),\\
&(\text{Coffee-Shop, Gas-Station, Pizza}),\\
&(\text{Office, Campus, Lab}),\\
&(\text{Cafeteria, Lab, Campus}),\\
&(\text{Coffee-Shop, Grocery-Store, Dry-Cleaning})\big]\\
\mathit{loc\text{-}reduce}(\mathcal{L}_s, K) = \big[&(\text{Home, Neighbor1, Neighbor2}),\\
&(\text{Coffee-Shop, Gas-Station, Pizza}),\\
&(\text{Campus, Lab}),\\
&(\text{Coffee-Shop, Grocery-Store, Dry-Cleaning})\big]
\end{aligned}
$$

Figure 3.17: Reducing semantic locations by paired-keys

Using the example segmented timeline $\mathcal{T}_s$ from 3.13, and using the semantic locations, $\mathcal{L}_s$, generated in Figure 3.18 from singleton keys, as input to *loc-timeline* (Algorithm 10) the result would be as viewed in Figure 3.19. Some outcomes of the algorithm to note, is that in some cases, consecutive segments have been merged into a single sematnic location segment. For example $c_8$ and $c_{22}$ were combined since both segments mapped to the semantic location of (Campus), and the start time from segment $c_8$ was paired with the end time of $c_{22}$, to create the time span in that location segment. The two segments that are both geographically distinct locations but share the semantics of being a Coffee-Shop, were mapped to the same semantic location.

Two visualizations can be used to observe the mobility pattern that exists in the timeline resulting from *loc-timeline*; activity and transition.

$$\begin{aligned}K = \big[&\text{(Home), (Neighbor1), (Neighbor2), (Coffee-Shop), (Gas-Station),}\\ &\text{(Pizza), (Office), (Campus), (Lab), (Cafeteria), (Grocery-Store),}\\ &\text{(Dry-Cleaning)}\,\big]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_s = \big[&(\text{Home, Neighbor1, Neighbor2}),\\ &(\text{Coffee-Shop, Gas-Station, Pizza}),\\ &(\text{Office, Campus, Lab}),\\ &(\text{Cafeteria, Lab, Campus}),\\ &(\text{Coffee-Shop, Grocery-Store, Dry-Cleaning}),\big]\end{aligned}$$

$$\begin{aligned}\textit{loc-reduce}(\mathcal{L}_s, K) = \big[&(\text{Home, Neighbor1, Neighbor2}),\\ &(\text{Coffee-Shop}),\\ &(\text{Campus}),\big]\end{aligned}$$

Figure 3.18: Reducing semantic locations by singleton-keys

Activity graphs for each semantic location is shown in Figure 3.20. These graphs display the intervals that a person was known to be at the specified location, and when they were not, over time, and help visualize patterns (if any) over time.

Transition diagrams display each semantic location as a node, and directional edges are made between the nodes that have had a transition take place from a source location to a target location. Table 3.1 displays the count of all the transitions that have occurred in the semantic timeline from Figure 3.19, with the resulting transitions visualized in the transition digram in Figure 3.21.

## 3.3.1 Evaluation

To evaluate the localizations generated by the system, three different measures were created: activity in feature space, regularity in feature space, and normalized information.

```
where: $c_s$ - a subset of the hierarchical timeline
       $\mathcal{L}_s$ - list of semantic locations
result : $key_0$ - semantic key with the largest weight
       $weight_0 - weight$

loc-resolve($c_s$, $\mathcal{L}_s$)
    $key_0 = empty$
    $weight_0 = 0$
    for $l_s$ in $\mathcal{L}_s$ do
        $weight = count(\mathcal{SSID}(c_s) \cap l_s)$
        if $weight > weight_0$ then
            $key_0 = l_s$
            $weight_0 = weight$
        end
    end
```

**Algorithm 9:** Maximal likelihood estimation to find the semantic location that represents a segment

### Feature space

The feature space for a location was defined as a stream of activity features. An activity feature is a tuple of: an integer value representing day of the week, an integer representing the hour interval, and a ping value which is either 0 or 1, representing whether or not the location was visited within that time interval; $\langle weekday$, $hour\text{-}interval$, $ping\rangle$. The $hour\text{-}interval$ is determined by what interval the timestamp takes place in if we divide the time of day by the number of hours we define each interval to represent. For the evaluation, 3 hours was used for the divisor, which provides hour intervals of 0 to 7, with 0 being the interval $[12am, 3am)$ and 7 being $[9pm, 12am)$. This feature space, encompasses all the historical data pertaining to a single location.

### Activity measure

The activity measure in the feature space, is the sum of $ping$ over the entire feature space. Since the timeline of activity is aggregated into time segments, multiple

```
where: $\mathcal{T}_s$ - hierarchical segmented timeline
       $\mathcal{L}_s$ - collection of semantic locations list pairs
result : $\mathcal{T}_l$ - timeline of semantic locations

loc-timeline($\mathcal{T}_s$, $\mathcal{L}_s$, $weight_{min}$)
    $\mathcal{T}_l$ = empty list
    $t_0$ = empty
    $t_1$ = empty
    $key_0 = empty$
    for $c_s$ in $\mathcal{T}_s$ do
        $\langle key, weight\rangle$ = loc-resolve($c_s, \mathcal{L}_s$)
        if $weight > weight_{min}$ then
            if $key_0 <> key$ and $t_0$ is not empty then
                append $\langle key_0, t_0, t_1\rangle$ to $\mathcal{T}_l$
                $key_0 = key$
                $t_0 = start(c_s)$
                $t_1 = end(c_s)$
            else
                if $t_0$ is empty then
                    $t_0 = start(c_s)$
                    $key_0 = key$
                end
                $t_1 = end(c_s)$
            end
        end
    end
```
**Algorithm 10:** Generating the timeline of semantic locations: loc-timeline

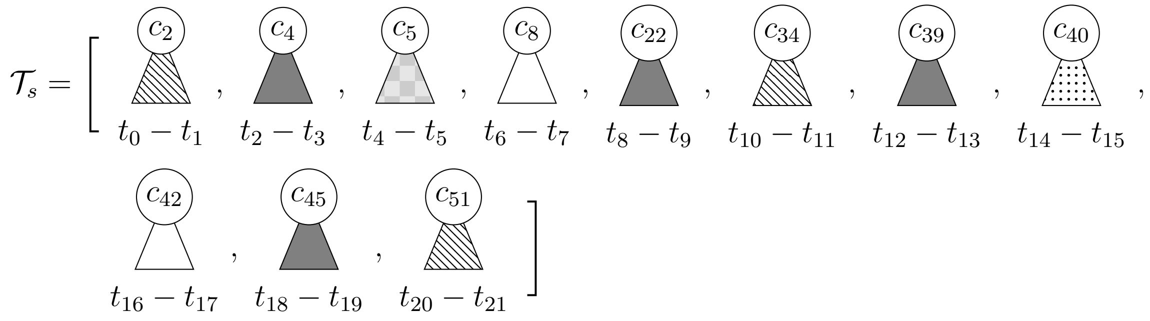
consecutive readings that take place within the same interval of time that makes up a single interval, would count as a single segment. When a location has only been visited once over an entire timeline, it's activity measure will reflect that fact by having a total close to 1.

### Regularity measure

Regularity is the predictability of the locations based upon some predictive model that is trained and tested using a data set created from the feature space.

A sliding window was used over the feature space to generate the data set. The

$$\mathcal{L}_s = [(\text{Home, Neighbor1, Neighbor2}), (\text{Coffee-Shop}), (\text{Campus}), ]$$



$\mathcal{T}_s =$
$c_2$, $t_0 - t_1$
$c_4$, $t_2 - t_3$
$c_5$, $t_4 - t_5$
$c_8$, $t_6 - t_7$
$c_{22}$, $t_8 - t_9$
$c_{34}$, $t_{10} - t_{11}$
$c_{39}$, $t_{12} - t_{13}$
$c_{40}$, $t_{14} - t_{15}$
$c_{42}$, $t_{16} - t_{17}$
$c_{45}$, $t_{18} - t_{19}$
$c_{51}$, $t_{20} - t_{21}$

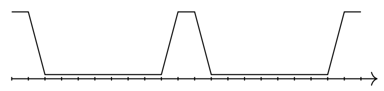
$$\begin{aligned}
\textit{loc-timeline}\,(\mathcal{T}_s, \mathcal{L}_s) = \big[\; &\langle(\text{Home, Neighbor1, Neighbor2}), t_0, t_1\rangle\,, \\
&\langle(\text{Campus}), t_2, t_3\rangle\,, \\
&\langle(\text{Coffee-Shop}), t_4, t_5\rangle\,, \\
&\langle(\text{Campus}), t_6, t_9\rangle\,, \\
&\langle(\text{Home, Neighbor1, Neighbor2}), t_{10}, t_{11}\rangle\,, \\
&\langle(\text{Campus}), t_{12}, t_{13}\rangle\,, \\
&\langle(\text{Coffee-Shop}), t_{14}, t_{15}\rangle\,, \\
&\langle(\text{Campus}), t_{16}, t_{19}\rangle\,, \\
&\langle(\text{Home, Neighbor1, Neighbor2}), t_{20}, t_{21}\rangle\,\big]
\end{aligned}$$

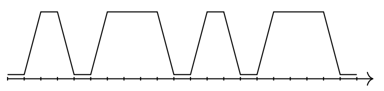
Figure 3.19: Generating a semantic timeline

input features, $x$, were $n$ consecutive activity features, and the output, $y$, was the $ping$ value within the next consecutive activity feature.

Figure 3.22 illustrates a timeline of activity features, the feature space, and the resulting data set, using $n = 5$ consecutive activity features for the input. Multiple values of $n$ were experimented with, but $n = 5$ resulted in the best predictive model. Too small of a $n$ resulted in an underfitting problem, and too high of a $n$ resulted in too much bias.

The individual activity features, $\langle weekday, hour\text{-}interval, ping\rangle$, in $x$, and the $ping$ value in $y$, get converted to binary representation using dummy variables.


(a) (Home, Neighbor1, Neighbor2)


(b) (Campus)

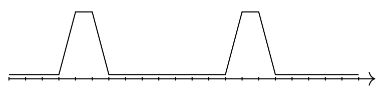

(c) (Coffee-Shop)

Figure 3.20: Activity of semantic location

| | | Target | | |
|---|---|---|---|---|
| | | Home | Campus | Coffee |
| Source | Home | 0 | 2 | 0 |
| | Campus | 2 | 0 | 2 |
| | Coffee | 0 | 2 | 0 |

Table 3.1: Transition count matrix

With the training set being made up of 80% of the data generated over the timeline, the remaining 20% is used as the test data.

A neural network [5] was selected as the modeling tool. The advantage is that neural networks have mature learning algorithms to generate the specific model parameters based on the training data, which is the historic data, and is excellent at capturing patterns.

A shallow multi-layer perceptron (MLP) architecture is used to fit a best model using the feature space as illustrated in Figure 3.23. TensorFlow [1] with Keras [4] was used to create a sequential model, as shown in Listing 3.1. When creating the model, the softmax function was used to normalize the data at activation. Since the output is in the form of a multi-class classification the loss function is set to *categorical cross entropy*, with a stochastic gradient descent optimizer. Accuracy from the the test case was used, and saved, as the regularity measure for the location. Training error from attempting to fit a model also contributes to the regularity value as a 0.

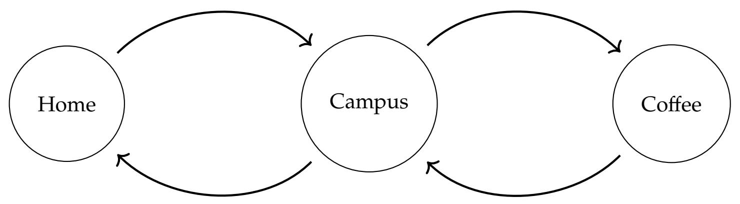
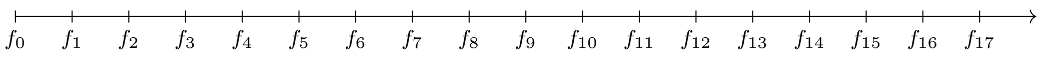


Figure 3.21: Transition diagram


$f_0$ $f_1$ $f_2$ $f_3$ $f_4$ $f_5$ $f_6$ $f_7$ $f_8$ $f_9$ $f_{10}$ $f_{11}$ $f_{12}$ $f_{13}$ $f_{14}$ $f_{15}$ $f_{16}$ $f_{17}$

$$x = \left[\,(f_0, f_1, f_2, f_3, f_4)\,, (f_1, f_2, f_3, f_4, f_5)\,, (f_2, f_3, f_4, f_5, f_6) \ldots (f_{i-5}, f_{i-4}, f_{i-3}, f_{i-2}, f_{i-1})\,\right]$$

$$y = [ping(f_5), ping(f_6), ping(f_7)...ping(f_i)]$$

Figure 3.22: Training and test set creation

```
model = Sequential()
model.add(Dense(input_dim=x_size, output_dim=y_size))
model.add(Activation("softmax"))
model.compile( loss='categorical_crossentropy',
               optimizer='sgd',
               metrics=['accuracy'])
model.fit(train_x, train_y, verbose=0)
```

Listing 3.1: Tensor Flow with Keras implementation

Locations that have been visited too few of times in the timeline will not have enough data to successfully train a model, and thus, would not have a regularity value. Inversely, a location that has been visited for a significant amount of time, and therefore has a high enough activity value, but was never visited again, will show a significantly high regularity score close to 1.

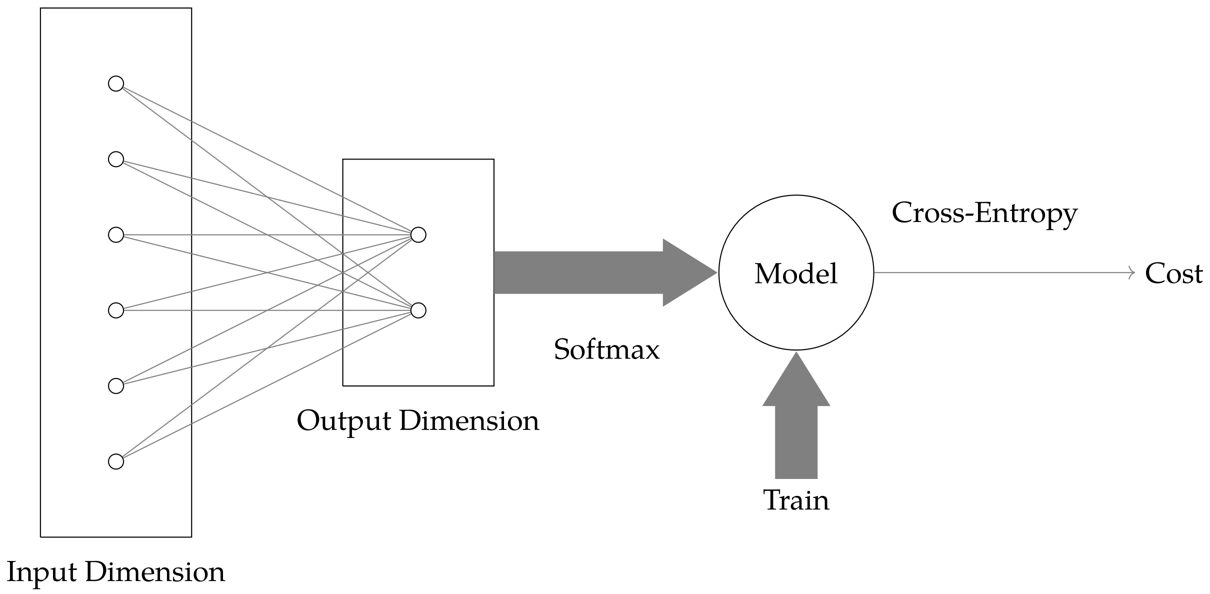


Figure 3.23: Sequential Neural Network

When comparing the regularity measure against the other measures for determining the quality of a location within a localization, on real data, the regularity measure was the most effective. The results of applying all the measures on the collected data, is described in Chapter 5.

**Normalized information**

The normalized information measure, is a value of the information content in the timeline assuming a weekly schedule. The sum of pings over every interval for every day of the week was taken along the timeline feature space, and normalize it against the sum of pings over the whole timeline. This creates a probability distribution over a week long period, segmented by the intervals. Algorithm 11 provided a normalized entropy value, given the probability distribution. The normalized entropy value is saved as the normalized information measure.

Resulting normalized information values close to 1, represent a location that has been seen very few times in the activity timeline, and so these locations don't hold

```
where: $x$ - probability distribution
       $n$ - is the number of segments in the probability distribution
result : A normalized entropy
```

$$\frac{\sum_{i=0}^{n-1} \log_2 x_i}{\log_2 n}$$

**Algorithm 11:** Normalized entropy of probability distribution

any significance to the mobility pattern. Inversely, a normalized information value close to 0, represents a location that is almost always being seen. An example of a location with 0 normalized information, would be the root cluster of the hierarchy, since it represents all places seen, since an individual always exists at a location, this would have no valuable information.

# Chapter 4

# System Implementation

The implementation includes a mobile application to collect data, and a collection of Python scripts that implements the algorithms described in Chapter 3, with some improvements made to increase performance.

## 4.1 An app for mobile data collection

A portable means of data collection was required, that would seamlessly integrate into a person's life without disruption, so that the person would be encourage to keep the data collecting device with them at all times, to ensure the data collected is as representative of the person's mobility as possible. The device chosen for data collection was the smart phone, as people carry these devices for the most part throughout their day. An Android application was built for easy access to the WiFi scanning capabilities.

Development of Android applications can be achieved by many tools, but to access the Wifi scanning capabilities of Android devices, the easiest method to achieve this project's goals was through creating the application natively with Java.

The prototype of the Android application written in Java achieved the data collecting objective, but with room for improvement. The second and final implementation of the Android application was with Kotlin [9] and the ANKO library [25], which resulted in a stable application that better suited our data collecting objective.

Both versions of the application utilized an internal SQLite database for storage of the WiFi scans.

Andoid applications are made up of ”activities”. In a model-view-controller architecture the activity would be the controller and would call upon a user interface view to display the model. The main activity in the data collection has a single view as seen in figure 4.1, and contains a toggle switch for turning on/off the timed scanning service, as well as displaying basic statistics of the current collection of scans.

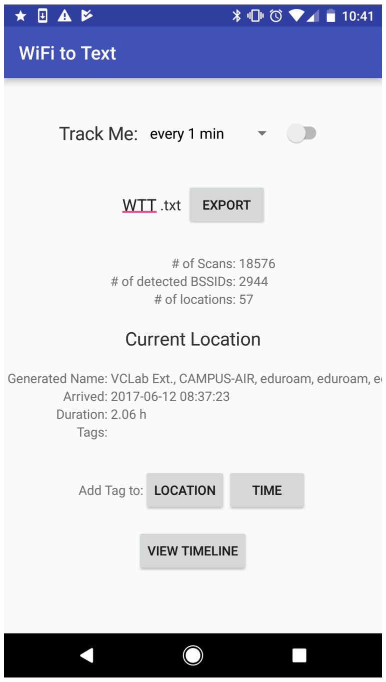
Services in Android development are used for long running or on-going processes. The timed scanning service that gets invoked by the main activity is an on-going process that triggers the WiFi scan service at an interval set in the main activity UI. Since the WiFi scanning process is lengthy, it was implemented as a long running service.

The WifiManager library used to scan Wifi signals, in the WiFi scanning service, returns an array of BSSID, SSID, signal strength, timestamp . The BSSID is the MAC address of the device. The SSID is the user generated name that has been assigned to the device. The signal strength is in a decibel format.

It is important to mention that the scans that the Android WifiManager are executing are *passive*, and that they do not risk a person’s privacy. *Passive* scans simply means that the device is listening for beacons emitted from WiFi hotspots, and does not actually emit its own signal or information.

These scans are stored in an SQLite version of their original representation, in a table, OBSERVATION, within the local SQLite database on the device. The OBSERVATION table has 5 columns: *id*, *timestamp*, *ssid*, *bssid*, *strength*. The *id* is used as a primary key to ensure uniqueness of all records. Once the current scan of WiFi signals are stored in the database, that invocation of the WiFi scanning service is concluded.

This raw data is extracted from the local database as JSON, which is transferred to a computer to be converted into an SQLite database that replicates the structure that was implemented in the mobile application, and then fed into the pipeline for off-line analysis.



Track Me: every 1 min

WTT .txt

# of Scans: 18576
# of detected BSSIDs: 2944
# of locations: 57

Current Location

Generated Name: VCLab Ext., CAMPUS-AIR, eduroam, eduroam, e
Arrived: 2017-06-12 08:37:23
Duration: 2.06 h
Tags:

Figure 4.1: WiFi Scanner Mobile Application

## 4.2 A pipeline for mobility pattern analysis

The Python scripts form a pipeline [15] consisting of: reading generator, cluster generator, hierarchical timeline building, movement segmentation, physical location identification, and then semantic location grouping.

### 4.2.1 Hierarchical clustering

The reading generator retrieves the records from the database, and streams reading objects to the cluster generator. The cluster generator converts the readings into cluster objects that then streams the clusters to the hierarchical timeline, using the Python equivalent of $ht\text{-}append$ (Algorithm 2).

The time it took to append each reading to the hierarchical timeline was recorded, to analyze the performance of $ht\text{-}append$. Figure 4.2 displays the time it took to append each reading (in milliseconds), over the number of readings that existed in the hierarchy at the time of appending, on real data collected from the mobile application. Even though the average case complexity of $ht\text{-}append$ is $\mathcal{O}(log(r))$, when used on real data, we see that the amount of readings already in the hierarchy does not increase the time it takes to append a new reading. It is the nature of $ht\text{-}append$ on real data that makes it appealing for on-line clustering.

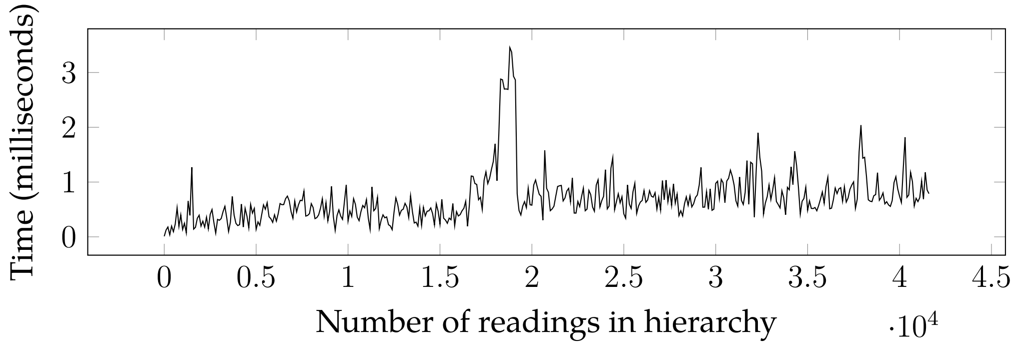


Figure 4.2: Performance of hierarchical append function

### 4.2.2 Segmentation

The hierarchical timeline is segmented in the Python equivalent of $ht\text{-}segment$ (Algorithm 3). The complexity of $ht\text{-}segment$ is hindered by the execution and complexity of the $sim$ measure (Definition 1). The performance of $ht\text{-}segment$ was improved in two ways: sampling and caching. The usage of $sim$ in $ht\text{-}segment$, was replaced with $min\text{-}sim$ (Algorithm 12), which implements both the sampling and caching of a minimum similarity.

```
where: $c$ - cluster
       $n$ - number of readings in a sample
result : minimum similarity
$min\text{-}sim(c, n)$
    if $c$ has cached minimum similarity then
        return cached value
    else
        if $c$ is a leaf cluster then
            $m = 1$
        else
            $\nu_{left} = min\text{-}sim(left(c))$
            $\nu_{right} = min\text{-}sim(right(c))$
            if $\nu_{left} = 0$ or $\nu_{right} = 0$ then
                $m = 0$
            else
                $\omega_{left} = sample\text{-}readings(left(c), n)$
                $\omega_{right} = sample\text{-}readings(right(c), n)$
                $M = list(\nu_{left}, \nu_{right})$
                for $j$ in $\omega_{left}$ do
                    for $k$ in $\omega_{right}$ do
                        append $sim(j, k)$ to $M$
                    end
                end
                $m = min(M)$
            end
        end
    end
    $cache(c, m)$
    return $m$
```

**Algorithm 12:** Cached minimum similarity

A list of similarity values (using the original $sim$ function) was created, comparing the readings from the left child to the readings in the right child, and then select the minimum value of these to be the minimum similarity value for the cluster. The minimum similarity is selected to ensure that clusters at a high level, which include the same location multiple times in between other locations, are split into smaller segments.

Caching the minimum similarity value in the cluster, allows us to avoid duplicate computations from being executed.

Before calculating the minimum similarity for a cluster, the cached values for both children is checked, and if either has a minimum similarity of 0, then 0 is used as the minimum similarity for the current cluster, which also improves performance of the segmentation.

### 4.2.3 Localization

The Python implementation of $loc\text{-}physical$ which generates the collection of physical locations, $\mathcal{L}_p$, is just as described in Algorithm 4. The Python implementation of $loc\text{-}semantic$ (Algorithm 6) and $loc\text{-}merge$ has been modified so that the semantic locations contain the physical locations that they were generated from, rather than just a set of SSID values. This allows for evaluation of the generated locations to be completed in a more efficient manner.

During implementation of $loc\text{-}merge$, it was discovered that using more than two SSID values as combinations in the collection of possible keys to merge locations on, resulted in an insignificant amount of improvement to the localization than the physical localization of $\mathcal{L}_p$.

# Chapter 5

# Case Study

## 5.1 Description of the case study

The case study utilizes the data collected, on Adele Hedrick's Android phone, using the mobile data collection application. The data includes a couple weeks in the Fall semester of 2016, an excursion out of the country to a conference in December, and a couple weeks in the Winter semester of 2017, and there is a cap in the data over the Winter holidays.

Adele would describe her frequent behavior as: going to the campus most weekdays, few weekdays working from home, picking up her children from their own school before and after going to the campus, teaching assistant responsibilities throughout the week, with weekends usually spent at home with a few excursions.

The data set includes 41,619 readings, with 14,758 unique BSSIDs, and 4,619 unique SSIDs discovered. This difference in unique BSSIDs observed versus unique SSIDs observed displayed in figure 5.1, indicates that many hotspots share the same SSID value.

The top 20 SSIDs with the highest count of BBSIDs are displayed in table 5.1.

|  | Total # observed |
|---|---|
| BSSID | 14758 |
| SSID | 4619 |

Table 5.1: Total number of BSSID vs SSID observed in case data set

The SSID with the highest count of BSSIDs is an empty string, or an unset SSID. This is an example of noise in the dataset which we hope would be filtered out through the thresholds in the algorithm. The rest of the SSIDs listed, represent areas that would require more than one hotspot to provide Wi-Fi coverage to the whole area. The SSIDs; ”YYZ Corp” and ”Toronto Pearson Wi-Fi”, are both SSIDs that represent Toronto Pearson Airport. The SSID; ”McCarran WiFi”, belongs to the McCarran Airport in Las Vegas, and the SSID, ”MonteCarloWiFi”, belongs to the hotel that Adele stayed at in Las Vegas. Another significant SSID is ”CAMPUS-AIR”, which belongs to the hotspots at University of Ontario Institute of Technology (UOIT). All of these mentioned SSIDs, do in fact represent large physical areas that require multiple hotspots to provide Wi-Fi coverage to.

The top 20 SSIDs that have the highest observation count are displayed in table 5.2. As can be expected, the SSIDs; ”CAMPUS-AIR” and ”MonteCarloWiFi” make the top 20 in the list. The other significant set of SSIDs listed are the ones with the prefix, ”SSK”, which belong to Adele’s hotspots at her home. The other SSIDs listed belong to neighbours and hotspots near the lab she works from at UOIT’s campus, as well as her home.

## 5.2 Discussion and visualization

Three levels of localization were observed and evaluated: L0, L1 and L2. L0 is the physical localization through grouping of BSSIDs using $loc\text{-}physical$. The L1 localization, localizes L0 semantically, with $loc\text{-}semantic$ and $loc\text{-}reduce$, using pairs of

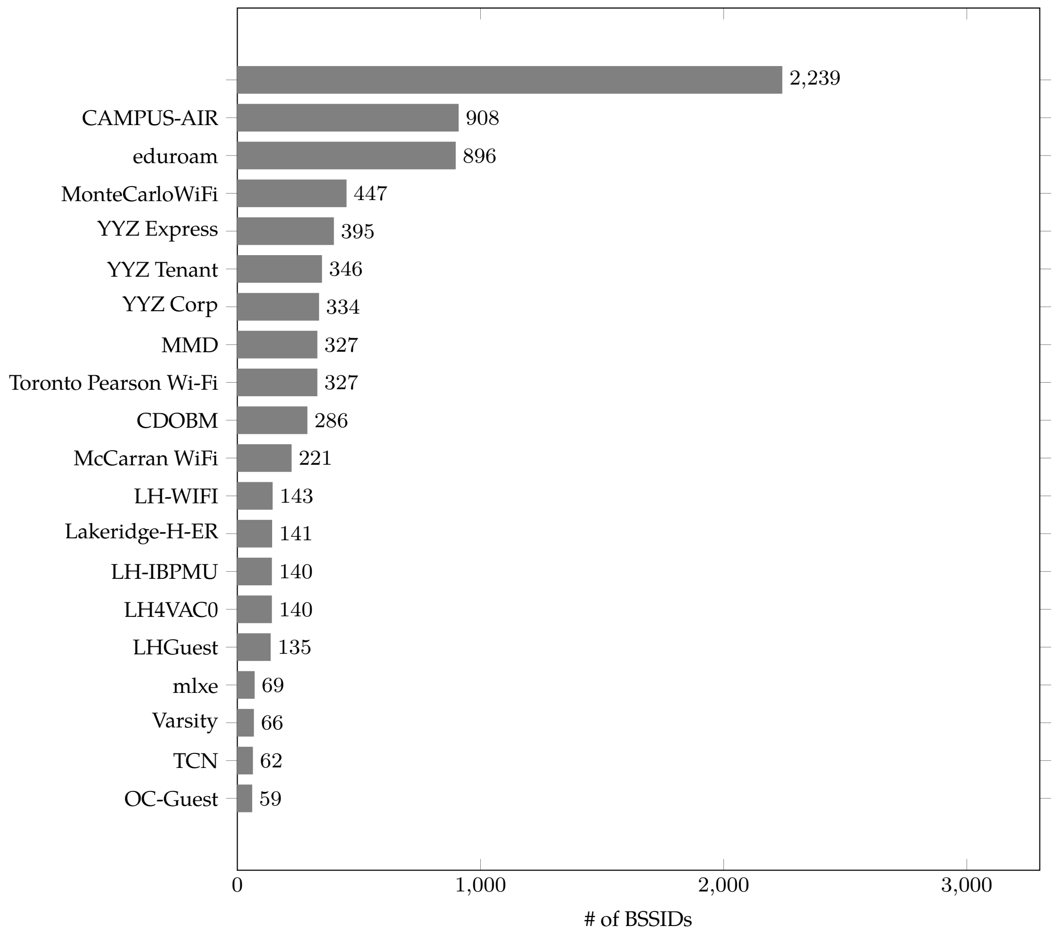


Figure 5.1: BSSID count for top 20 SSIDs

SSIDs as keys. The L2 localization also localizes L0 semantically, with *loc-semantic* and *loc-reduce*, but with single SSIDs as keys (L2).

The number of generated locations in each level of localization is compared in figure 5.3. Recalling from figure 5.1, the total unique BSSID count of 14,758, was reduced to 138 generated locations in L0 using *loc-physical*. The 138 physical locations were then reduced to 70 locations in L1, and 32 locations in L2.

Table 5.2 lists the semantic locations that were generated by the system at the L2 localization using single SSID values as keys. The SSIDs in this table have been concatenated together, separated by "\". The first 18 locations in the list are the

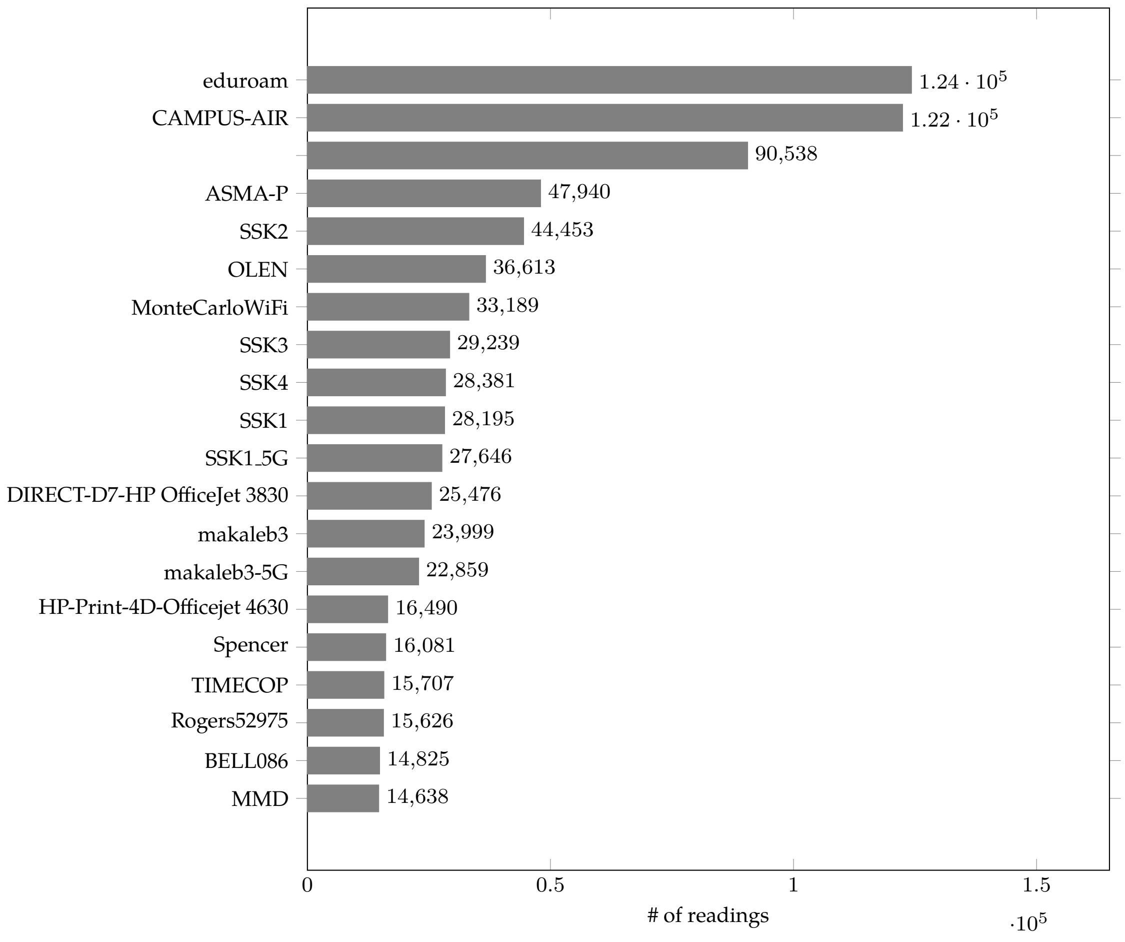


Figure 5.2: Reading count for top 20 SSIDs

result of a merge as they only show a single SSID being used as the name. Of the 18 locations, we see the large geographical areas have been merged; "YYZ TENANT", "YYZ EXPRESS", "MONTECARLOWIFI" and "CAMPUS-AIR". The home location "SSK2" has been grouped into a single location, and the neighbors SSIDs have been grouped in with the more significant SSIDs that they were observed with.

Activity graphs were defined as being the timeline of a single location. The high points indicate that the person was at that location at that time, and the low points indicate that the person was not at that location at that time. Figure 5.4:a, shows the activity of location "SSK2" which represents Adele's home, figure 5.4:b, shows

| ID | Name |
|---|---|
| 0 | SUNDANCE |
| 1 | ASUS |
| 2 | YYZ TENANT |
| 3 | SCA2 |
| 4 | BEYONDGUESTWIFI |
| 5 | ATTWIFI |
| 6 | BELL709 |
| 7 | WMDEMO |
| 8 | GUEST WIRELESS |
| 9 | YYZ EXPRESS |
| 10 | TCONNECT |
| 11 | MLXE |
| 12 | ROGERS16355 |
| 13 | VARSITY |
| 14 | MCCARRAN WIFI |
| 15 | SSK2 |
| 16 | MONTECARLOWIFI |
| 17 | CAMPUS-AIR |
| 18 | BELL117/BMOBILE/DIRECT-FB-HP OFFICEJET PRO 8710 |
| 19 | OC-GUEST/OSC/TCN |
| 20 | J47CIX6F/M4M786IA/P39TXD7W |
| 21 | BELL192/BELL224/ROGERS55298 |
| 22 | ASHLEY FURNITURE HOMESTORE/ASHLEY GUEST WIFI/JYSKHOF |
| 23 | EMERALDSHARK/OSHAWAOFFICE/WIFI123 |
| 24 | RMTAP/TAP GUEST/THE TAP |
| 25 | ATT-WIFI/ATTWIFI - PASSPOINT/STEEL |
| 26 | BELL402/POP LED SIGN/STARBUCKS WIFI |
| 27 | TRG 3PD/TRG BO/TRG INV |
| 28 | BLACKSTARGROUP/GEICO GAMING/PRODUCTION |
| 29 | CVWIFI/CWIFI/MINT |
| 30 | LH-IBPMU/LH-WIFI/LH4VAC0 |
| 31 | OUTBOXSCANNERS/T-MOBILE ARENA/TASSTMOBILE |

Table 5.2: L2 Semantics

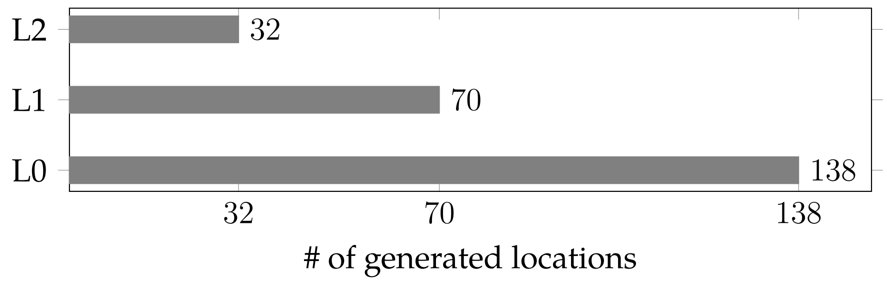
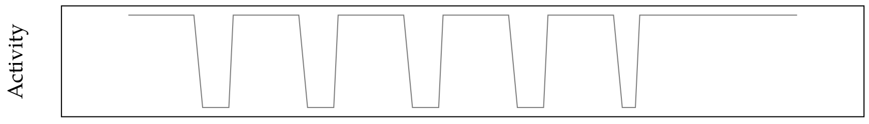


Figure 5.3: Total number of generated locations in localizations

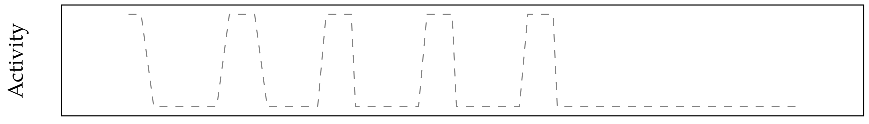
the activity of location ”CAMPUS-AIR”. We can see that for the most part, these two graphs are inverses of one another, other than the time Adele left her home on Saturday, and didn’t travel to the campus when she left. This inverse behavior is more evident when we put both plots on the same graph as depicted in figure 5.4:c. The activity graphs display the mobility pattern of a person for a single location, and combined together we can achieve more insights.

The timeline of all locations, allows for the tabulation of how many transitions from one location to another have taken place, and transition diagrams can be made with this information. Transition diagrams consist of nodes that represent locations, and directed edges that represent that at least one transition has taken place from one location to another. The size of the nodes indicate the amount of readings that have taken place at that location. Figure 5.5 compares the transition diagrams from the three levels of localization: L0, L1 and L2. There is a lot of information in L0 and L1, but it is L2 that provides the most appropriate level of abstraction to gain insight into a person’s mobility pattern.

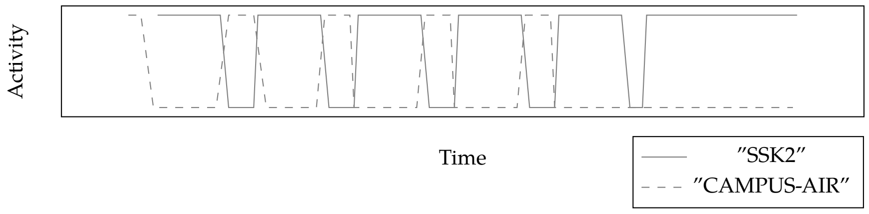
A closer look at the L2 transition diagram in figure 5.6, with IDs that correspond to table 5.2, we can get a high level view of Adele’s movement over her timeline. Her home, ”SSK2”, which is ID 15 in the diagram, is the location she has spent the most time at. The campus, ”CAMPUS-AIR” is 17, and we can see there have been



(a) SSK2 (ID 15)



(b) Campus-Air (ID 17)



(c) SSK2 (ID 15) and Campus-Air (ID 17)

Figure 5.4: L2 Week of Activity

transitions from her home to the campus, as well as indirect routes from home to campus and vice versa.

Adele's trip to Las Vegas is also displayed in the diagram. ID 9 and 2, are both at Pearson Airport in Mississauga, and it can be observed that she traveled home from the airport directly, but took a different route on her way to the airport. ID 14 is the McCarran Airport in Las Vegas, and ID 16 is the hotel she stayed at, which was hosting the conference. Adele explored some of the surrounding sights, but the hotel is the most significant node in the branch of her trip.

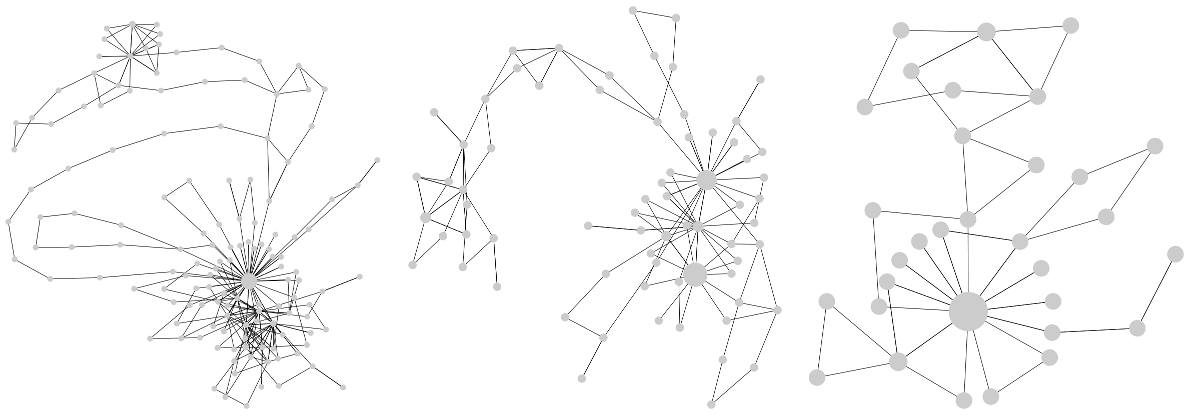

(a) L0 (b) L1 (c) L2

Figure 5.5: Transition diagrams with different levels of localization

## 5.3 Evaluation of localizations

Three evaluation measures; activity, normalized information and regularity, to each location in each localization. Table 5.3 displays the result of the measures in each location within the L2 localization, and the figures; 5.7 shows the locations with an activity value $> 1$, 5.8 shows the locations with a normalized information content $< 1$, and 5.9 shows the locations with a regularity value $> 0$.

The results in able 5.3 and the results of the other localizations is summarized in table 5.4. Figures 5.10, 5.11 and 5.12, provide a visual representation of the tabled data. With the three measures, L0 (the localization using only BSSID values) results with many locations, with the smallest ratio of significant locations, and therefore a mobility pattern is not decipherable. The L1 localization using pairs of SSID values as keys, significantly reduces the amount of locations, and improves the ratio of significant locations to insignificant ones. The last localization, L2, made from singletons of SSID values as keys, has the best performance by providing the most reduced collection of locations, which has the best ratio of significant to insignificant locations.

| ID | Total Time | Information | Activity | Regularity |
|---|---|---|---|---|
| 0 | 1.13 h | 0.8278045662 | 2 | — |
| 1 | 54.43 m | 1 | 1 | — |
| 2 | 3.12 m | 1 | 1 | — |
| 3 | 20.52 m | 1 | 1 | — |
| 4 | 2.53 m | 1 | 1 | — |
| 5 | 1.09 h | 0.8278045662 | 2 | 0.75 |
| 6 | 1.34 h | 0.8278045662 | 2 | — |
| 7 | 1.67 h | 0.8278045662 | 2 | — |
| 8 | 54.85 m | 1 | 1 | — |
| 9 | 3.63 h | 0.7270766946 | 3 | 0.8571428571 |
| 10 | 3.90 m | 1 | 1 | — |
| 11 | 28.58 m | 1 | 1 | — |
| 12 | 23.57 m | 1 | 1 | — |
| 13 | 6.83 m | 0.669052758 | 5 | 0.8823529412 |
| 14 | 9.24 h | 0.6556091324 | 4 | 0.6666666667 |
| 15 | 883.21 h | 0.0110419581 | 312 | 0.9230769231 |
| 16 | 82.63 h | 0.1550558453 | 30 | 1 |
| 17 | 78.30 h | 0.3223341936 | 38 | 0.8536585366 |
| 18 | 1.02 m | 1 | 1 | — |
| 19 | 12.18 m | 1 | 1 | — |
| 20 | 9.05 m | 1 | 1 | — |
| 21 | 2.07 m | 1 | 1 | — |
| 22 | 6.72 m | 1 | 1 | — |
| 23 | 3.83 m | 1 | 1 | — |
| 24 | 1.72 h | 1 | 1 | — |
| 25 | 13.22 m | 0.8278045662 | 2 | — |
| 26 | 2.06 h | 1 | 1 | — |
| 27 | 1.09 h | 1 | 1 | — |
| 28 | 2.05 m | 1 | 1 | — |
| 29 | 23.45 m | 1 | 1 | — |
| 30 | 1.66 h | 0.8278045662 | 2 | — |
| 31 | 1.03 m | 1 | 1 | — |

Table 5.3: L2 Evaluation

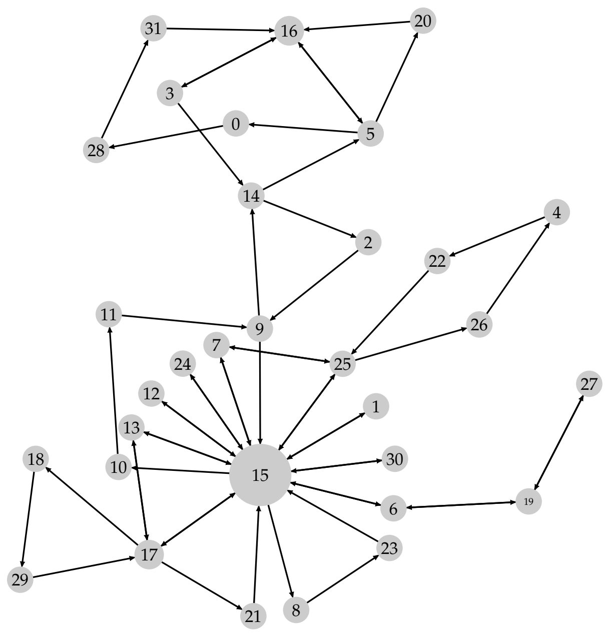


Figure 5.6: Transition diagram of L2

| | Total | Activity | | Regularity | | Information | |
|---|---|---|---|---|---|---|---|
| | | $> 1$ | Ratio | $> 0$ | Ratio | $< 1$ | Ratio |
| $\mathcal{L}_0$ | 138 | 31 | 0.2246376812 | 26 | 0.1884057971 | 30 | 0.2173913043 |
| $\mathcal{L}_1$ | 70 | 19 | 0.2714285714 | 15 | 0.2142857143 | 19 | 0.2714285714 |
| $\mathcal{L}_2$ | 32 | 12 | 0.375 | 7 | 0.21875 | 12 | 0.375 |

Table 5.4: Localization comparisons

The locations that have been isolated as relevant, within each measure, for each localization, have been visualized in the transition diagrams in figures 5.13, 5.14 and 5.15. The result is that the selected filters for each measure, results in subsets of locations that have a significant intersection. The normalized information and regularity filtered locations are generally a subset of the locations filtered based on activity, and this is the case for every location with the L2 localization, as made evident in table 5.3.

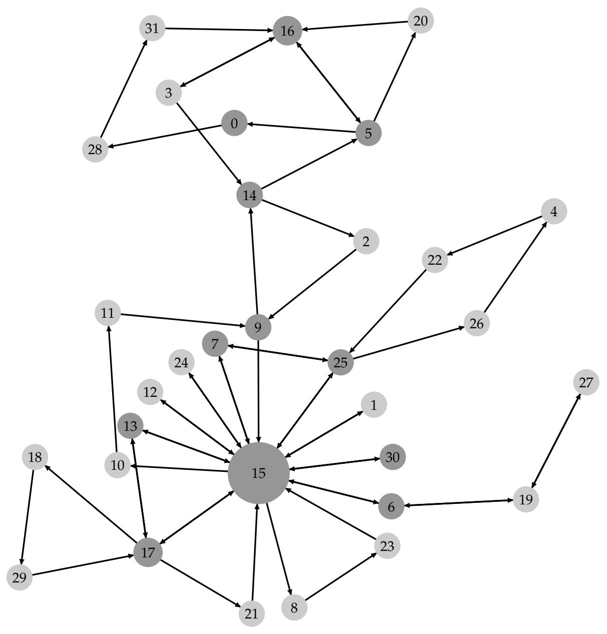


Figure 5.7: Transition diagram of L2 marking activity $> 1$

To analyze the relationship between the measures of locations in a localization, and to discover any correlation between the locations identified as good, versus locations identified as bad, a series of 2D plots and a 3D plot for each localization were created.

For the L0 localization, figures 5.16 and 5.17 display the relationship, if any, between the measures; for the L1 localization, figures 5.18 and 5.19, display the relationships, if any; and the L2 localization, figures 5.20 and 5.21, display the relationships, if any.

The Activity vs. Information plots 5.16 (a), 5.18 (a) and 5.20 (a), suggest that there is a inversely linear correlation between the normalized information content and the activity measures. The Activity vs. Regularity plots: 5.16 (b), 5.18 (b) and 5.20 (b), and the Regularity vs. Information plots: 5.16 (c), 5.18 (c) and 5.20 (c), suggest that the locations labeled as meaningful, exist in a large cluster, while the outliers resulted in being the locations that were labeled as not meaningful by the filters. The measure that demonstrated the most effectiveness at identifying loca-

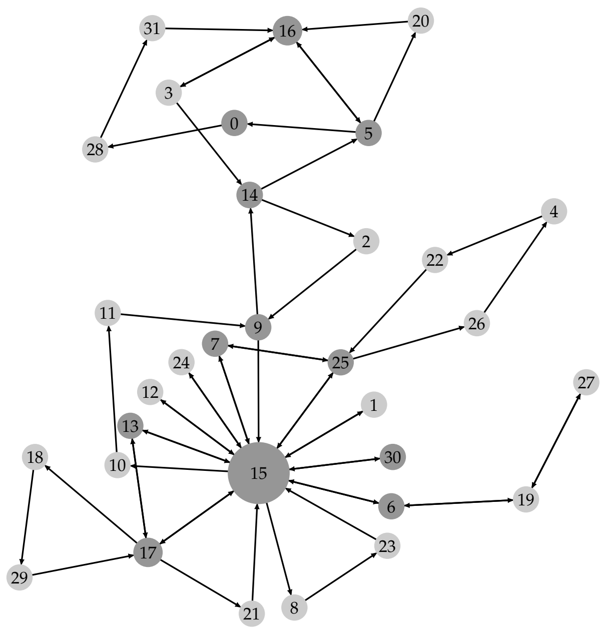


Figure 5.8: Transition diagram of L2 marking information $< 1$

tions as good versus bad in the localizations, was the Regularity measure. The Regularity measure, alone, could determine whether or not a location is good or bad.

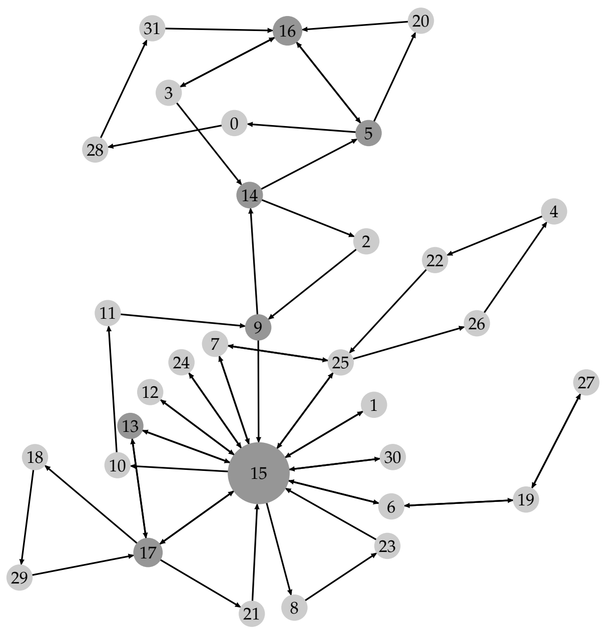
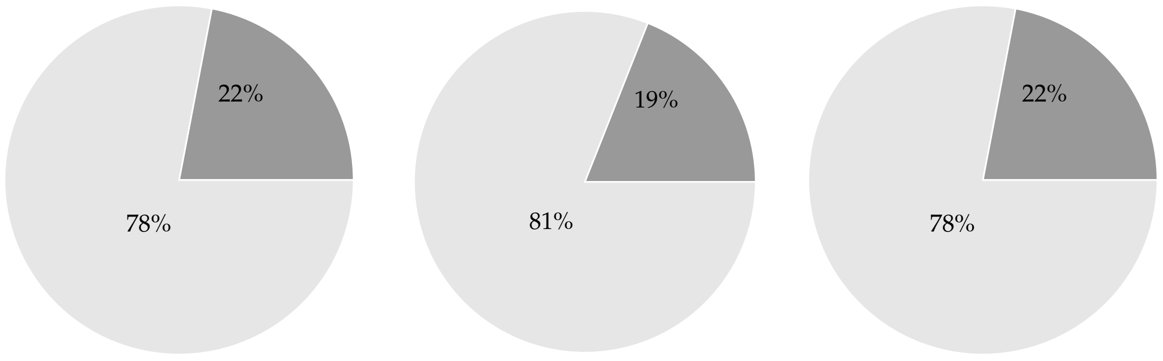
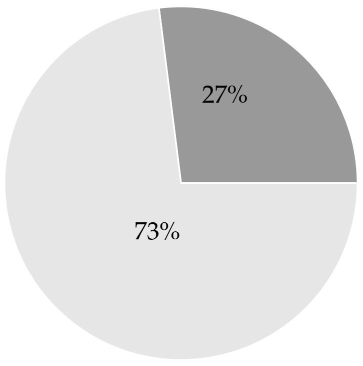


Figure 5.9: Transition diagram of L2 marking regularity $> 0$

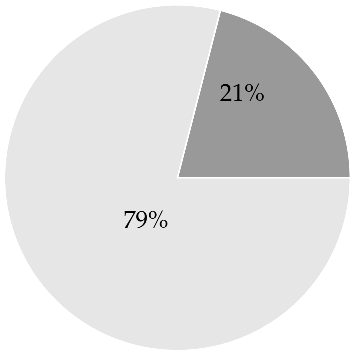


(a) Activity $> 1$ (b) Regularity $> 0$ (c) Information $< 1$

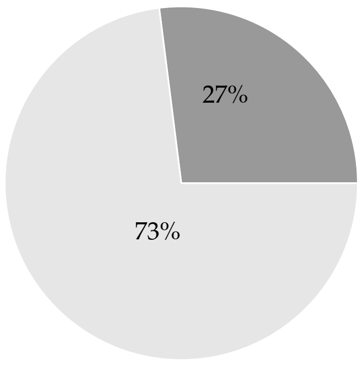
Figure 5.10: L0 evaluation comparison



(a) Activity > 1

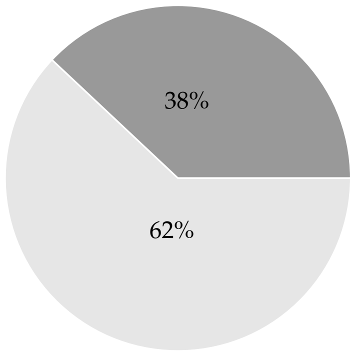


(b) Regularity > 0

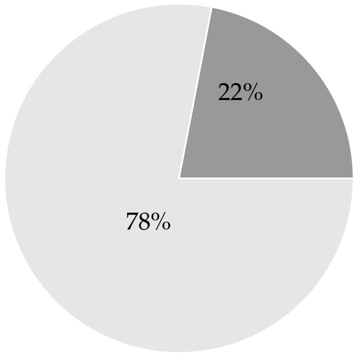


(c) Information < 1

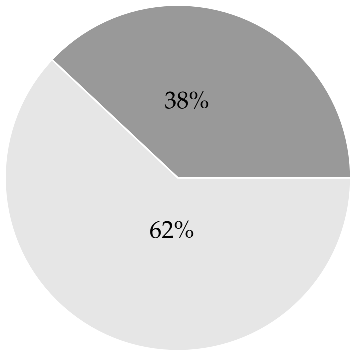
Figure 5.11: L1 evaluation comparison



(a) Activity > 1

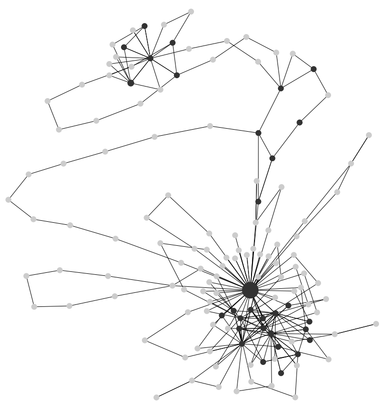


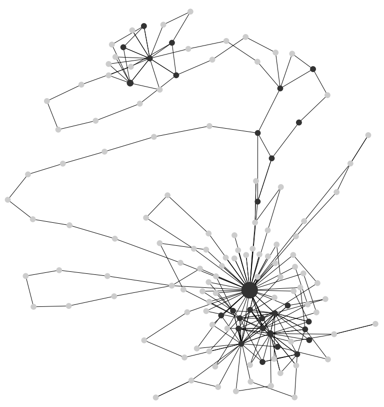
(b) Regularity > 0



(c) Information < 1

Figure 5.12: L2 evaluation comparison


(a) Activity > 1 marked


(b) Information < 1 marked

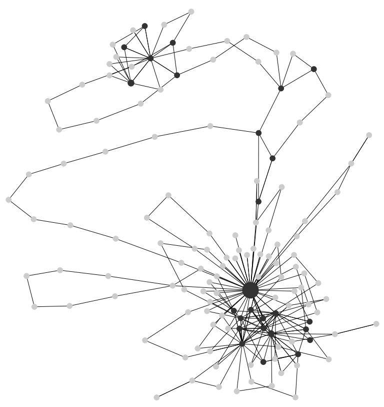
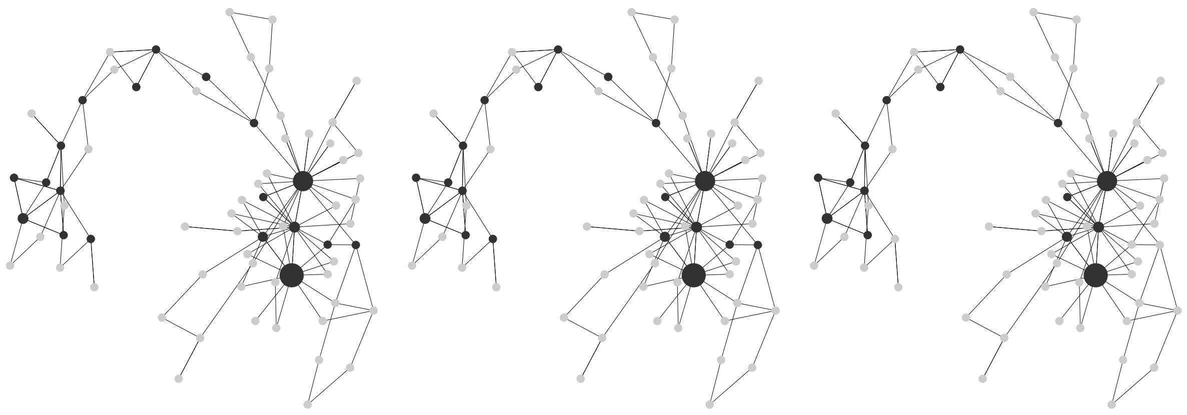
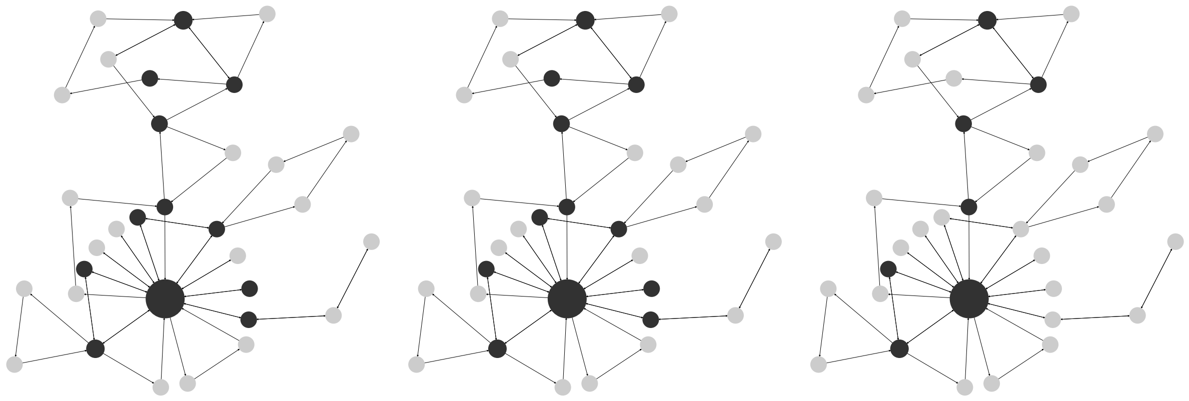

(c) Regularity > 0 marked

Figure 5.13: L0 transition diagram



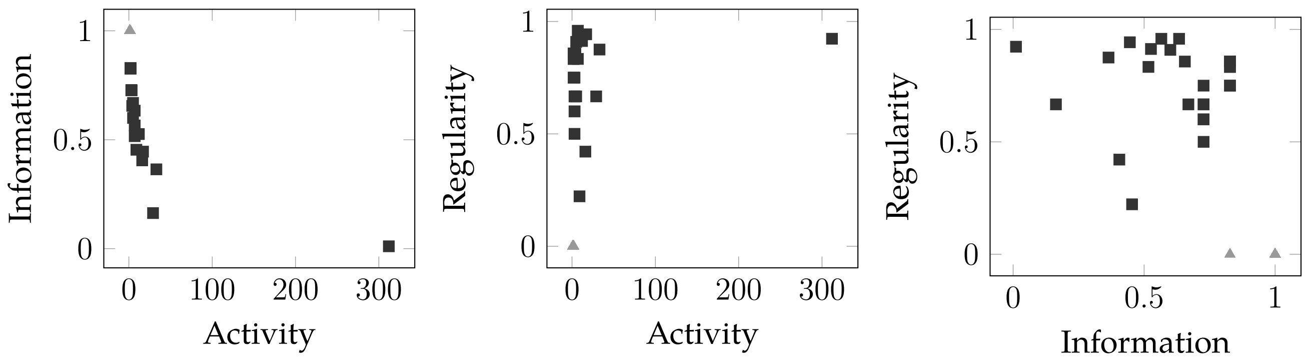
(a) Activity > 1 marked (b) Information < 1 marked (c) Regularity > 0 marked

Figure 5.14: L1 transition diagram



(a) Activity > 1 marked (b) Information < 1 marked (c) Regularity > 0 marked

Figure 5.15: L2 transition diagram



(a) L0 Activity vs. Information (b) L0 Activity vs. Regularity (c) L0 Information vs. Regularity

Figure 5.16: L0 2D evaluation comparison

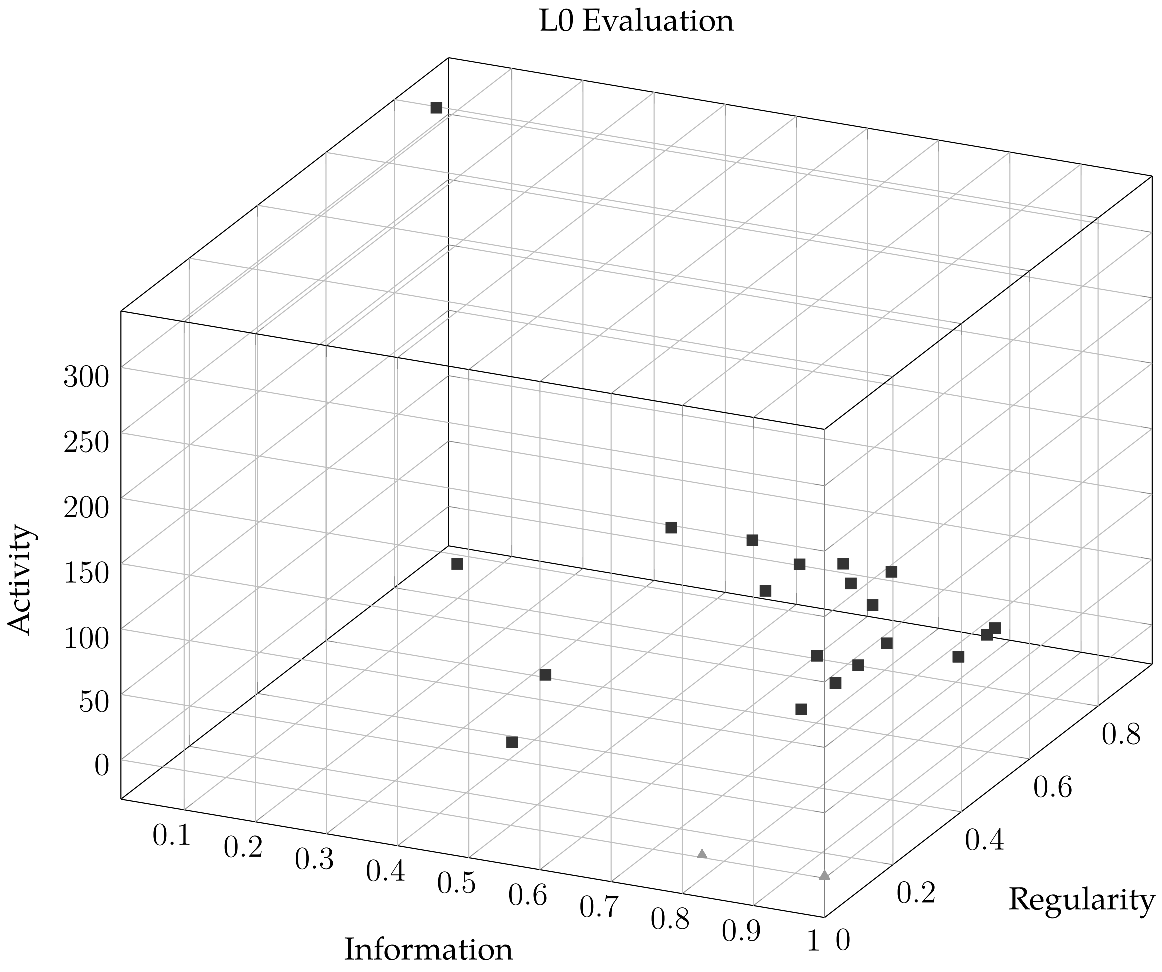
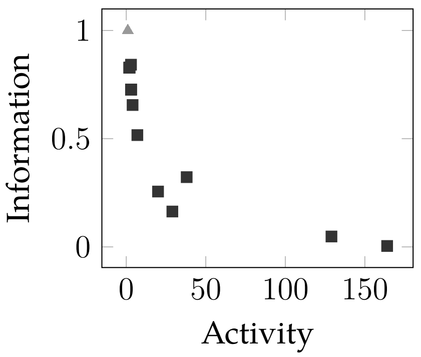
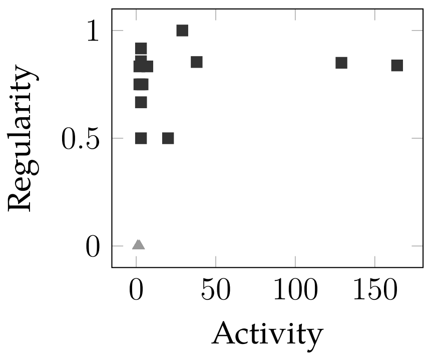


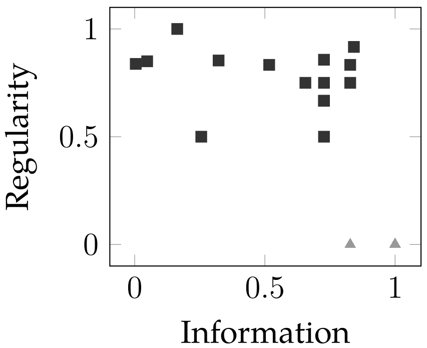
Figure 5.17: L0 3D evaluation comparison



(a) L1 Activity vs. Information



(b) L1 Activity vs. Regularity



(c) L1 Information vs. Regularity

Figure 5.18: L1 2D evaluation comparison

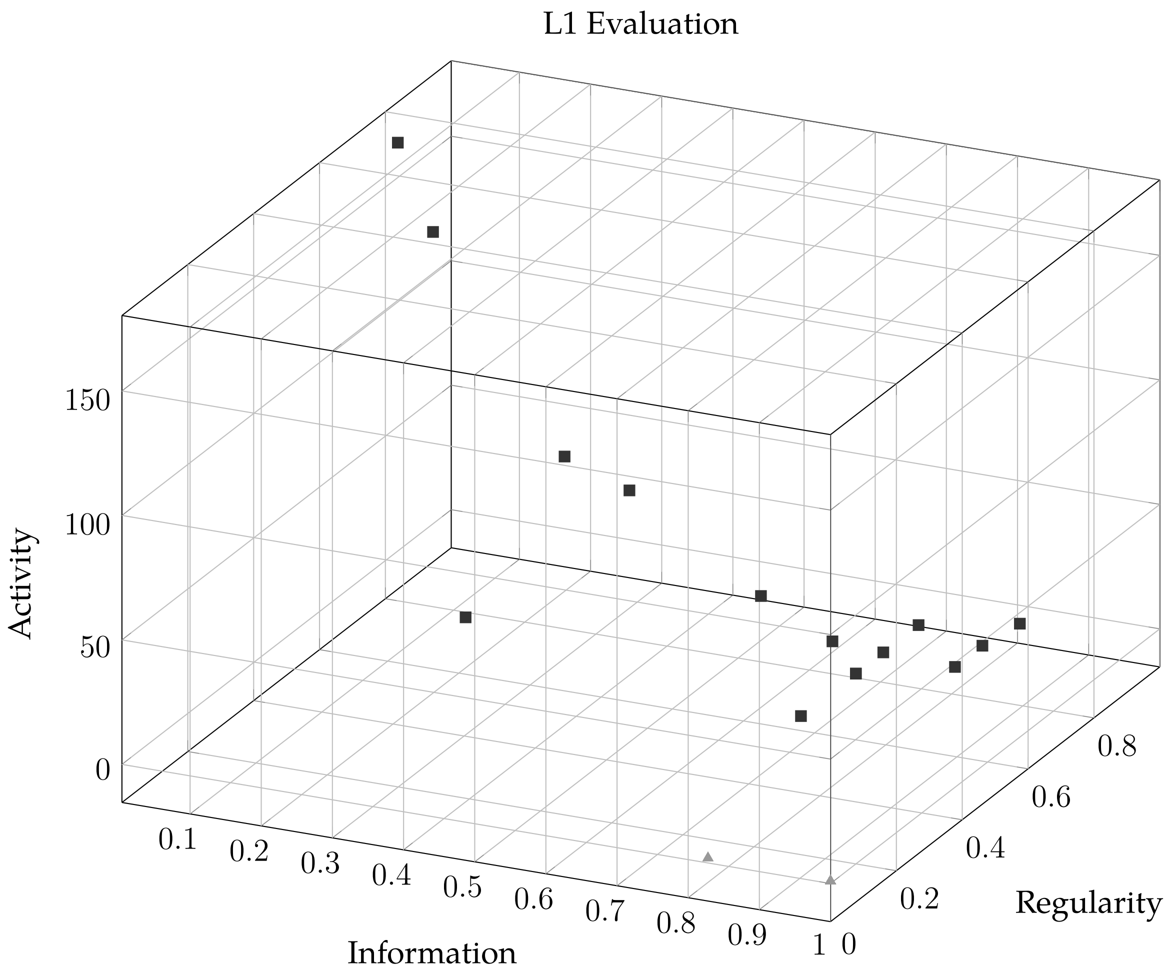
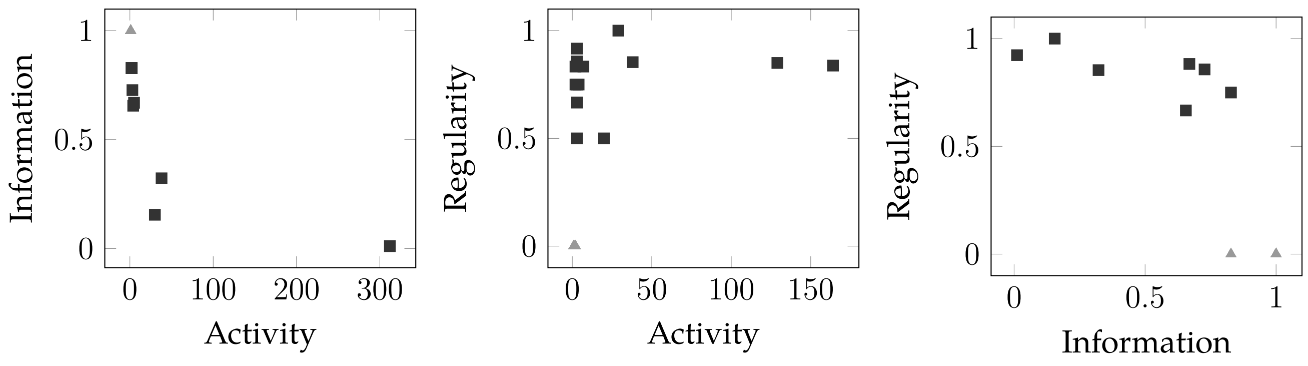


Figure 5.19: L1 3D evaluation comparison



(a) L2 Activity vs. Information (b) L2 Activity vs. Regularity (c) L2 Information vs. Regularity

Figure 5.20: L2 2D evaluation comparison

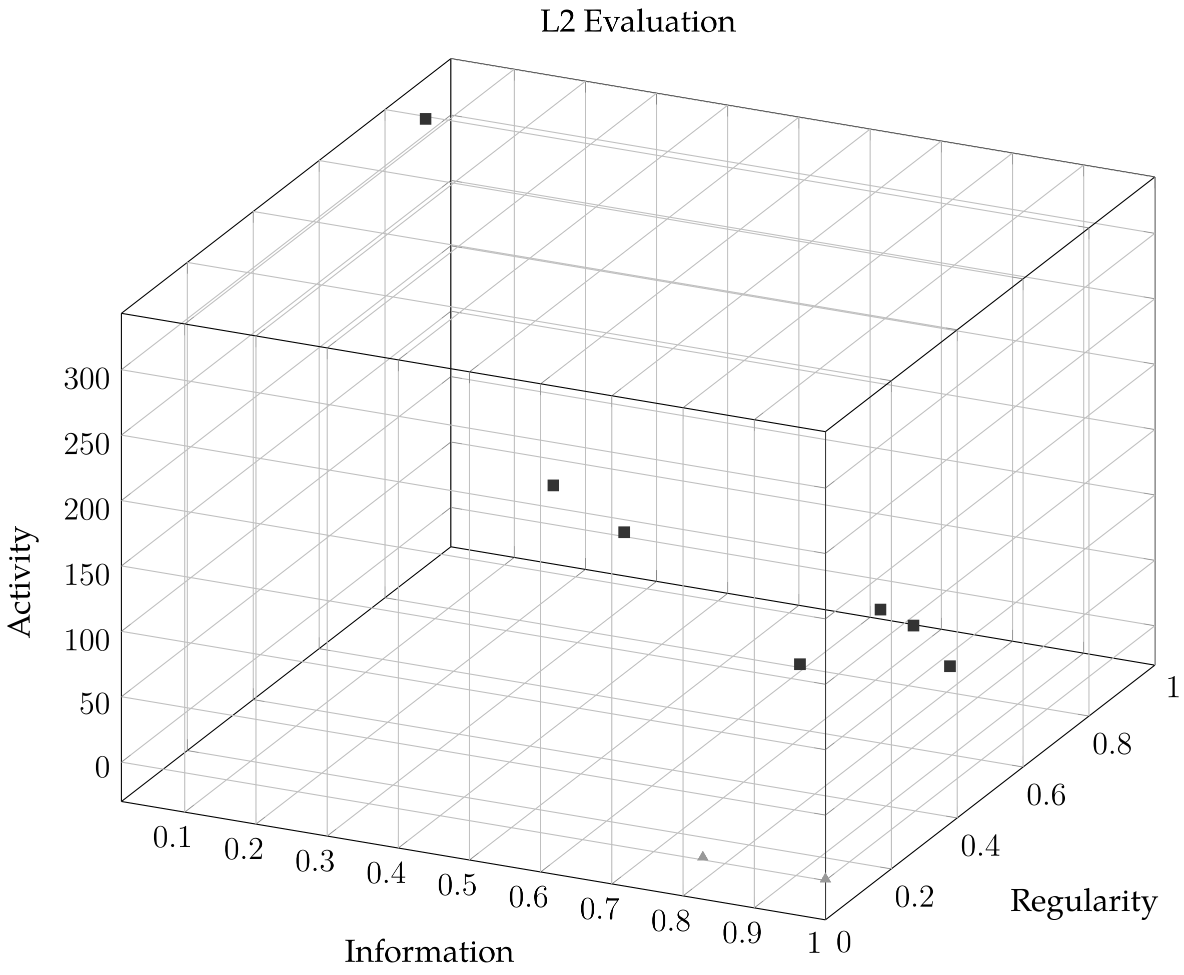


Figure 5.21: L2 3D evaluation comparison

# Chapter 6

# Conclusion

## 6.1 Contribution and summary

In this work, a person's timeline of observed WiFi signals were collected by a mobile device, through an Android mobile application that was created for the purpose of this project. The Android application was implemented in Kotlin and collected readings on a minute interval, using the WiFiManager library.

The hiererchical timeline clustering algorithm presented, proved to be an efficient means to clustering the timelime of readings in an online manner. The performance of the algorithm on real data, showed that the time for appending a new reading to the hierarchy held no dependency on the amount of readings in the hierarchy, thus making it a suitable choice for online clustering of time series data.

Segmentation of the hierarchical timeline consisted of identifying clusters in the hierarchy that consisted of a similarity measure between the children of a cluster being compared to a threshold. The segmentation was improved upon in implementation by using a minimum similiarity function that sampled the readings contained in a cluster, rather than computing the similarity between all readings, and cached values so similar computations were not re-evaluated.

The segmented timeline provided the input for creating the physical localization, which was then used as the input for the semantically grouped localizations. The semantics for the physical locations, were obtained from the SSID values of hotspots which are user defined values, typically representative of the organization, business or home it belongs to.

The localizations generated from the real data were evaluated by the three measures: activity, normalized information and regularity, which showed that semantically grouped locations were an improvement on the physically grouped locations.

The visualizations created from the localizations and the timeline, provided a succinct summary of the mobility pattern that represents the person's movement.

## 6.2 Lessons learned

This project provided lessons learned in mobile application development, data analysis and data visualization.

### 6.2.1 Mobile application development

The initial prototype of the data collection mobile application was created natively for Android using Java. Even though this was a stable solution, another iteration of the application was created using Kotlin [9, 25]. Kotlin allowed for the creation of a native Android application, as it compiles into Java byte code, but the language is far more concise and elegant than Java. The ANKO library available in Kotlin, provided the structures necessary for Android development in a clean and efficient manner. The Kotlin made application was made with a significantly smaller amount of code than the Java implementation, but resulted in a far more maintain-able system.

### 6.2.2 Data analysis

This project provided valuable education in obtaining and analysis of data.

The process of scanning for WiFi signals and storing them in an SQLite database, and then finding a means of transferring that information provided it's own set of challenges; size of data, format of data and method of transfer.

Data analysis within the project, provided the opportunity to become familiar with Pythons extensive libraries for data analysis and the *pandas* library [34]. *Pandas* offers a data structure that is common to data science programming languages and libraries: data frames. Data frames are an essential data structure for data science, and the implementation in the pandas library includes all the required data frame functions to make manipulation and transformation of data sets concise and elegant in Python.

Research into information theory provided a means to establish the normalized information content evaluation measure, and research into machine learning lead to the regularity evaluation measure. The topic of machine learning provided a plethora of different algorithms to experiment with useing scikit learn [37], and once the decision to use neural networks as the regularity measure was made, TensorFlow [1] with Keras [4] was utilized.

### 6.2.3 Data visualization

Initial experimental visualizations were created with matplotlib [17], which provided quick visual feedback and allowed for initial insights into the data collected.

Displaying the mobility patterns in the form of transition diagrams proved to be too complex for the traditional graphing libraries available in Python. This lead to research into D3.js [35]. D3.js allowed for the creation of interative visualizations; specifically, the transition diagrams seen in Chapter 3 and 5. The transition diagrams were created with force layouts in D3.js which allowed the interactive visualization to optimize the position of the nodes and edges in a force similation and take user input to manipulate the position further.

## 6.3 Future work

Possible future work includes: applying the algorithms to other time series data, implementing the algorithms on mobile devices, and integrating the project into mobile applications to enhance a person's experience.

Other time series data from different sensors that exist on mobile devices, or devices that can benefit from locally stored and analyzed data would benefit from these algorithms and should be explored.

Implementing the algorithms directly on the mobile device to test performance at different frequency of gathering sensor data is a possible area to explore, and finding ways to enhance applications with predictive knowledge of a person's patterns extracted from the sensor data.

# Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Bahl, P., and Padmanabhan, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, Ieee, pp. 775–784.

[3] Calabrese, F., Diao, M., Di Lorenzo, G., Ferreira, J., and Ratti, C. Understanding individual mobility patterns from urban sensing data: A mobile phone trace example. *Transportation research part C: emerging technologies 26* (2013), 301–313.

[4] Chollet, F., et al. Keras. `https://github.com/fchollet/keras`, 2015.

[5] Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., and Yang, C. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering 13*, 1 (2001), 64–78.

[6] Constandache, I., Gaonkar, S., Sayler, M., Choudhury, R. R., and Cox, L. Enloc: Energy-efficient localization for mobile phones. In *INFOCOM 2009, IEEE* (2009), IEEE, pp. 2716–2720.

[7] Delafontaine, M., Versichele, M., Neutens, T., and Van de Weghe, N. Analysing spatiotemporal sequences in bluetooth tracking data. *Applied Geography 34* (2012), 659–668.

[8] Diao, M., Zhu, Y., Ferreira Jr, J., and Ratti, C. Inferring individual daily activities from mobile phone traces: A boston example. *Environment and Planning B: Planning and Design 43*, 5 (2016), 920–940.

[9] Dmitry Jemerov, S. I. *Kotlin in Action*. Manning Publications Co., 2016.

[10] Feng, C., Au, W. S. A., Valaee, S., and Tan, Z. Received-signal-strength-based indoor positioning using compressive sensing. *Mobile Computing, IEEE Transactions on 11*, 12 (2012), 1983–1993.

[11] Ferris, B., Haehnel, D., and Fox, D. Gaussian processes for signal strength-based location estimation. In *In proc. of robotics science and systems* (2006), Citeseer.

[12] Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. Understanding individual human mobility patterns. *arXiv preprint arXiv:0806.1256* (2008).

[13] Guo, D., Zhu, X., Jin, H., Gao, P., and Andris, C. Discovering spatial patterns in origin-destination mobility data. *Transactions in GIS 16*, 3 (2012), 411–429.

[14] Hatami, A., and Pahlavan, K. A comparative performance evaluation of rss-based positioning algorithms used in wlan networks. In *Wireless Communications and Networking Conference, 2005 IEEE* (2005), vol. 4, IEEE, pp. 2331–2337.

[15] Hedrick, A., Pu, K. Q., and Zhu, Y. Hierarchical temporal mobility analysis with semantic labeling. In *Computational Science and Computational Intelligence (CSCI), 2016 International Conference on* (2016), IEEE, pp. 1321–1326.

[16] Hedrick, A., Zhu, Y., and Pu, K. Modeling transition and mobility patterns. In *International Conference on Applied Human Factors and Ergonomics* (2017), Springer, Cham, pp. 528–537.

[17] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering 9*, 3 (2007), 90–95.

[18] Isaacman, S., Becker, R., Cáceres, R., Kobourov, S., Martonosi, M., Rowland, J., and Varshavsky, A. Identifying important places in people's lives from cellular network data. In *International Conference on Pervasive Computing* (2011), Springer, Berlin, Heidelberg, pp. 133–151.

[19] Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., and Shahabi, C. Big data and its technical challenges. *Commun. ACM 57*, 7 (July 2014), 86–94.

[20] Jensen, B. S., Larsen, J. E., Jensen, K., Larsen, J., and Hansen, L. K. Estimating human predictability from mobile sensor data. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on* (2010), IEEE, pp. 196–201.

[21] Jiang, S., Guan, W., Zhang, W., Chen, X., and Yang, L. Human mobility in space from three modes of public transportation. *Physica A: Statistical Mechanics and its Applications 483* (2017), 227–238.

[22] Kaemarungsi, K., and Krishnamurthy, P. Modeling of indoor positioning systems based on location fingerprinting. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (2004), vol. 2, IEEE, pp. 1012–1022.

[23] Kim, D. H., Kim, Y., Estrin, D., and Srivastava, M. B. Sensloc: Sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2010), SenSys '10, ACM, pp. 43–56.

[24] Kushki, A., Plataniotis, K. N., and Venetsanopoulos, A. N. Kernel-based positioning in wireless local area networks. *Mobile Computing, IEEE Transactions on 6*, 6 (2007), 689–705.

[25] Leiva, A. *Kotlin for Android Developers*. Leanpub, 2016.

[26] Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge University Press, 2014.

[27] Letchner, J., Fox, D., and LaMarca, A. Large-scale localization from wireless signal strength. In *Proceedings of the national conference on artificial intelligence* (2005), vol. 20, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 15.

[28] Li, B., Salter, J., Dempster, A. G., and Rizos, C. Indoor positioning techniques based on wireless lan. In *LAN, First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications* (2006), Citeseer.

[29] Lin, K.-S., Wong, A. K.-S., Wong, T.-L., and Lea, C.-T. Adaptive wifi positioning system with unsupervised map construction. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (2015), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 636.

[30] Liu, K. R. Signal processing techniques in network-aided positioning. *IEEE Signal Processing Magazine 1053*, 5888/05 (2005).

[31] Liu, T., Bahl, P., and Chlamtac, I. A hierarchical position-prediction algorithm for efficient management of resources in cellular networks. In *Global Telecommunications Conference, 1997. GLOBECOM'97., IEEE* (1997), vol. 2, IEEE, pp. 982–986.

[32] Ma, J., Li, X., Tao, X., and Lu, J. Cluster filtered knn: A wlan-based indoor positioning scheme. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a* (2008), IEEE, pp. 1–8.

[33] Mayorga, C. L. F., Della Rosa, F., Wardana, S. A., Simone, G., Raynal, M. C. N., Figueiras, J., and Frattasi, S. Cooperative positioning techniques for mobile localization in 4g cellular networks. In *Pervasive Services, IEEE International Conference on* (2007), IEEE, pp. 39–44.

[34] McKinney, W. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 51 – 56.

[35] Murray, S. *Interactive Data Visualization for the Web.* O'Reilly Media, Inc., 2013.

[36] Paul, A. S., and Wan, E. A. Wi-fi based indoor localization and tracking using sigma-point kalman filtering methods. In *Position, Location and Navigation Symposium, 2008 IEEE/ION* (2008), IEEE, pp. 646–659.

[37] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[38] Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (2000), ACM, pp. 32–43.

[39] Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., and Sievänen, J. A probabilistic approach to wlan user location estimation. *International Journal of Wireless Information Networks 9*, 3 (2002), 155–164.

[40] Seshadri, V., Zaruba, G. V., and Huber, M. A bayesian sampling approach to in-door localization of wireless devices using received signal strength indication. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on* (2005), IEEE, pp. 75–84.

[41] Smailagic, A., and Kogan, D. Location sensing and privacy in a context-aware computing environment. *Wireless Communications, IEEE 9*, 5 (2002), 10–17.

[42] Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. Limits of predictability in human mobility. *Science 327*, 5968 (2010), 1018–1021.

[43] Versichele, M., Neutens, T., Delafontaine, M., and Van de Weghe, N. The use of bluetooth for analysing spatiotemporal dynamics of human movement

at mass events: A case study of the ghent festivities. *Applied Geography 32*, 2 (2012), 208–220.

[44] Xu, Y., Shaw, S.-L., Zhao, Z., Yin, L., Fang, Z., and Li, Q. Understanding aggregate human mobility patterns using passive mobile phone location data: A home-based approach. *Transportation 42*, 4 (2015), 625–646.

[45] Youssef, M. A., Agrawala, A., and Shankar, A. U. Wlan location determination via clustering and probability distributions. In *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on* (2003), IEEE, pp. 143–150.