Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Congzheng Song April 11, 2016

Using Deep Recurrent Neural Networks to Estimate Influenza Prevalence from Mobile Phone Records

by

Congzheng Song

Ymir Vigfusson
Adviser

Department of Mathematics and Computer Science

Ymir Vigfusson
Adviser

Gari Clifford
Committee Member

Shamim Nemati
Committee Member

2016

Using Deep Recurrent Neural Networks to Estimate Influenza Prevalence from Mobile Phone Records

by

Congzheng Song

Ymir Vigfusson
Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelors of Science with Honors

Department of Mathematics and Computer Science

2016

Abstract

Using Deep Recurrent Neural Networks to Estimate Influenza Prevalence from Mobile Phone Records

By Congzheng Song

Early detection of Influenza would save millions of people from suffering and death, however the detection itself still remains challenging. Previous influenza surveillance systems require the clinical data of infected individuals or search queries about Influenza which depend heavily on Internet usage. On the other hand, another data resource, mobile phone data, remains popular globally. The hypothesis in our project is that we can use mobile phone data to model the individual behavioral change and apply them to a larger population with unknown prevalence to accurately detect new diseases or epidemics in their early stages. In this thesis, we will focus on a more specific question towards validating the hypothesis. That is, can we detect human behavioral changes using mobile phone data and to use them to build an appropriate individual sickness prediction model?

To answer this question, we first define several metrics that can be extracted from mobile phone data and show that they exhibit the behavioral changes when people are sick. Next, we setup a supervised learning task where we want to predict when a mobile phone user will be sick given the set of metrics we define. We further develop a novel deep learning model for this task and show that our model outperform other machine learning models.

Using Deep Recurrent Neural Networks to Estimate Influenza Prevalence from Mobile Phone Records

by

Congzheng Song

Ymir Vigfusson
Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelors of Science with Honors

Department of Mathematics and Computer Science

2016

Acknowledgment

First, I owe my deepest gratitude to Prof. Ymir Vigfusson for providing the support, resource and patient guidance. Without him this thesis cannot be done. I want to also thanks committee members: Prof. Gari Clifford and Prof. Shamim Nemati for lending their time, advice, and expertise for this thesis. This work would not have been possible without their direction. Furthermore, I thank Erik, Becky, Leon and everyone else who are also involved in this project for the discussion and assistance, and laughs.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Seasonal epidemics causes between three and five million cases of severe illness and between 250,000 and 500,000 deaths every year around the world [42]. In 2009, the H1N1 pandemic affected millions of people in the world and resulted in large number of fatalities. Early detection of Influenza would save millions of people from suffering and death, however the detection itself still remains challenging. Influenza has traditionally required symptomatic individuals to seek treatment or advice [21] and syndromic surveillance system [15] depends majorly on accurate both virologic and clinical data, including the number of physician visits, where there might be a lag of couple of days or even a week from onset of illness until affected people visiting doctor.

More recently, surveillance system comprising search queries [25] and social medias [48, 44] are proposed. These systems have enormous data that are already collected and thus are passive and non-invasive since they do not require additional individual medical records or interviews. However, these surveillance systems is that they rely on Internet usage. This limits their applicability in areas regularly prone to infectious disease outbreaks such as Africa. Globally, 3.2 billion people are using the Internet by end 2015, while 4 billion people from developing countries remain offline, representing 2/3 of the population residing in developing countries [10].

1.2 Motivation

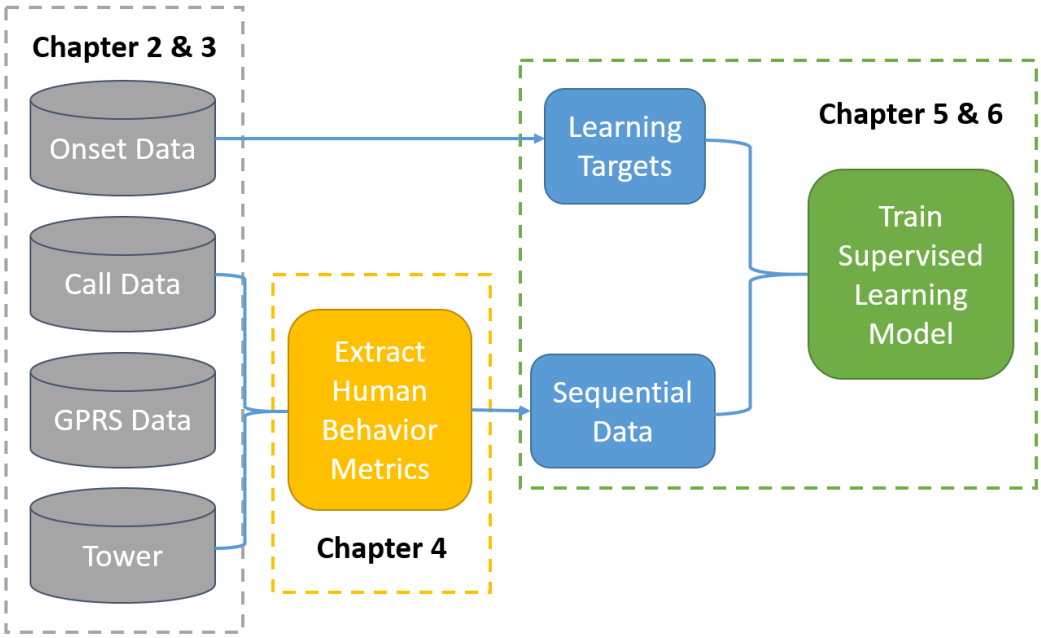
On the other hand, another data source remains popular and pervasive globally: mobile phone data. Mobile phone users have increased from 700 million in 2000 to 7 billion by the end of 2014, covering more than 96% of the world's population, almost the twice of Internet coverage. Mobile phone providers maintain Call Data Records (CDR) for all calls involving to their subscribers for billing purposes. Each record contains various attributes of the call, such as time, duration, completion status, source number, and destination number, which we will describe the data in details in Chapter 3.

The hypothesis in our project is that we can use CDR data to model the individual behavioral change and apply them to a larger population with unknown prevalence to accurately detect new diseases or epidemics in their early stages. Building automatic real-time system for Influenza surveillance using CDR data would be our ultimate goal. In this thesis, we will focus on a more specific question towards validating the hypothesis. That is, **can we detect human behavioral changes using CDR data and to use them to build an appropriate individual sickness prediction model?**

1.3 Outline

In this thesis, We first review previos literature on modeling human behavior and conducting various predictions task using mobile phone data in Chapter 2. Then we describe in detail our CDR dataset as well as the labeled patients' data in Chapter 3. Following that in Chapter 4, we provide a number of metrics that are extracted from CDR data and are helpful in describing human behaviors as well as the changes in behavior during people are sick. We will provide a deep learning model that utilizes the behavior metrics to predict individual sickness in Chapter 5. In the end, we show our experiments methods and current results for sickness prediction in Chapter 6.

Figure 1.1: Components of architectural pipeline, and how they are described in this thesis



Chapter 2

Related Work

In this chapter, we will review studies on using mobile phone data to understand human behavior and perform various prediction task. Also, we will review latest trends in applying deep learning to healthcare problem.

2.1 Human Behavior Study

Many studies have tried to model human behavior from mobile phone data. Gonzalez et al. [27] showed that human trajectories exhibit a high degree of temporal and spatial regularity. Farrahi and Gatica-Perez [18, 19] successfully discovered various types of human daily routines using probabilistic topic modeling [5]. Cho et al. [14] demonstrated that humans experience a combination of periodic movement that is geographically limited. By measuring the entropy of each individual's trajectory, Song et al. [49] find a 93% potential predictability in user mobility across the whole user base, regardless of the significant differences in the travel patterns. All these researches have discovered the regularity of human behavior pattern to some extent and showed the feasibility of using mobile phone data to quantify these pattern.

2.2 Classification Using CDR data

Other researches showed the potential that human behavior can be quantified using mobile phone data and used for further classification task. Frias-Martinez et al. [22] used mobile phone data for characterizing and automatically identifying the gender of a cell phone user in a developing economy based on behavioral, social and mobility variables. de Montjoye et al. [17] showed that personality can be reliably predicted from standard mobile phone logs by using a set of novel psychology-informed indicators that can be computed from mobile phone data. Bogomolov et al. [6] applied metrics extracted from mobile phone and demographic data to predicting crime in a geographic space. We will adapt some of the metrics developed in these works and analyzing them to see if these metrics are able to exhibit the deviation of behaviors from regular patterns during people are sick.

2.3 Differences in Our Work

Most of these previous works on CDR data are using models like SVM [16] or decision tree [8]. Deep learning, one of the latest trend in machine learning, has been proved to be successful in many different areas [38]. Thought there are some attempts to use deep learning at health-care [39] or CDR data [20], we have not seen any work on the combination of both, which is to use deep learning to prediction sickness using CDR data. Thus, we would like to build a deep learning model based on our knowledge of human behavior and test that if deep learning captures the changes in behavior better than other model in terms of prediction accuracy.

Chapter 3

Dataset Description

In this chapter, we will provide a detailed description of our dataset, including CDR data and data for onset dates of mobile phone users during 2009 H1N1 Influenza. We then joint these two dataset and give some basic statistics of our data.

3.1 Icelandic CDR Data

Use mobile phone data from Iceland to analyze behavior changes during Influenza have several advantages. First, Iceland has near perfect mobile phone usage, and the population are homogeneous from which we can expect the behavior patterns would be similar. Also, Iceland is an island and thus there would be less migration in population. However, since Iceland is a small country with many unique qualities, the generalization of our analysis on other country is yet to be tested.

We are fortunate to have access to fully anonymized CDR data for cell phone use in Iceland from October 2008 to 2012 by the Icelandic telecom provider Siminn, whose subscribers covers 30% to 50% population in Iceland. The data are stored in three tables. We show some examples of records from the data.

Call Data

Each row of the table is a call detailed record (CDR). The first two columns are user ids that are involved in this record. If entry type is 11, then a called b. If entry type is 31,

ID a	ID b	Time	Tarif	Entry Type	Tower	Tarif Type	Duration
113604**	113735**	2009-08-25 10:45:2*	GGSM	11	274010001004E	PREP	432
49815**	108928**	2009-08-25 20:32:3*	GGSM	11	27401001F03AB	PREP	93
52223**	107910**	2009-08-25 16:47:4*	GGSM2	11	27401000B029D	PREP	395
109492**	108117**	2009-08-25 19:39:0*	GGSM7	11	27401012D0C4C	POST	7

Table 3.1: Call Data Table Example

then b called a. If entry type is 1, then a texted b. If entry type is 0, then b texted a. Tower ID is the hex id for the cell phone tower in tower table, we will use it to get the GPS coordinates of that tower. Column Tarif or Tarif Type are information for the phone account type, which are of not our interest in this project. Duration is the call duration and it would be zero if the row is a record for text.

GPRS Data

ID	Time	Tower	Prepay	Upload Volume	Download Volume	Total Volume
113675**	2009-02-01 08:12:3*	439	0	4370	7724	12094
108265**	2009-02-01 09:26:0*	3046	0	107697	937039	1044736
113624**	2009-02-01 07:04:2*	460	0	9215	12632	21847
108265**	2009-02-01 08:16:0*	3046	0	160495	884349	1044844

Table 3.2: GPRS Data Table Example

For each row, GPRS data table has user id and corresponding tower id and usage of data, where the total volume is the same as the sum of upload volume and download volume. **Prepay** column remain largely static throughout the data.

Cell Tower

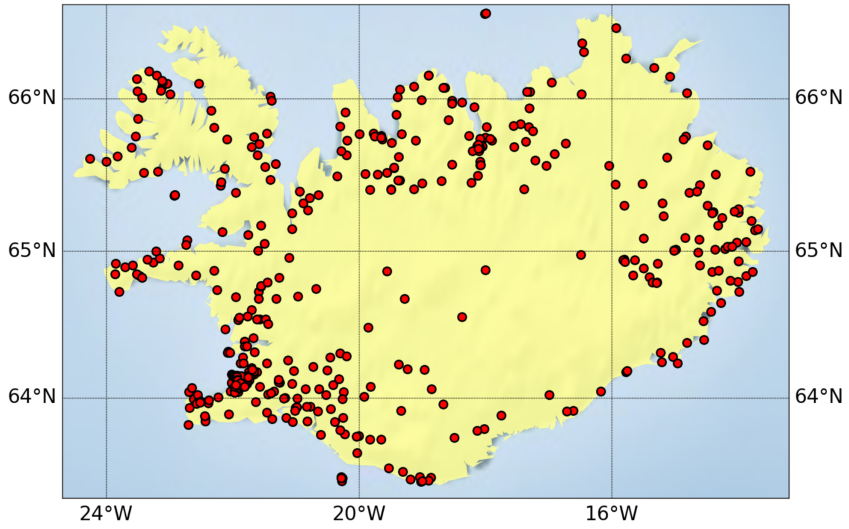
Every tower table row contains tower id and hex tower id, which are corresponding to the tower id in Call Data table and GPRS Data table respectively. Also, we have their GPS coordinate in latitude and longitude. **Loc** column describe the region where the tower is located. Notice that the first three rows share the same GPS coordinate yet they are marked as different tower by **Tower ID** and **Cell** columns. This is because GSM cell tower

Tower ID hex	Tower ID	Loc	Cell	Lat	Lon
0001	2740100010001	Öskjuhlíð	OSKJUHN	64.127**	-21.905**
0002	2740100010002	Öskjuhlíð	OSKJUHA	64.127**	-21.905**
0003	2740100010003	Öskjuhlíð	OSKJUHV	64.127**	-21.905**
0004	2740100010004	Kristkirkja	KRISTKN	64.147**	-21.948**

Table 3.3: Tower Data Table Example

has three different angles, each covering 120 degrees and thus in total the three directed antennas on the tower can cover 360 degrees.

Figure 3.1: Cell Tower Distribution in Iceland

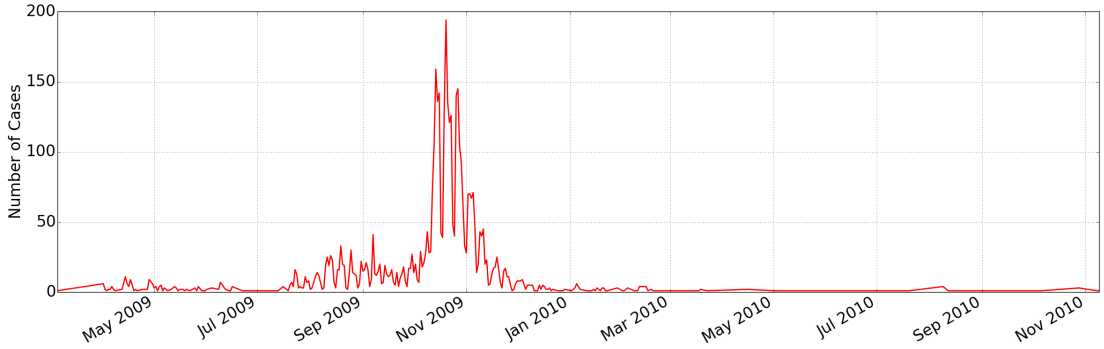


Here in Figure 3.1 we plot the geographical distribution of cell towers in Iceland using the GPS coordinates in tower table. As we can see, most towers are in the coastline and there the density of towers in , Reykjavík ,the capital is the highest since that is where most people live.

3.2 Onset Data

Through collaboration with the Icelandic Centre for Health Security and Communicable Disease Control at the Directorate of Health (CHSCDC), we have been given access to a subset of anonymous individual electronic medical records from 2009 H1N1 Influenza.

Figure 3.2: Number of Onset Cases Over Time



Of all diagnosed cases, it is possible to match **4,294** individuals to traces in the CDR, comprising a total of over 6.5 million mobile data records.

3.3 Privacy and Approvals

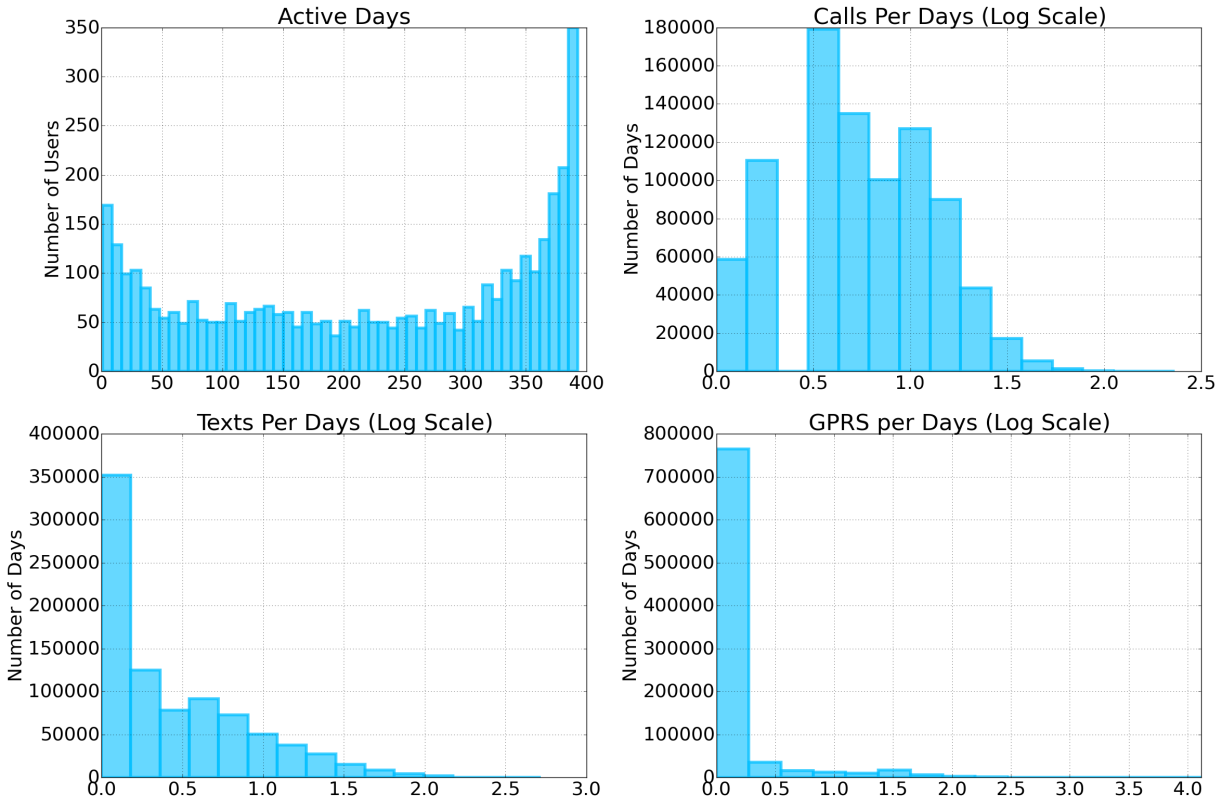
Individual anonymity and respecting privacywas preserved while generating this dataset. Phone numbers were encrypted using industry standard software by collaborators at Siminn. These anonymous IDs were then made available to the CHSCDC along with the standard Icelandic national identification number ('Kennitala') of each of the end users. CHSCDC used these to match the encrypted phone numbers to health records, generating a fully anonymous dataset that was then released to us.

Since all of the data has been anonymized and does not contain personally identifiable information, the Iceland Data Protection Authority (Persónuvernd) has stated that it requires no approval to conduct the proposed research.

3.4 Grouping Data

To utilize all the data we have for the set of patient, we decide to merge the call data and GPRS data for each patient. Since both call data and GPRS data provide time stamp and GPS coordinate, grouping them together will give us a higher resolution about when the user has been where. We then for each user, group their data by day since human behavior tend to be regular on daily basis. We plot some distributions of the raw grouped

Figure 3.3: Raw Distributions for Grouped Data. Number of calls, texts and GPRS are scaled by log function with base 10. Though most of the days have little texts or GPRS records, we have a set of users who have calls regularly, providing enough data for further analysis.



data in Figure 3.3.

Our time period of interest is days around 2009 H1N1 outbreak in Iceland. Thus, we only collect data from February 2009 to March 2010 for further analysis. Within this period, we have **3,931** users who have received Influenza-like illness diagnosis. These user have total **868,033** days active in aggregation during this period.

Chapter 4

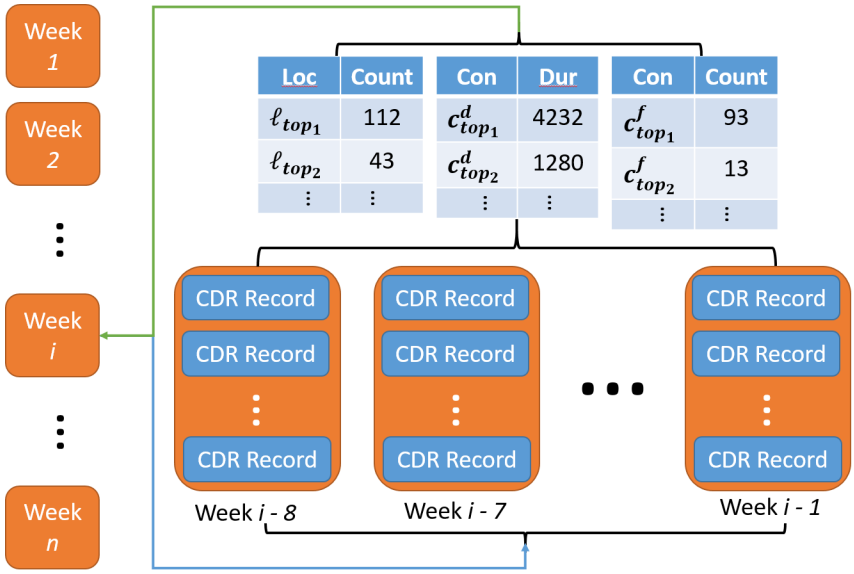
Human Behavior Metrics

Significant effort[17, 27] has been made to quantify human behavior given CDR data. Here we show our set of metrics that we believe are useful to describe human behavioral changes when they are sick. Given a user u and a time period T (e.g. each consecutive three hours or a day) and all the mobile phone records $r_1, r_2, ..., r_n$ that u made in T, we want to extract metrics from the records. These records can be used to describe the behavior of the user. Once we have a thorough understanding of the behavior through these metrics, we can use them in various prediction tasks. In our case, it would be to predict the user's sickness given metrics for some time periods. A summary of all metrics is showed in Table 4.1.

4.1 Notation

We reduce each mobile phone record into the following notation: for record r_i, we have the time stamp t_i, the GPS location ℓ_i in a 2-tuple (ϕ_i, λ_i) with latitude ϕ_i and longitude λ_i, the call duration τ_i, and correspondent's id c_i. For all records in T, let $\mathcal{C}$ be the set of all unique contacts with whom u has interacted and $\mathcal{L}$ be the set of all unique locations that u visited. We define the function of counting as $f : \mathcal{E} \mapsto \mathbb{N}$, where $\mathcal{E}$ is the set of entities $\mathcal{C} \cup \mathcal{L}$ in which we are interested in. For example, $f(c_i)$ would be the number of times that u has calls or texts with user c_i. We also define another function $g : \mathcal{C} \mapsto \mathbb{N}$ and $g(c_i)$ is the call duration $\sum \tau$ between u and c_i during T.

Figure 4.1: Work Flow for Extract Top Information: For each week i in u, group previous eight weeks records for i and rank the location visited by count, contact by call duration and count. Save the top k information for week i for later metrics extraction.



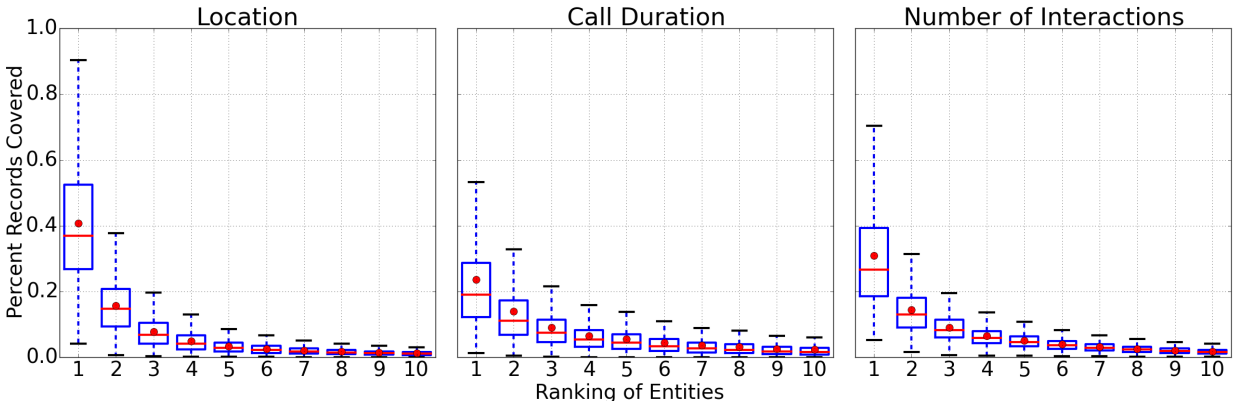
4.2 Past Top Information

Some basic top information that can be extracted from all the records from time periods T_p which precede T within some limit (e.g. see example for past 8 weeks in Figure 4.1). The reason why we extract top information is that we want some measurement of the regularity of human behavior. For example, given a week of interest, we extract the most frequently visited location ℓ from its previous 8 weeks, and measure how many time the user stay at ℓ for this week. This captures regularity in the user's whereabouts.

Top Locations We define $L_k = \{\ell_{top_1}, \ell_{top_2}, \ldots, \ell_{top_k}\}$ as the set of top k most frequently visited locations by u in T_p.

Top Contacts Similar to top locations, we also define two sets of top contacts ranked by call duration $C^d_k = \{c^d_{top_1}, c^d_{top_2}, \ldots, c^d_{top_k}\}$ and ranked by frequency of calls and texts $C^f_k = \{c^f_{top_1}, c^f_{top_2}, \ldots, c^f_{top_k}\}$ in T_p.

Figure 4.2: Percent Information Covered by Top Entities for All Past 8 Weeks for randomly sampled 100 users. The horizontal axis is the rank for the entity, e.g 1 means top 1 contact for Call Duration plot.



Choosing *k* An important parameter here we have to choose is how many (k as defined above) top entities we want to extract for each user. We want a k to be small since most people tend to talk most to one or two friends rather than everyone in their social network and they tend to stay at one or two locations more often then others. Thus, it is meaningless for us to have a large k since most of them would be rare cases.

To illustrate our point, we here plot percent of records covered by top information with k up to 10 in Figure 4.2. We can see that top one and two locations can cover 60% of the records, indicating most people tends to spend majority of time at few locations. On the other hand, user's contacts are more diverse. Thus, it is not very helpful to include top three contact and beyond by observing that they do not cover much records.

4.3 Call Pattern Metrics

We make an assumption that unhealthy people would make longer phone interactions during they are sick because they tend to make calls to their close friends or family members to tell about their conditions. However, we also expect that they would make less number of interactions and interact with fewer people since they might free from working and thus make less work calls.

4.3.1 Basic Phone Usage

These set of metrics provide some very basic description of user's phone usage behavior, which could effectively show the deviation of behavior during sick from normal.

Incoming/Outing Calls and Texts For all records in T, we first count the number of all incoming and outgoing calls and texts separately. Then, for calls, we sum up the incoming and outgoing call duration as well. When we have the duration and count for calls, we further calculate the average duration per incoming and outgoing call as another metric.

Call to Interaction Ratio We also want to determine if a user makes more calls or texts in T. Thus, we calculate ratio of the number of calls to number of all interactions (calls and texts).

Inter-Event Time For two consecutive records r_{i-1}, r_i and their time stamps t_{i-1}, t_i, we calculate the time elapsed in seconds $\Delta t_{i-1} = t_i - t_{i-1}$ as inter-event time. For all records in T, we thus get a sequence of inter-event time $[\Delta t_1, \Delta t_2, \ldots, \Delta t_{n-1}]$. We then compute the mean and standard deviation from the sequence as our metrics.

4.3.2 Social Behavior

As mentioned above, we deem that user would have less contacts but also interact more with their intimates. We find a set of metrics that can quantify these behaviors.

Unique Contact Interacted For correspondents in all records in T, we find the unique set of contact $\mathcal{C}$ that user called or texts and use $|\mathcal{C}|$ as our metric.

Contact Entropy For each unique contact c_i in $\mathcal{C}$, we find the corresponding frequency $f(c_i)$ of visiting c_i in T. Let $F_c = \sum_i f(c_i)$, we then calculate the Shannon entropy H_c of location by:

$$H_c = -\sum_i f(c_i) \log \frac{f(c_i)}{F_c}$$

We also do the same entropy calculation for unique incoming contacts and outgoing contacts.

Interaction with Top Contacts As we defined in 4.2, we have two sets of top contacts that we are interested in knowing how u interact with them. We first calculate $f(c^f_{top_i})$ and $g(c^d_{top_i})$ for $i = 1$ to k as metrics. Since the raw count of frequency and call duration might vary between users and time, we also calculate the percentage of interacting with top contacts. That is, we find the sum of frequency F_c and call duration D_c ($= \sum g(c)$) of communicating all contacts $c \in \mathcal{C}$. And then we use $\frac{f(c_{top_i})}{F_c}$ and $\frac{g(c_{top_i})}{D_c}$ for $i = 1$ to k as another set of metrics.

Contact to Interactions Ratio This metric measured by $\frac{|\mathcal{C}|}{\sum_{c\in\mathcal{C}} f(c)}$,would help us to determine how diverse the set of contacts of u is in T. We assume people would call fewer contacts but longer time during they are sick, since they would make much more phone call to strangers when working.

Percent Initiated Interactions Another metric that might help describing the behavior during sickness is the percent of people initiating the interactions, since people might call or text more to tell their friends and family they are sick. We calculate the percent initiated calls, texts and all interaction using outgoing interaction divided by all interaction.

4.4 Mobility Metrics

Movement pattern is another important set of behaviors that we are interested in. During people are sick, we assume that people would travel less and tend to stay at home due to the fact that they might feel uncomfortable and take day-off from work.

4.4.1 Traveling Diversity

We here define a set of location diversity measurements for users and we expect that the location diversity would decrease during people are sick.

Unique Locations Visited For locations in all records in T, we find the unique set of locations $\mathcal{L}$ that user visited and use $|\mathcal{L}|$ as our metric.

Location Entropy For each unique location ℓ_i in $\mathcal{L}$, we find the corresponding frequency $f(\ell_i)$ of visiting ℓ_i in T and apply the same method in Section 4.3.2 for calculating location entropy. If a user stay at only one place for entire time period, then the entropy would be zero. On the contrary, the location entropy would be high if the user visited a lot places.

Visits of Top Locations Similar to interacting with top contact, here we define the count and percent of visiting top towers. This could effectively tell us if user stay at places like home more than other places more during sickness.

Number of Transitions We define transition as a movement between different location. For example, at time stamp t_{i-1} and t_i, u moved from ℓ_{i-1} to ℓ_i. Then, if $\ell_{i-1} \neq \ell_i$, the tuple (ℓ_{i-1}, ℓ_i) count as a transition. We find the total amount of transitions in T as a metric to describe u's movement.

4.4.2 Movement Behavior

Apart from diversity, we also provide metrics that could directly quantify movement behavior, where the higher the metrics the more the user traveled. We believe that these measurements would drop during users' sickness.

Distance Traveled Given two consecutive records r_{i-1}, r_i and their GPS coordinates $(\phi_{i-1}, \lambda_{i-1})$, (ϕ_i, λ_i), we can calculate the great circle distance d_{i-1} in kilometer between them using Haversine formula, given $r = 6371$ km:

$$d_{i-1} = 2r \arcsin \sqrt{\sin^2(\frac{\phi_{i-1} - \phi_i}{360\pi}) + \cos \frac{\phi_{i-1}}{180\pi} \cdot \cos \frac{\phi_i}{180\pi} \cdot \sin^2(\frac{\lambda_{i-1} - \lambda_i}{360\pi})}$$

Thus, given a sequence of n records in T, we can extract a sequence of distances $[d_1, d_2, \ldots, d_{n-1}]$ between each two consecutive records. We further develop two metrics using the distance sequence. First, we consider the total distance traveled in T, which is

$\sum_{i=1}^{n-1} d_i$. Also, we add $\max_{i=1}^{n-1} d_i$ as another metric, which is the maximum distance between two locations that the user visited.

Mean and Variance of Speed From the sequences of distances we get above and the sequence of inter-event time we get from Section 4.3.1, we are able to calculate the speed $v_i = \frac{d_i}{\Delta t_i}$ during traveling each distance d_i for $i = 1$ to $n-1$ and thus we get a sequence of speed $[v_1, v_2, \ldots, v_{n-1}]$ in T.

Radius of Gyration We follow the same procedure of calculating radius of gyration proposed in [27]. That is, given the set of unique locations L and their corresponding visiting frequency in T, we measure the radius smallest circle that cover all these locations.

4.5 GPRS Data Usage

Though we do not find a clear relationship between sickness and GPRS data usage, including these records can help us increase = users' temporal and geographical resolution. Also, we consider the number of GPRS records the total upload volume and the total download volume in T as metrics to describe how u use GPRS data.

4.6 Categorical Variables

Apart from the sets of numerical variables we described above, we also define a set of categorical variables that applicable to all users.

Modal Location We first extract $\ell_{mode} = \arg\max_{\ell \in \mathcal{L}} f(\ell)$ and if $\ell_{mode} \in L_k$, we label mode location as k for $\ell_{mode} = \ell_{top_k}$, otherwise we label it as -1.

Modal Contact Similar to mode location, we find two different mode contacts $c^f_{mode} = \arg\max_{c \in \mathcal{C}} f(c)$ and $c^d_{mode} = \arg\max_{c \in \mathcal{C}} g(c)$ and label them accordingly with C^f_k and C^d_k

Modal Transition For all possible transitions (as we defined in Section 4.4.1) in T, we find the most frequent one, and label the two GPS coordinates with the same methods of labeling mode location. For example, (1, 2) would mean the that u travel mostly from u's first top location to second top location.

Modal Location-Contact Pair For each CDR records r_i, we pair the location ℓ_i and correspondent c_i as a tuple (ℓ_i, c_i) and we find the most frequent pair. We also label the pair using the same labeling methods above. Notice that we would have two categorical variables for location-contact pair, since the contact can be labeled by C_k^f and C_k^d and they have different meaning.

4.7 Analysis of Metrics

4.7.1 Box-Plots

To examine if these metrics extracted from CDR data could correctly show our assumption about what behavior would change during people are sick, we plot these metrics for the sick population with respect to the days within two weeks of their onset date. For each day in these two week range, we find all users who are active on that day and calculate the numerical behavior metrics accordingly and then generate the box-plot for each day.

On each day of interest, apart from statistics measured by box-plot, we also plot the Kolmogorov–Smirnov (K-S) distance between distribution of each metric on that day the the distribution of each metric for all users over all time period. Given two empirical distribution function $F(x)$ with m samples and $G(x)$ with n samples, K-S distance $D_{m,n}$ is calculated by $D_{m,n} = \max_x |F(x) - G(x)|$. This distance is indicated by the color of each box, where the greener the color indicate the larger the distance.

4.7.2 Observations

There are 58 of these time series box-plots, and we show them in Appendix A. From observations out of these plot, we can answer some of our assumptions below.

Do People Call More? Another set of behaviors that exhibit the deviation from normal during onset is call patterns and social behaviors. We can observe that people tend to talk longer on phone during they sick by looking at plots for average duration per call. However, number of calls is dropping slightly except for the day of onset, possibly due to people calling doctors. We could not observe clear changes in metrics related to text pattern.

Changes in Social Behavior? We have several metrics to describe user's social behavior, and from these we can observe that patients are interacting with fewer people except for the day on onset. Sick users also call their best friend longer in terms of raw duration, however, the percentage of calling best friend does not show significant difference. We deem this is due to their call duration also increases so that the percentage does not change much. Additionally, interaction with strangers increase drastically for the date of onset, which again might be attribute to interactions with doctors.

Do People Move Less? From plots for distance traveled, radius of gyration, location entropy, unique location so on, it is not hard to find out that these signals start to drop when close to onset date and achieve the lowest point at onset or the day after onset. Also, the K-S distance are larger during days close to onset than other days. It is very likely that during patients were sick, they were either told by the doctor that they should not travel much or they feel too uncomfortable to move around.

Changes in Daily Schedule? We are also be able to see that people spending more time at their top location and much less time at strange location. It is possibly due to the fact that the top location indicates home and thus many patients were taking sick-day off and spent more time at home.

On top of box-plots, we plot two heat-maps in Figure 4.3 and 4.4, where the x-axis is the days to onset and y-axis is hour and the color shows the percentage of staying at top locations. It is not hard to see that people spending more time at their top location in the day time during their sickness.

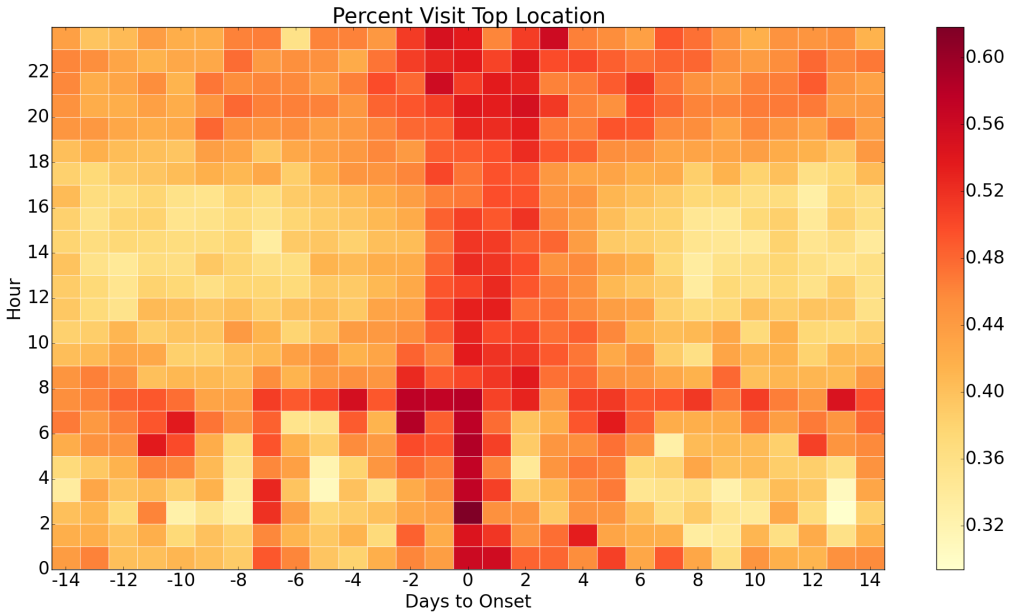


Figure 4.3: Mean percent of visits that are users' most frequently visited tower (possibly home location) during past 8 weeks.

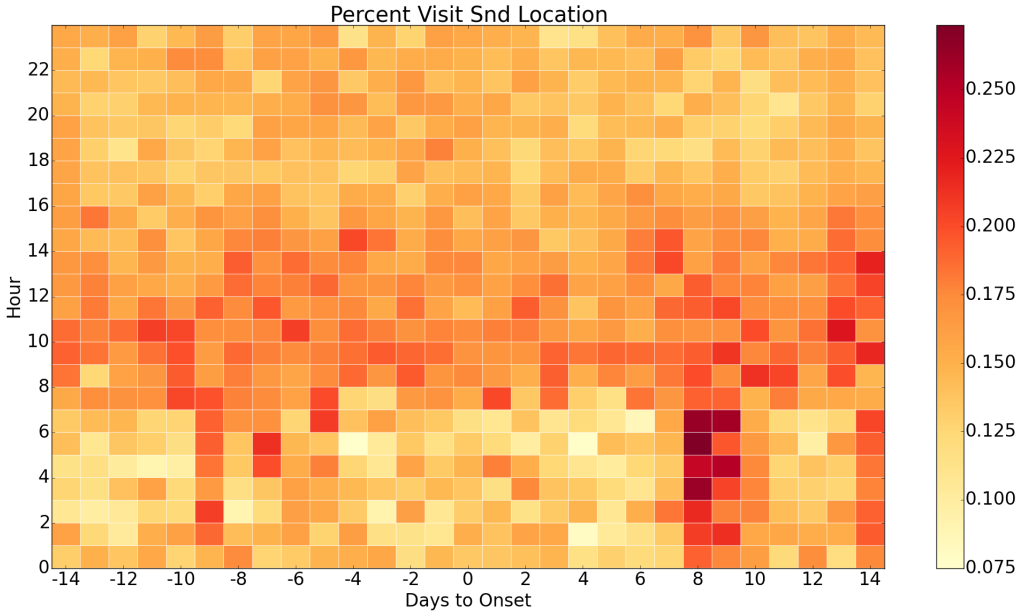


Figure 4.4: Mean percent of visits that are users' second most frequently visited tower (possibly work location) during past 8 weeks.

Table 4.1: Summary of Metrics

Type	Name	Unit
Mobility	Distance Traveled (Total, Max)	km
	Radius of Gyration	km
	Mean of Speed	km/sec
	Variance Speed	km^2/sec^2
	Unique Locations	count
	Location Entropy	N/A
	Percent at Top Locations	percent
Call Pattern	In/Out Calls, Texts	count
	In/Out Calls Duration (Total, Mean)	sec
	Contact Entropy (In, Out, All)	N/A
	Unique Contacts	count
	Contacts to Interaction Ratio	count/count
	Call to Interaction Ratio (In, Out, All)	count/count
	Percent Interact with Top Contacts	percent
	Mean of Inter-Event Time (Call, Text, All)	sec
	Variance of Inter-Event Time (Call, Text, All)	sec^2
GPRS	Number of GPRS Records	count
	Upload/Download Volume	bytes
Categorical	Mode Location	N/A
	Mode Contact	N/A
	Mode Location-Contact Pair	N/A
	Mode Transition	N/A

Chapter 5

Model

In this chapter, we first provide background knowledge for sequential modeling using deep learning. We then formalize our task as a supervised sequence classification problem and design a deep learning model that can be applied to our problem.

5.1 Preliminaries

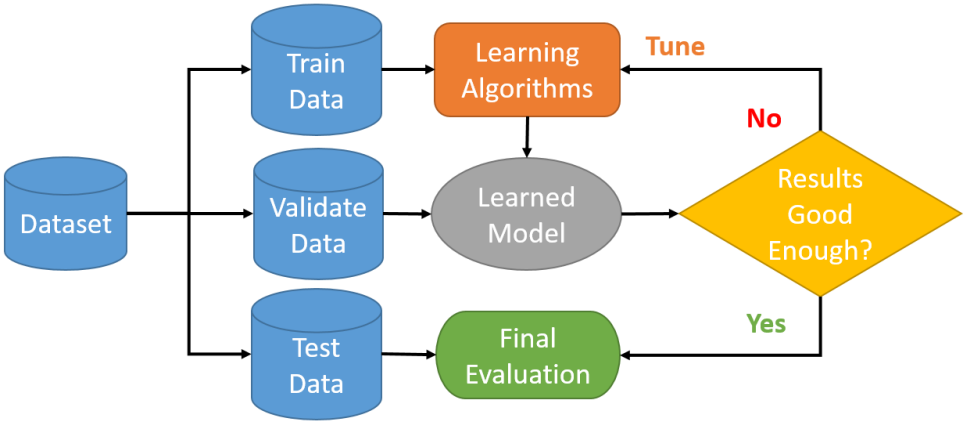
In this section, we will briefly review the problem setup for supervised learning and some machine learning algorithms for classification and sequence modeling.

5.1.1 Supervised Learning

Consider a dataset S with m data points $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)\}$, where each (x^i, y^i) is called a training example and x^i is a feature vector and y^i is the ground truth label (i.e, class). Supervised learning algorithm wants to learn a function (hypothesis) $f : \mathcal{X} \mapsto \mathcal{Y}$, where x^i is drawn from input space $\mathcal{X}$ and y^i is drawn from the output space $\mathcal{Y}$. In other words, the function is trying to predict y_i, given the input feature vector x_i.

To measure how well the function predicts the true values, we need a loss function determine the performance of the hypothesis, $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$. For training example (x^i, y^i), the loss is calculated by $L(y^i, h(x^i))$. During the training process, we will feed the input feature vectors into the hypothesis function and get a loss for each training example. Ideally, we want to minimize the loss over the whole data distribution $\mathcal{D}$ over

Figure 5.1: Supervised Learning Work Flow



$\mathcal{X} \times \mathcal{Y}$.

Another important concept in supervised learning is called generalization error, which measures the performance of the learned hypothesis on data that are not involved in the training process. If the generalization error is high for the hypothesis, then our model is not very useful for unseen data or new data and thus would have little usage in real world application.

A typical work flow in supervised learning setting is shown in Fig 5.1 . We will partition the datasets into training set, validation set and testing set. We train our model on training set and tune the parameters of the model based on the performance of validation set. Once we believe there are no more improvement can be made on validation set, we evaluate the generalization error on testing set.

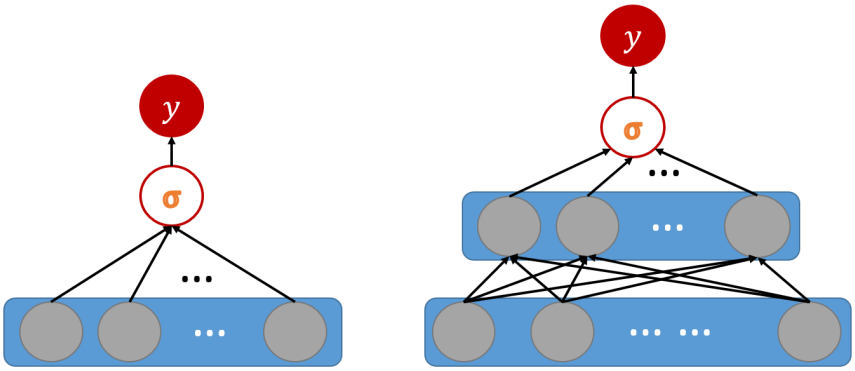
Logistic Regression

Consider a binary classification problem, where $\mathcal{Y} = \{0, 1\}$. Suppose we have a binary classification training example (x, y). In logistic regression, the hypothesis function $f(x; \theta) = \sigma(W^T x + b)$, where σ is known as sigmoid function or logistic function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The range of sigmoid function is a real value between 0 and 1. Then we make the following assumptions in logistic regression: $P(y = 1|x) = f(x; \theta)$ and $P(y = 0|x) = 1 - f(x; \theta)$. Thus, we would predict $y = 1$ whenever $P(y = 1|x) > P(y = 0|x)$ and $y = 0$ otherwise.

Figure 5.2: Graphical Representation for Logistic Regression(Left) and one hidden layer Multi-layer Perceptron(Right). Notice that the only difference between MLP and Logistic Regression is that MLP has an extra hidden layer in between input x and y.



Multi-Layer Perceptron

Multi-layer Perceptron (MLP) or Artificial Neural Network (ANN) can be represented graphically as in Fig 5.2.

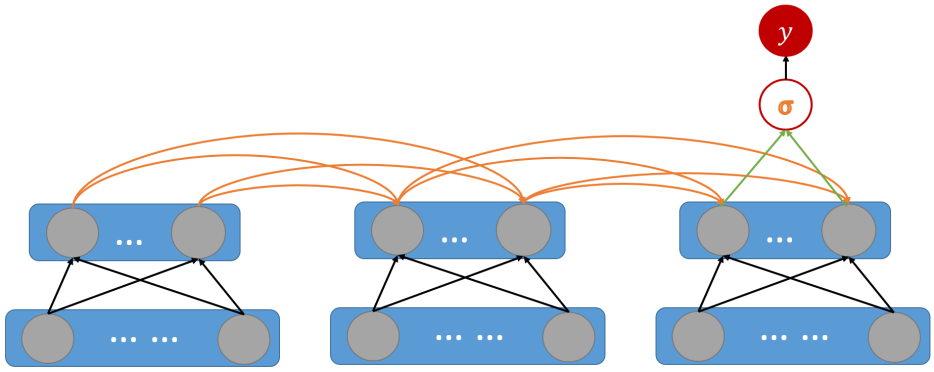
The connection between hidden layer and y is the same as logistic regression. However, there is an extract step that transforms input $x \in \mathbb{R}^n$ to hidden layer $h \in \mathbb{R}^k$ by a mapping function $g : \mathbb{R}^n \mapsto \mathbb{R}^k$. Usually $h = g(x) = a(W_1^T x + b^1)$, where $W_h \in \mathbb{R}^{n \times k}$ and $b^h \in \mathbb{R}^k$ and a is known as activation function. Typical choice for a are hyperbolic tangent, sigmoid function and more recently different kinds of rectifiers [26, 32]. After we obtain hidden layer h, we use it as an input to logistic regression to perform classification. The final output of $f(x;\theta) = \sigma(W_2^T a(W_1^T x + b^1) + b^2)$ where $W_2 \in \mathbb{R}^k$ $b_2 \in \mathbb{R}$ and $\theta = \{W_1, W_2, b_1, b_2\}$ are the set of parameters we learn from training.

The assumption in MLP is that it maps the input x to hidden layer through an nonlinear transformation which projects input data into a linear separable space where logistic regression can achieve better results.

5.1.2 Sequence Classification

Consider the input is now a sequence $x = [x_1, x_2, \ldots, x_s]$, where each $x_t \in \mathbb{R}^n$, and we still have $y \in \{0, 1\}$.Recurrent neural networks (RNN) are designed to model sequential data by taking each input vector at a time.

Figure 5.3: Vanilla Recurrent Neural Networks for sequence of three feature vectors. The different color of the line indicates different weight matrices. The current hidden layer and current input vector share the same weight matrix W while the current hidden layer and previous hidden layer share another weight matrix U.



Vanilla Recurrent Neural Networks

At time step (sequence index) t, the vanilla RNN will map the input vector to the hidden layer $h^t \in \mathbb{R}^k$ by a function $h_t = a(W^T x_t + U^T h_{t-1} + b)$, where h_{t-1} is the hidden layer from previous time step, $W \in \mathbb{R}^{n \times k}$, $U \in \mathbb{R}^{k \times k}$ and $b \in \mathbb{R}^k$. We essentially obtain a hidden layer sequence $h = [h_1, h_2, \ldots, h_s]$ after feeding the input sequence to RNN. Then, we use the last hidden layer, h_s, as the input to logistic regression to perform classification for the whole sequence.

Long Short-term Memory

LSTM [33] with forget gate [24] is proposed to capture long term dependency in the input sequence. Instead of storing information in hidden layer as vanilla RNN does, LSTM adds an additional memory cell that controls what information should be read, write and forget. The information flow are determined by three gates: input gate, output gate, forget, where the values are between 0 and 1 and parameters for the gates are learned through training.

Mathematically, LSTM can be represented by:

$$i_t = \sigma(W_{in}^T x_t + U_{in}^T h_{t-1} + b_{in}) \tag{5.1}$$

$$o_t = \sigma(W_{out}^T x_t + U_{out}^T h_{t-1} + b_{out}) \tag{5.2}$$

$$f_t = \sigma(W_f^T x_t + U_f^T h_{t-1} + b_f) \tag{5.3}$$

$$g_t = \tanh(W_c^T x_t + U_c^T h_{t-1} + b_c) \tag{5.4}$$

$$c_t = i_t \odot g_t + f_t \odot c_{t-1} \tag{5.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{5.6}$$

where $\odot$ denotes element-wise multiplication of two vectors, $i_t, o_t, f_t \in \mathbb{R}^k$ are the gates values, $g_t \in \mathbb{R}^k$ is the transformed incoming memory cell value, and $c_t \in \mathbb{R}^k$ is the final memory cell value determined by previous memory cell and incoming information flow, h_t is the hidden layer value calculated from output gate of memory cell. Same as vanilla RNN, we add a classifier which takes last hidden layer as input.

Bidirectional LSTM

Bidirectional LSTM [28] is first proposed for speech recognition. Instead of training one LSTM for sequence $x = [x_1, x_2, \ldots, x_s]$, we can flip the input sequence as $x^{back} = [x_s, x_{s-1}, \ldots, x_1]$ and train two LSTMs for both x and x^{back}. Then at each time stamp t, we have two hidden representation h_t and h_t^{back}. By concatenating them, we can get a representation contains both past and future information.

5.1.3 Gradient Based Learning

The loss function for probabilistic binary classification is typically cross entropy:

$$L(y, h(x;\theta)) = -(y \log(h(x;\theta)) + (1-y)\log(1-h(x;\theta))) \tag{5.7}$$

for training example (x, y). Suppose there are m training examples, we then want to minimize:

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(y^i \log(h(x^i;\theta)) + (1-y^i)\log(1-h(x^i;\theta)))$$

This optimization problem can be solved using gradient descent algorithm [12] which will be further covered in Section 6.2.2.

Back-Propagation

In neural network model, though we will still using gradient descent algorithm, the gradient of the parameters from input to hidden layer cannot be calculated directly from the loss function. Instead, we will use back-propagation algorithm [46] to calculate the gradient of the loss with respect to the parameters W^h and b^h.

Optimization in RNN

Vanilla RNN is not widely used due to the problem during loss minimization. Back-propagation through time (BPTT) [51] is the common algorithm for RNN to perform gradient descent. During BPTT, the gradient signal is multiplied by the weight matrix associated with the connections between the hidden layers at different time stamps. Thus for Vanilla RNN, derivatives are susceptible to vanishing if the weights are small, or exploding if weights are large [3].

Though LSTM has solved gradient vanishing problem by gating the information flow in memory cell so that activation of memory cell does not overwrite by inputs [30]. It still has the gradient exploding problem. To solve this problem, gradient clipping [43] is usually applied during training RNN.

5.2 Problem Formalization

Before applying the deep learning models to our dataset, we need to first define our problem in machine learning settings.

5.2.1 Sick Days Classification

Since we know the exact dates of the unhealthy mobile phone users went to doctor, it is very natural for us to create a supervised learning problem using the labeled days information. In the simplest setting, we can extract feature vector using the mobile phone record for each day for each unhealthy user, and then feed the feature vector into a binary classifier which predicts if the user goes to hospital or not on that day.

However, simply predicting if a user goes to hospital on certain day might not be a very appropriate target, since we want to capture the human behaviors that are related to sickness rather than the fact of going to hospital. Even though one is sick on a certain day, one might not go to doctor for many reasons. Also, if we only label the day of onset as positive class and all other days as negative class, we will end up having extremely imbalanced classes, which would hurt the performance of machine learning models.

Thus, we want to predict if the user is sick or not on a certain day given the user's previous mobile records. The problem now is to label the days that users are sick given their onset dates.

5.2.2 Labeling Sick Days

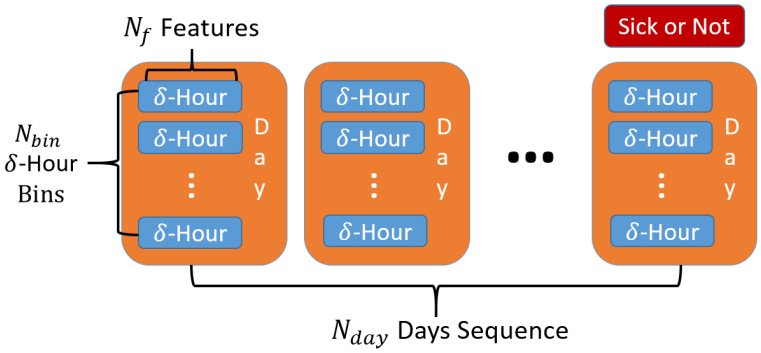
Many efforts have been made to study the changes in symptom severity of Influenza around the onset day [11, 34, 37]. In general, these studies showed that the total symptoms scores are high from 0 to 3 days after onset. Given the results in these studies, we choose to label the days that are 0 to 4 days after the onset day as sick days, and all the other days as healthy days.

5.2.3 Binning Records

After we have grouped data by user and by days, we decide to bin all the records in each day into segments of hours. The reason of doing it is that we think people tend to have a schedule on each day and they would behave differently in theses segments. For example, people are more likely to stay at home during 3 a.m. to 6 a.m. than 9 a.m to 12 p.m. Also, people tend to travel more during daytime than nighttime.

If we simply extract features for each day using all records in that day and ignore the time stamp, we would lose the periodicity of human behavior on each day. Thus, in order to capture the different behavior habit at different time, we bin the records for each day into N_{bin} evenly δ-hour segments, where $N_{bin} \times \delta = 24$. Then, we extract features for each of the δ-hour segments given the records.

Figure 5.4: Sequential Data Representation: Each blue box is a feature vectors for each δ-hour bin, each orange box contains N_{bin} blue boxes and each green box contains N_{day} orange boxes. The red box is the true label for the last day and also the whole sequence.



5.2.4 Sequential Data

The features for each δ-hour bin are extracted as the metrics we defined in Chapter 4. Thus, we have a feature vector with N_f dimensions for each δ-hour. Then, for each day, we will have N_{bin} of these feature vectors.

In many time series models or sequential models, it is a common scene that they will make use of the features in previous time period as part of the features for current time. In our case, we include the features from previous $N_{day} - 1$ days for each day, resulting in a N_{day} days sequence. The label of the sequence is the label of the last day in the sequence.

5.3 Proposed Methods

As we have setup the classification problem mentioned in previous section, we here propose a deep recurrent neural network architecture that can model our input sequential data and perform sick days classification.

5.3.1 Hierarchical LSTM

Since our data point is a sequence, we could choose to use normal LSTM to perform classifications. This can be done by flatten our N_{day}-days sequence into a sequence of

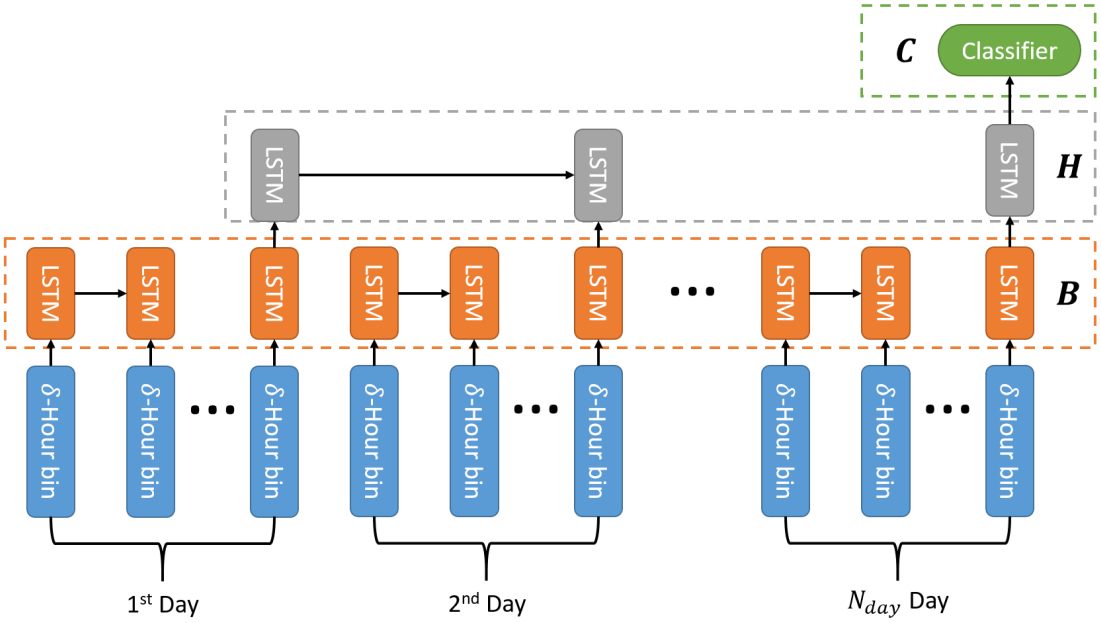


Figure 5.5: Hierarchical LSTM

$N_{day} \times N_{bin}$ δ-hour bins. However, this way, we are treating all δ-hour segments in a long linear chain, ignoring the periodicity of day.

To address the periodicity of daily schedule, we propose a new topology for LSTM called hierarchical LSTM, as showed in Fig 5.5. This is a two stage LSTM process, but is different from stacked LSTM. Our input sequence $x \in \mathbb{R}^{N_{day} \times N_{bin} \times N_f}$ is a three-way tensor:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1N_{bin}} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2N_{bin}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N_{day}1} & x_{N_{day}2} & x_{N_{day}3} & \dots & x_{N_{day}N_{bin}} \end{bmatrix}$$

where each $x_{ij} \in \mathbb{R}^{N_f}$ represents the feature vector for a δ-hour segments, i is the day index and j is the δ-hour bin index. Thus, ith $\mathbb{R}^{N_{bin} \times N_f}$ slice represents the sequence of δ-hour bin vectors for the ith day.

In the first stage of hierarchical LSTM, we build an bottom layer LSTM B for each $\mathbb{R}^{N_{bin} \times N_f}$ slice r_i in x. That is, we pass $r_i = [x_{i1}, x_{i2}, \dots, x_{iN_{bin}}]$ to B to get a representation for the ith day. After B processed r_i, we end up having a hidden representation $h_i^{\mathbf{B}} = [h_{i1}^{\mathbf{B}}, h_{i2}^{\mathbf{B}}, \dots, h_{iN_{bin}}^{\mathbf{B}}]$. Since hidden layer of LSTM contains long-term memory, we use the

last element in $h_i^{\mathbf{B}}$, $h_{iN_{bin}}^{\mathbf{B}}$ as a representation for day i so that it memorizes all the information for all δ-hour segments on that day. Thus, after pass all $i \in \{1, 2, \ldots, N_{day}\}$, we have a sequence s of representation for each day, where $s = [h_{1N_{bin}}^{\mathbf{B}}, h_{2N_{bin}}^{\mathbf{B}}, \ldots h_{N_{day}N_{bin}}^{\mathbf{B}}]$.

Next, we create a higher layer LSTM $\mathbf{H}$, which takes the sequence s as input. Similar to $\mathbf{B}$, $\mathbf{H}$ generates a hidden layer representation $h^{\mathbf{H}} = [h_1^{\mathbf{H}}, h_2^{\mathbf{H}}, \ldots, h_{N_{day}}^{\mathbf{H}}]$ after processing s. Again, to represent the sequence s, we use the last element of $h^{\mathbf{H}}$, which has memorized long-term information for the whole sequence. Thus, for sequence s, we get its final representation $h_{N_{day}}^{\mathbf{H}}$ after two-stage passing of hierarchical LSTM.

Finally, we add a classifier $\mathbf{C}$ (Logistic Regression or MLP) on top of $\mathbf{H}$ and conduct classification given input $h_{N_{day}}^{\mathbf{H}}$. During back-propagation, $\mathbf{C}$ passes its error to $\mathbf{H}$, and $\mathbf{H}$ uses BPTT to calculate its gradients and error and pass it to $\mathbf{B}$. $\mathbf{B}$ then also uses BPTT to calculate the gradients and perform updates of weights.

5.3.2 Attention Mechanism

Recently, attention-based model has been developed and applied to vision [41, 31, 52], natural language processing [1] and so on. The basic idea is to create a “context vector" to tell the model where to look at, and thus the model knows when to pay attention to particular part of the input.

In our model, since we have a two LSTM ($\mathbf{B}$ and $\mathbf{H}$), we can employ attention mechanism by telling $\mathbf{H}$ where to look at for the hidden representation obtained by $\mathbf{B}$. Instead of passing only the last hidden layer of $\mathbf{B}$ to $\mathbf{H}$, we also construct a context vector by taking the weighted sum of all hidden layers in $\mathbf{B}$. We hypothesis that through attention, the model can better understanding the behavior importance in each δ-hour in determining sickness for each day. Graphic representation of context vector can be seen in Fig 5.6.

Mathematically, we add a another neural network to learn the weights for context in $\mathbf{H}$. Suppose we are at time step t when processing through $\mathbf{H}$ and have input as $h_t^{\mathbf{B}} = [h_{t1}^{\mathbf{B}}, h_{t2}^{\mathbf{B}}, \ldots, h_{tN_{bin}}^{\mathbf{B}}]$ instead of only $h_{tN_{bin}}^{\mathbf{B}}$. Then, the context vector κ_t for this time step is

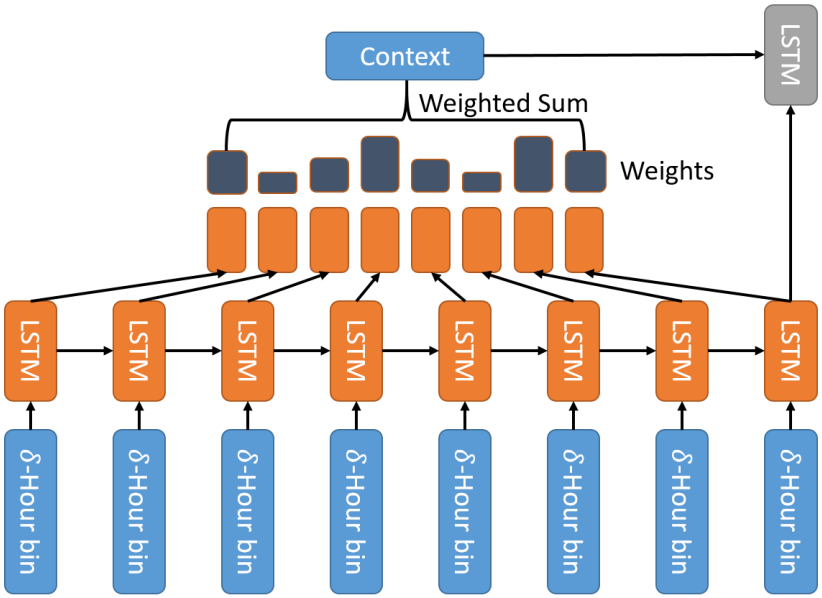


Figure 5.6: LSTM with Attention

calculated by:

$$\kappa_t = \sum_{j=1}^{N_{bin}} a_j h_{tj}^{\mathbf{B}} \tag{5.8}$$

$$a_j = \frac{e_j}{\sum_{i=1}^{N_{bin}} e_i} \tag{5.9}$$

$$e_j = v_a^T \tanh(W_a^T h_{tN_{bin}}^{\mathbf{B}} + U_a^T h_{tj}^{\mathbf{B}}) \tag{5.10}$$

where $v_a \in \mathbb{R}^k$, and $W_a, U_a \in \mathbb{R}^{k\times k}$, and we get raw weight value for each $h_{tj}^{\mathbf{B}}$ as a scalar e_j. Then we normalize e_j through softmax process showed in Equation 5.9 and get the final weights a_j, and we take the weighted sum of $h^{\mathbf{B}}$ as the context vector.

In order to include context vector into computation of H, we need to modify Equations 5.1 to 5.4 to the following:

$$i_t = \sigma(W_{in}^T x_t + U_{in}^T h_{t-1} + D_{in}^T \kappa_t + b_{in})$$
$$o_t = \sigma(W_{out}^T x_t + U_{out}^T h_{t-1} + D_{out}^T \kappa_t + b_{out})$$
$$f_t = \sigma(W_f^T x_t + U_f^T h_{t-1} + D_f^T \kappa_t + b_f)$$
$$g_t = \tanh(W_c^T x_t + U_c^T h_{t-1} + D_c^T \kappa_t + b_c)$$

where $D_{in}, D_{out}, D_f, D_c \in \mathbb{R}^{k\times k}$ and are parameters learned during training.

Chapter 6

Experiments

In this chapter, we will setup empirical experiments and provide analysis on models described in the previous chapter. More specifically, we will describe steps for preprocessing data for learning predictive model. We then will explain in details about the configuration of our model as well as comparison baseline model. We finally evaluate our model in different setting and analysis the results.

6.1 Preprocessing Steps

6.1.1 Data Imputation

Missing data is a major issue in our dataset. Many users do not have records in every binned hour segments. For each user, we thus replace the missing value with the median value of all previous values in the same bin for numerical metrics, and the most frequent value for categorical metrics.

6.1.2 One-Hot Encoding

For the categorical variables that we defined in Section 4.6, we use One-Hot encoding to transform the categories to numerical values. For example, the mode location variable could be [1, 2, -1] where 1 means top one location, 1 means top two location and -1 means all other location. Thus, we represent these three discrete value as [1, 0, 0], [0, 1,

0] and [0, 0, 1] separately instead of using 1, 2, -1.

6.1.3 Data Normalization

Each feature in each bin might have different meaning for different user. Thus, we decide to use a user-based normalization where for each user on each day, we normalize the features for all bins on that day with respect to the features for the same bins on previous days up to some limit (e.g previous 30 days). For a feature value x_{ijk} in bin j on day i indicating feature k, we find $x_{ijk}^{min} = \min_{d=0}^{30} x_{i-d,jk}$ and $x_{ijk}^{max} = \max_{d=0}^{30} x_{i-d,jk}$, then we do the transformation as following:

$$x_{ijk} = \frac{x_{ijk} - x_{ijk}^{min}}{x_{ijk}^{max} - x_{ijk}^{min}}$$

6.1.4 Data Filtering

To make our dataset better for analysis, we apply several rules for filtering data. First, we require each δ-hour bin to have some minimum amount of records since many of the metrics would be meaningless if there is only one record. Second, we require each day to have some minimum amount of bins. Third, for each N_{day} sequence, there need to some minimum amount of days. Additionally, due to our normalization step require previous data, we remove days which lacks enough data to scale.

Since there are much more healthy sequences then sick sequences, we only choose the sequences that are with 15 days of onset dates to train the model. After we have done the preprocessing and filtering, we obtain our final dataset with size showed in Table 6.1.

6.2 Model Configuration

6.2.1 Classifier

As we mentioned in 5.3.1, we need to add a classifier on top of the last hidden state of our HLSTM model so we can perform classification. We choose MLP over logistic

Table 6.1: Dataset size with respect to different bins and filtering rules. Min records is the minimum number of records required for each bin and min bins is the minimum number of bins required for each day.

δ	min records	min bins	# o f users	# of sequences	# of sick days
3	2	2	1,909	31,785	3,943
4	2	2	1,925	31,804	3,945
6	3	1	2,165	41,647	5,299
8	4	1	2,038	36,293	4,563
12	5	1	1,940	33,412	4,108
24	1	1	2,465	60,972	7,834

regression for the reason that MLP has another transformation of input layer to hidden layer which could extract latent features from its input.

6.2.2 Optimization

Mini-Batch Gradient Descent

Given the dataset $D = \{(x^i, y^i)\}_{i=1}^m$, in traditional batch learning, we pass the all training examples in D to perform parameters (θ) update given the gradient of loss $L(\theta)$ with respect to θ. On the other hand, stochastic gradient descent perform update after seen one training example. Here, we take an approach that is in between this two methods, where we update after seen b number of examples, where b can be any number between 1 and m. Given the learning rate is α and number of batches is $n = \frac{m}{b}$, an algorithm description of mini-batch gradient descent is showed in Algorithm 1 below.

Algorithm 1 Mini-Batch Gradient Descent

while L does not converge **do**
  for $i = 0$ to $n - 1$ **do** — loop over all mini-batches
    $L(\theta) = -\frac{1}{b}\sum_{j=i*b}^{(i+1)*b}$ CrossEntropy(x^j, y^j, h) — using5.7
    $\theta \leftarrow \theta - \alpha\frac{\partial L}{\partial \theta}$ — perform update

Adam

Optimizing with vanilla stochastic gradient descent can be slow to converge and trapped in local minimal. Instead, we choose to optimize our objective function using Adam [36], which is based on adaptive estimates of lower-order moments and appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. We set the hyper-parameters in Adam to default values as introduced in the original paper without further tuning.

6.2.3 Regularization

Deep neural networks are easily over-fitted with training data. Here we describe two popular methods in regularizing recurrent neural networks.

Weight Noise Adding a small Gaussian noise to weights of RNN is introduced in [29] to combat over-fitting problem. For each parameter in our HLSTM model, we add a Gaussian noise with a standard deviation of our choice during training, while we fixed the weights during validating and testing.

Dropout The idea for dropout [50] is simply to set some hidden units in the hidden layer of a neural network to zero by binomial distribution during each training iteration. By doing so, we are essentially training the model by sampling the features for hidden layer. Though dropout is originally applied to neural network, there are also attempts to apply dropout in RNN[53, 23]. We decide to apply the dropout techniques introduced in [23], where we apply dropout with the same binomial distribution at different time steps for both input feature vector and previous hidden layer.

6.2.4 Model Parameters

Complexity

The most complex model we use is Attention Hierarchical LSTM model, where the LSTM layers are all bidirectional. Thus, there are in total 4 different LSTM layers. Suppose the

hidden layer size for each LSTM is k. We will have 4 matrices with $N_f \times k$ parameters and 4 matrices with k^2 parameters and 4 bias vector with k parameters for each LSTM. For LSTM with attention, we have 6 more weight matrices with k^2 parameters and one vector with k parameters. For MLP binary classifier, suppose the hidden layer size is also k, then we have a weight matrix with $2k \times k$ parameters, another weight vector with k parameters and two bias with k and 1 parameters. In total, we will have number of parameters N_p as:

$$\begin{aligned} N_p =& 2(4N_f k + 4k^2 + 4k) + 2(4N_f k + 4k^2 + 6k^2 + 5k) + 2k^2 + 2k + 1 \\ =& 1 + (20 + 16N_f)k + 30k^2 \end{aligned}$$

Hyper-Parameters

Here we summarize all the hyper-parameters that we define rather than learned from training.

- Hidden layer size for LSTM (75) and MLP classifier (75)
- Number of layers for LSTM (1) and MLP classifier (1)
- Dropout rate for LSTM (0.2) and MLP classifier (0.1)
- Size of mini-batches (200)
- Initial learning rate (0.001)
- Standard deviation for weight noise (0.025)
- Number of epochs looping the training dataset (90)

6.3 Models for Comparison

For comparison purpose, we will evaluate our sequential data on several baseline models (including sequential and non-sequential) and our Attention Hierarchical LSTM model. After pre-processing and transforming the metrics, we end up with $N_f = 93$ features for each δ-hour bin.

6.3.1 Non-Sequential Models

We provide a list of non-sequential model which takes a single feature vector as input. Since our data is a sequence, we flatten the sequence into a feature vector with $N_{day} \times N_{bin} \times N_f$ dimensions.

Logistic Regression and MLP We follow the description in Section 5.1.1 for these two models. Both are implemented with Theano [4, 2] and trained with Adam. For MLP, we use one hidden layer with 500 neurons, and use leaky-ReLU [40] as activation function. We also apply dropout with probability of 0.5 for regularization.

Random Forest Random forest [7] is a ensemble method where it drops multiple bootstrapped sample from the training data and train a decision tree on each sample. We use the implementation provided by scikit-learn [45, 9] and set the number of trees to be 1000.

Support Vector Machine Support vector machine (SVM) [16] is a powerful linear classification algorithm. Along with kernel tricks [47], we can also determine a non-linear decision boundary. We use the implementation from scikit-learn and LIBSVM [13], and use a RBF kernel defined as $K(x, x') = \exp(-\gamma||x - x'||)$ with $\gamma = \frac{1}{N_{day} \times N_{bin} \times N_f}$.

6.3.2 Sequential Models

All sequential models are taking a sequence as input, and are implemented by Theano and optimized using Adam.

Long Short-Term Memory To use plain LSTM, we need to transform our sequence of days of sequence of bins into long sequence of $N_{day} \times N_{bin}$ bin vectors. We apply dropout described in 6.2.3 with probability of 0.2.

Attention Hierarchical LSTM Hierarchical LSTM model can take a three-way tensor as input and we also compare the results for models with/without attention mechanism. The hyper-parameters are as described in 6.2.4.

6.4 Results and Analysis

6.4.1 Receiver Operating Characteristic

Since our models all output probabilities and then use threshold to determine which class the input should be, it is natural for us to evaluate our model using Receiver Operating Characteristic (ROC) Curve. ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, in our case, various values in [0, 1].

ROC curves has true positive rate on the Y-axis and false positive rate on the X-axis. This means that the top left corner of the plot is the "ideal" point - a false positive rate of zero, and a true positive rate of one. It also indicates that larger area under the curve (AUC) is usually better, while a random classifier would give have a AUC score of 0.5.

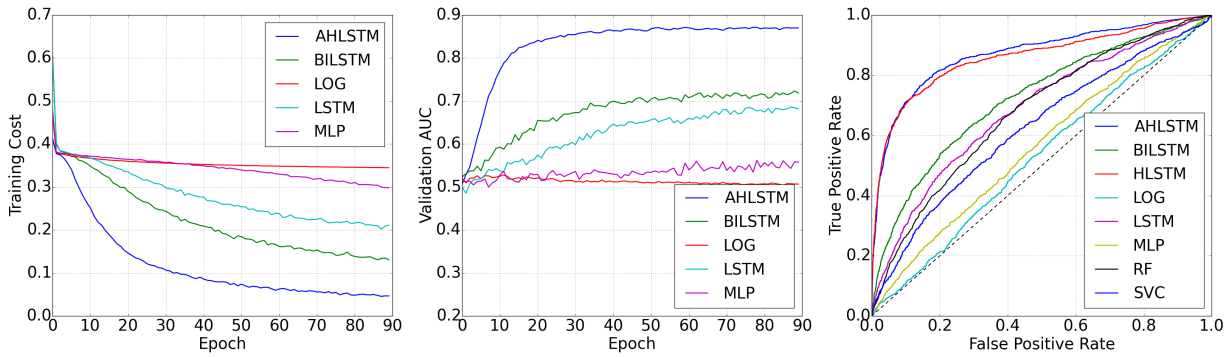
6.4.2 Analysis and Discussion

We first compare our Attention HLSTM model to other models using data binned in 8-hour. For the reason that this is still an ongoing research, we only report the evaluation on validation set. We plot the ROC curve for all models in Figure 6.1.

As we can see from the ROC curve and the table, Attention HLSTM model achieves the best AUC score. Without attention mechanism, HLSTM still outperforms other models. Sequential models such as Bidirectional LSTM and LSTM also get higher AUC scores than non-sequential model such as SVM, MLP and logistic regression. Random forest has similar performance to LSTM, which we believe bootstrapping helped random forest to remove variance in the data. SVM, as a non-linear model, outperforms MLP and logistic regression.

For the models trained with gradient descent, we can observe that Attention HLSTM model converges fastest compare to others, while also achieves the highest AUC score. On the other hand, logistic regression converges slowest, and its performance is nearly random even after many epochs of training.

Figure 6.1: Training Cost v.s. Epoch(left), AUC Score v.s. Epoch (middle) and ROC Curve(right) for all models trained and evaluated with data binned in 8-hours. **AHLSTM** stands for Attention Hierarchical LSTM model, **BILSTM** stands for Bidirectional LSTM model, **LOG** is logistic regression, **RF** is random forest and **SVC** is support vector machine.



Model	**AHLSTM**	**HLSTM**	**BILSTM**	**LSTM**	**MLP**	**LOG**	**RF**	**SVC**
AUC	**0.871**	0.859	0.723	0.685	0.561	0.527	0.676	0.623

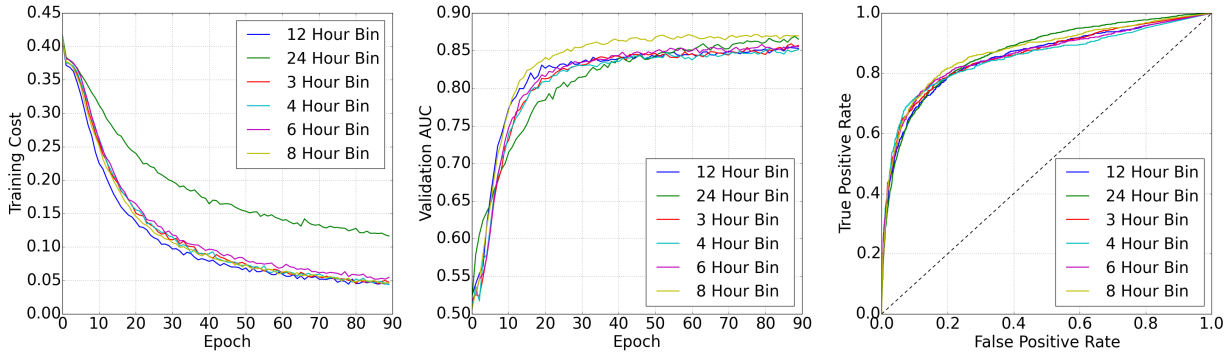
Table 6.2: AUC scores for above ROC curves

Does Different δ Help?

We also examine the performance of our Attention HLSTM model on data binned in different δ-hour segments. We plot the ROC curve as well as the training cost and AUC score over epochs in Figure 6.2. Since we have different filtering rules for different bins, theses datasets are different as well and thus directly comparing the score might be misleading. However, from the ROC curves and convergence of training cost, we do not observe much difference in applying different bins.

Notice that $\delta = 24$ is a special case for applying Attention HLSTM model since there is not recurrent relation between bins, thus the input is first transformed as a normal feed-forward neural network and then passed to the higher level LSTM. Also, the training cost for 24-hour bin is much higher while the AUC score on validation set is reasonably good. This might be due to the size of training data is larger, which is generally good for training a deep and complex neural network.

Figure 6.2: Training Cost v.s. Epoch(left), AUC Score v.s. Epoch (middle) and ROC Curve(right) with Attention HLSTM model on validation set using different δ value.



δ	3	4	6	8	12	24
AUC	0.858	0.852	0.857	**0.871**	0.854	0.868

Table 6.3: AUC scores for above ROC curves

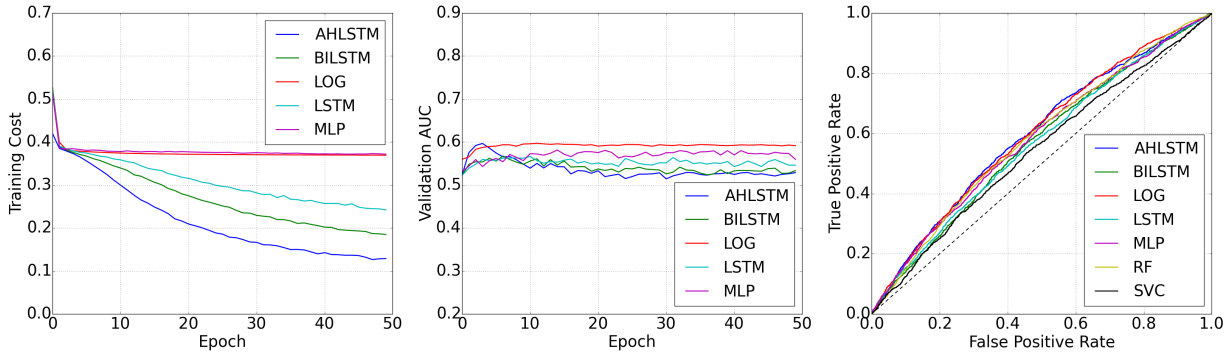
Generalization on New User?

So far we gather the sequential data for all users and then split them into training, validation and testing set. However, this might cause an over-fitting issue and low generalizability of our model, since for the same user there are data in both training and evaluation phase. Thus, we also apply a user-level splitting rule, where we split the users into training users, validation users and testing users. We use the data binned with 24-hour in this set of experiment for computational efficiency purpose. After splitting, we have 1,479 users for training, 493 users for validation and 493 users for testing. Similar to previous experiments, we plot the ROC curve training cost and validation AUC score over epoch, showed in Figure 6.3.

As showed in these plots, none of these models give comparable results to previous splitting rules. We no longer see that sequential models outperform non-sequential ones. Also, simple model such as logistical regression can achieve same results as our Attention HLSTM model, indicating an issue either in preprocessing steps or over-fitting problem using previous splitting rules.

In addition, we can see from the training cost plot that the training cost using At-

Figure 6.3: Training Cost v.s. Epoch(left), AUC Score v.s. Epoch (middle) and ROC Curve(right) with all models trained and evaluated on data binned with 24-hour



Model	**AHLSTM**	**BILSTM**	**LSTM**	**MLP**	**LOG**	**RF**	**SVC**
AUC	**0.596**	0.567	0.566	0.581	**0.596**	0.584	0.544

Table 6.4: AUC scores for above ROC curves

tention HLSTM model is converging, however the validation AUC score almost remains unchanged the whole time. Solving this pitfall in generalization on new users would be our next research focus.

Chapter 7

Conclusion

7.1 Summary of Work

In this work, we present an framework of applying deep learning on CDR data to predict users' sickness. We first show number of metrics extracted from CDR data that could effectively describe the changes in patients' behavior during they are sick. We visualize these metrics and utilize them for prediction. We build a specific deep recurrent neural network model to predict if a user is sick or not given a sequence of CDR data. We show that our model outperform other machine learning algorithms.

However, there are many limitations in this work. First, the hyper-parameters for our deep learning models are found by random search, which might not give the best results. Second, deep learning suffers from interpretability and so does our model. We could not tell what features are helpful and what happens in the network. Third, our model has low generalizability on new users' data, so does other less complex machine learning models. This might indicate that we need extract work on feature extraction and data pre-processing steps. Finally, we only train and evaluate our model a set of patients, ignoring the larger amount of data from healthy population.

7.2 Future Directions

In the next steps, we hope to fix the issue of our model's generalization on new users' data. There are works on understanding and visualizing RNN [35], we will focus on explaining the results we are getting from our deep learning model. We might also try to build new models that are not individual-level. Also, instead of using only patients' data, we will try to include data from healthy population either as new features or training data. Finally, we wish to evaluate our model on whole population over other time period which has a disease outbreak and to see if our model has the ability to detect the outbreak given unseen data.

Bibliography

[1] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[4] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[5] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[6] Andrey Bogomolov, Bruno Lepri, Jacopo Staiano, Nuria Oliver, Fabio Pianesi, and Alex Pentland. Once upon a crime: towards crime prediction from demographics and mobile data. In *Proceedings of the 16th international conference on multimodal interaction*, pages 427–434. ACM, 2014.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[9] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[10] ITU Telecommunication Development Bureau. Ict facts and figures – the world in 2015, 2015. URL `http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx`.

[11] Fabrice Carrat, Elisabeta Vergu, Neil M Ferguson, Magali Lemaitre, Simon Cauchemez, Steve Leach, and Alain-Jacques Valleron. Time lines of infection and disease in human influenza: a review of volunteer challenge studies. *American journal of epidemiology*, 167(7):775–785, 2008.

[12] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées.

[13] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[14] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[15] Ben S Cooper, Richard J Pitman, W John Edmunds, and Nigel J Gay. Delaying the international spread of pandemic influenza. *PLoS Med*, 3(6):e212, 2006.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic, and Alex Sandy Pentland. Predicting personality using novel mobile phone-based metrics. In *Social computing, behavioral-cultural modeling and prediction*, pages 48–55. Springer, 2013.

[18] Katayoun Farrahi and Daniel Gatica-Perez. What did you do today?: discovering daily routines from large-scale mobile data. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 849–852. ACM, 2008.

[19] Katayoun Farrahi and Daniel Gatica-Perez. Discovering routines from large-scale human locations using probabilistic topic models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1):3, 2011.

[20] Bjarke Felbo, Pål Sundsøy, Alex 'Sandy' Pentland, Sune Lehmann, and Yves-Alexandre de Montjoye. Using deep learning to predict demographics from mobile phone metadata, 2015.

[21] Douglas M Fleming and Alex J Elliot. Health benefits, risks, and cost-effectiveness of influenza vaccination in children. *The Pediatric infectious disease journal*, 27 (11):S154–S158, 2008.

[22] Vanessa Frias-Martinez, Enrique Frias-Martinez, and Nuria Oliver. A gender-centric analysis of calling behavior in a developing economy using call detail records. In *AAAI Spring Symposium Series*, 2010. URL `https://www.aaai.org/ocs/index.php/SSS/SSS10/paper/view/1094/1347`.

[23] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. 2015.

[24] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[25] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[26] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[27] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[28] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[29] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[30] Alex Graves. *Supervised sequence labelling*. Springer, 2012.

[31] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Dennis KM Ip, Lincoln LH Lau, Kwok-Hung Chan, Vicky J Fang, Gabriel M Leung, JS Malik Peiris, and Benjamin J Cowling. The dynamic relationship between clinical symptomatology and viral shedding in naturally acquired seasonal and pandemic influenza virus infections. *Clinical Infectious Diseases*, page civ909, 2015.

[35] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[36] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Lincoln LH Lau, Benjamin J Cowling, Vicky J Fang, Kwok-Hung Chan, Eric HY Lau, Marc Lipsitch, Calvin KY Cheng, Peter M Houck, Timothy M Uyeki, JS Malik Peiris, et al. Viral shedding and clinical illness in naturally acquired influenza virus infections. *Journal of Infectious Diseases*, 201(10):1509–1516, 2010.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[39] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[40] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Citeseer, 2013.

[41] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[42] World Health Organization. Influenza fact sheet, 2003. URL `http://www.who.int/mediacentre/factsheets/2003/fs211/en/`.

[43] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318, 2013. URL `http://jmlr.org/proceedings/papers/v28/pascanu13.html`.

[44] Michael J Paul and Mark Dredze. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272, 2011.

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

[46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[47] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[48] Alessio Signorini, Alberto Maria Segre, and Philip M Polgreen. The use of twitter to track levels of disease activity and public concern in the us during the influenza a h1n1 pandemic. *PloS one*, 6(5):e19467, 2011.

[49] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[51] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[52] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057, 2015.

[53] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Appendix A

We plot the behavior metrics according to the description in Section 4.7.1. All the plots have the same x-axes, which is the days to onset. Y-axes are different and their units are showed in 4.1.

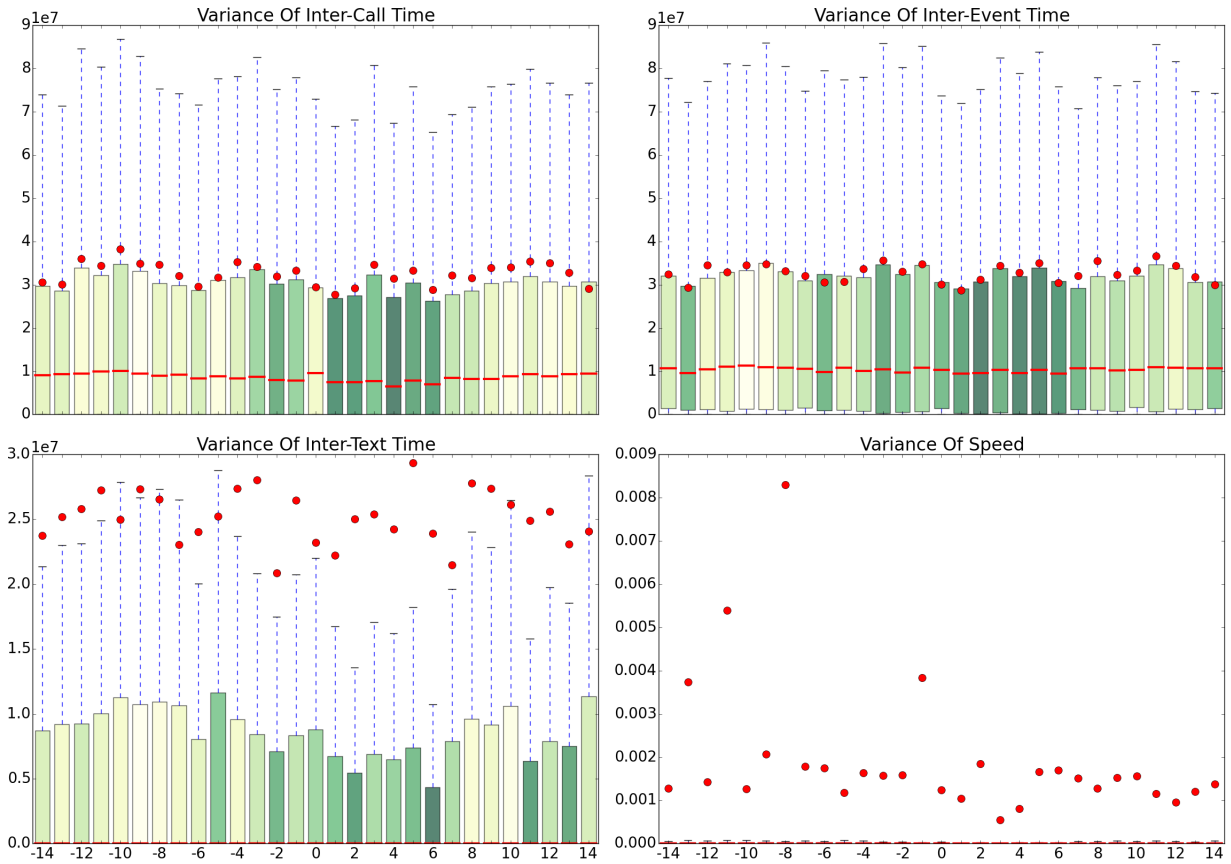


Figure A.1: Plots for variances of inter-event (call, text, all) time and speeds

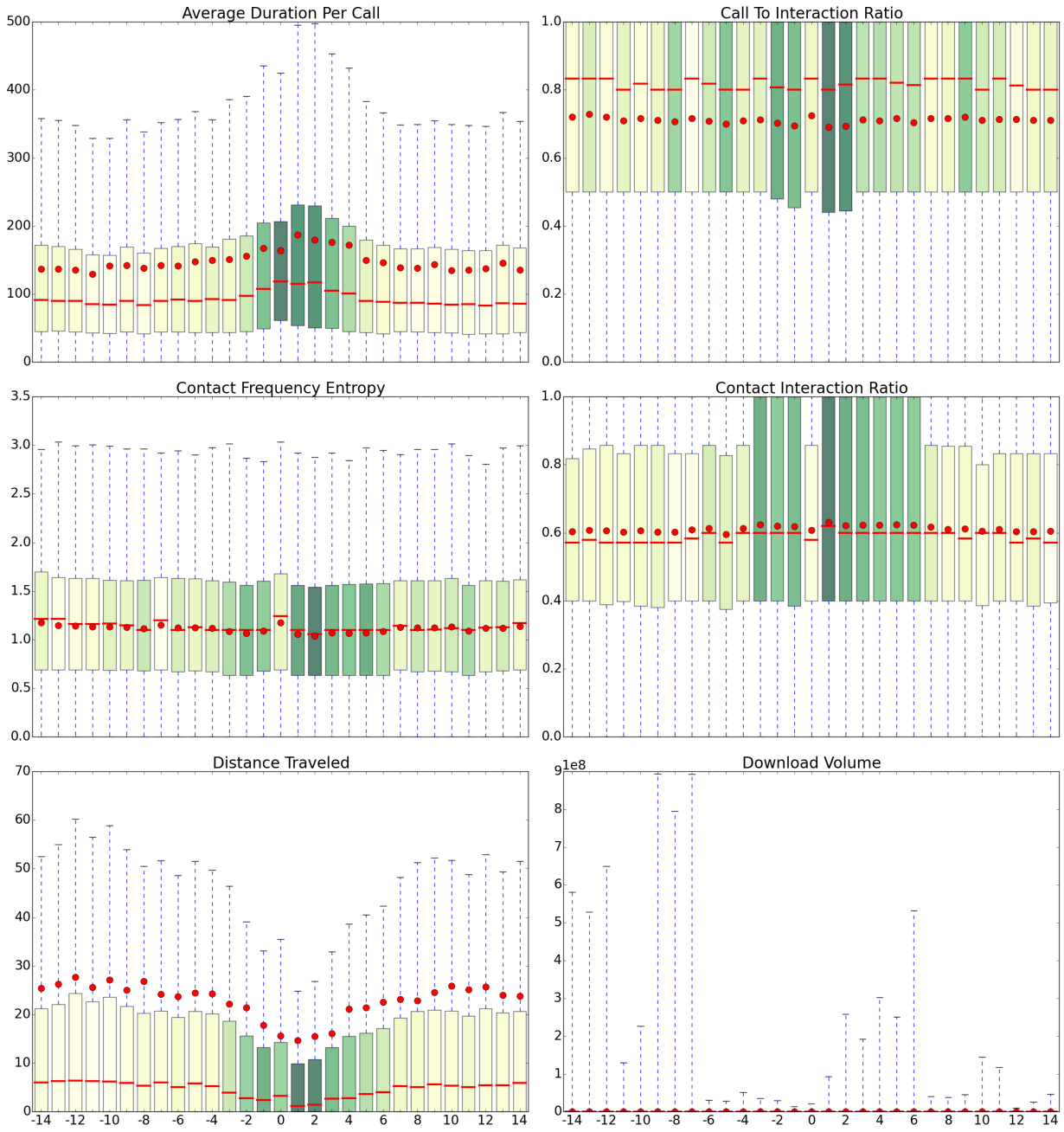


Figure A.2: Plots for average duration per call, call to interaction ratio, contact entropy, contact to interaction ratio, distance traveled and download volume

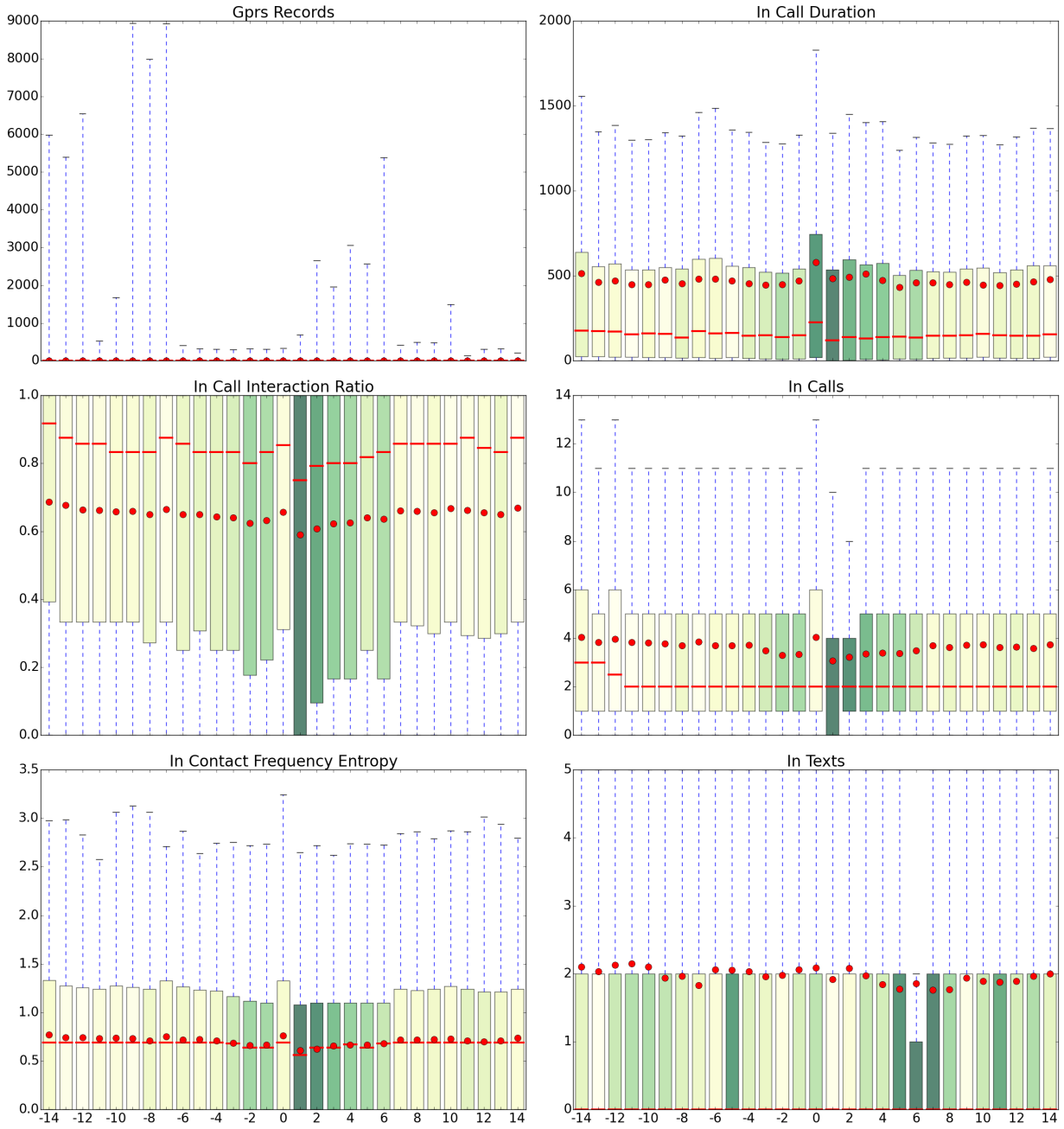


Figure A.3: Plots for number of GPRS records, total incoming call duration, incoming call to interaction ratio, number of incoming calls, incoming contact entropy and number of incoming texts

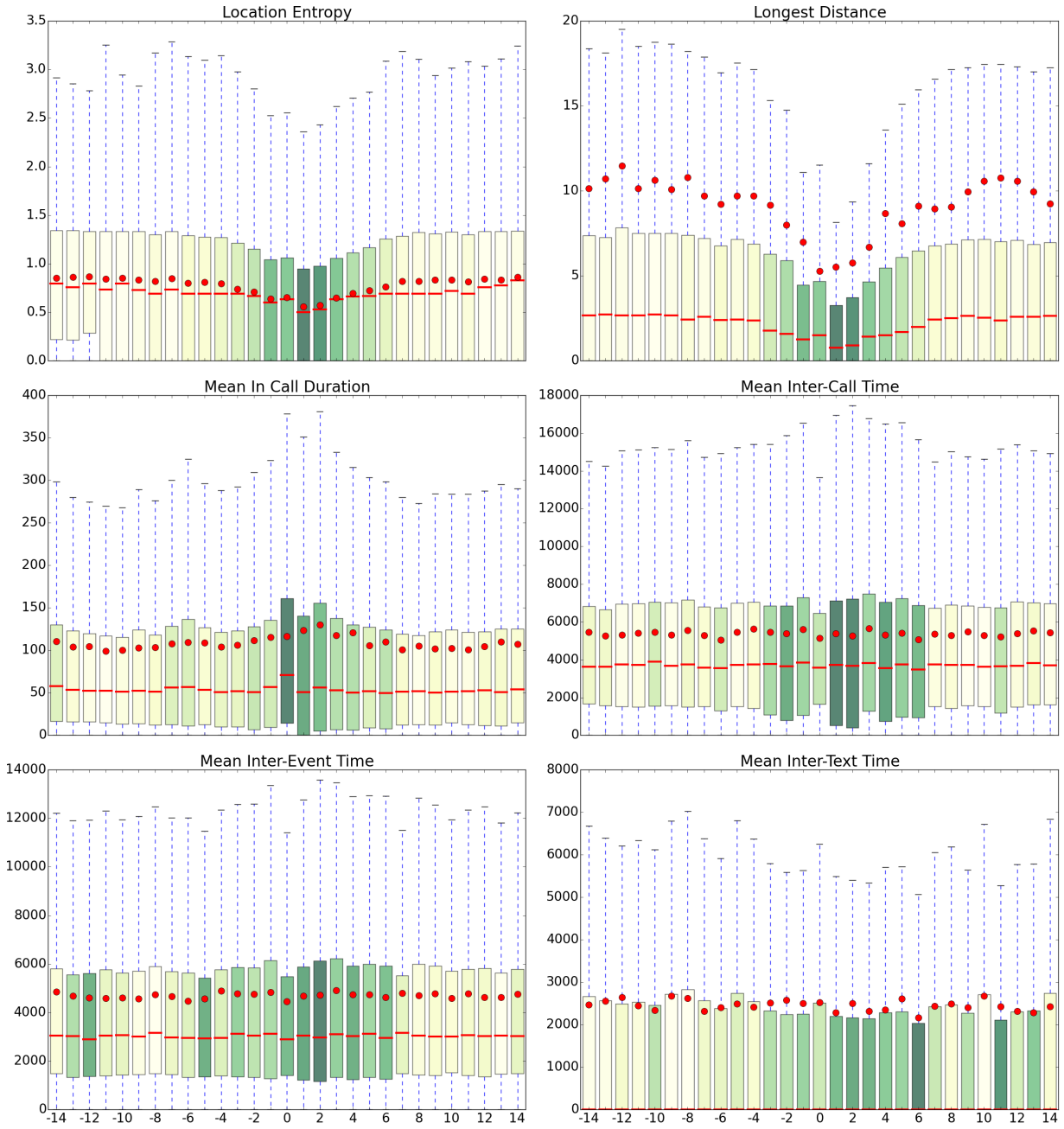


Figure A.4: Plots for location entropy, longest distance traveled, average incoming call duration, average inter-event(call, text, all) time

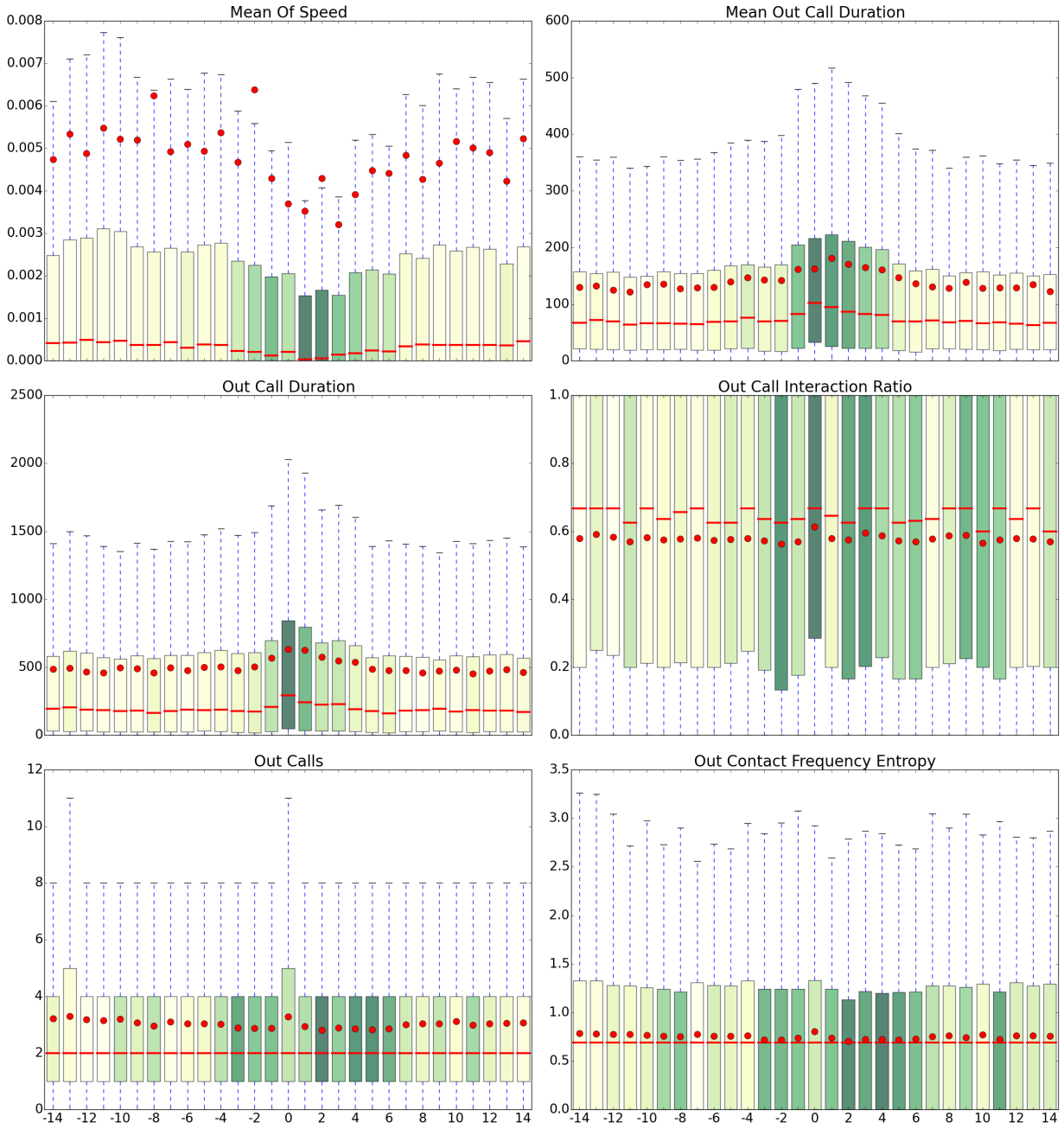


Figure A.5: Plots for average speed, average outgoing call duration, total outgoing call duration, outgoing call to interaction ratio, number of outgoing call and outgoing contact entropy

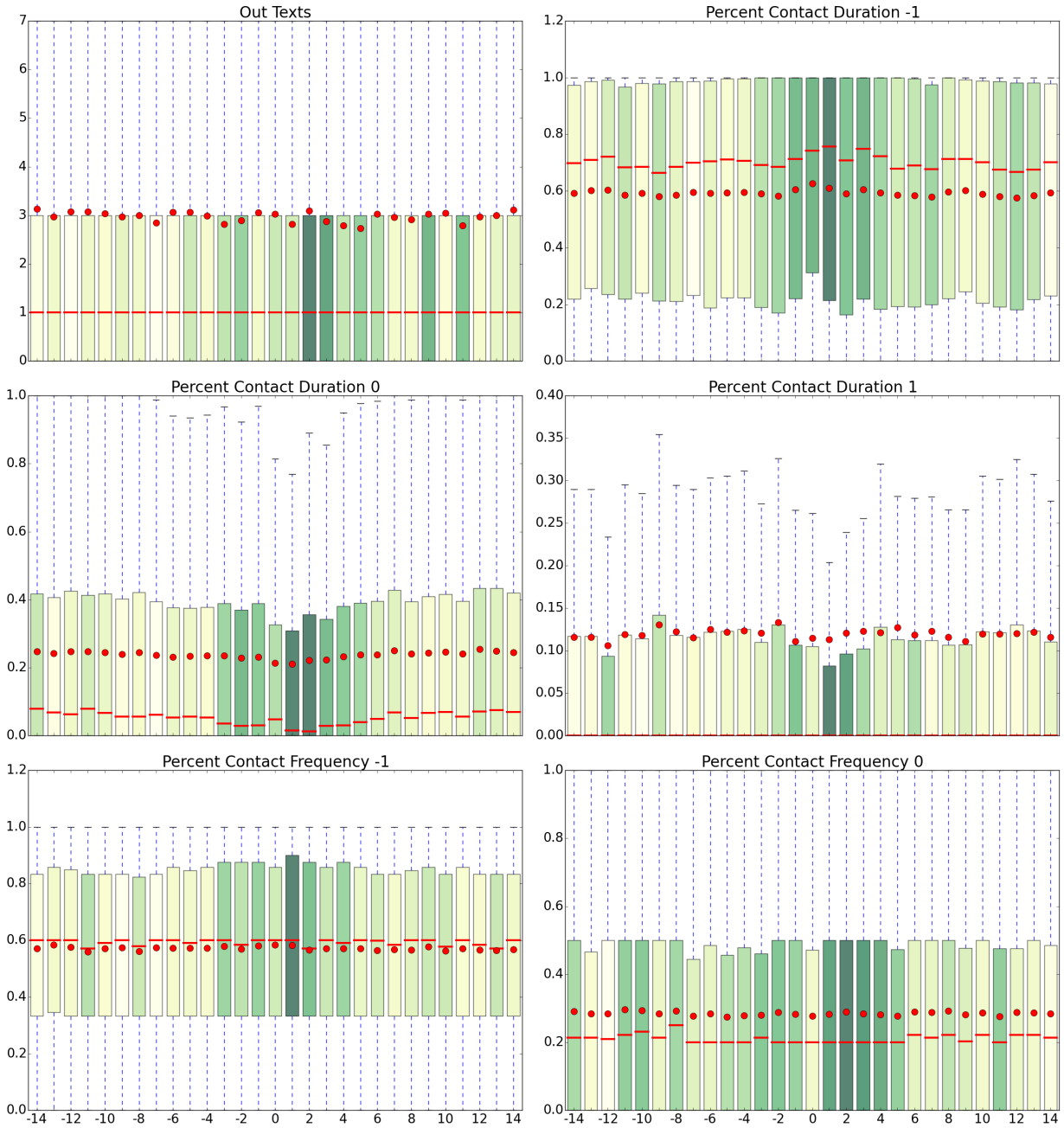


Figure A.6: Plot for number of outgoing text, percent duration talked to strangers(-1), percent duration talked to top contact(0), percent duration talked to second top contact(1), percent interacted with strangers(-1), percent interacted with top contact(0)

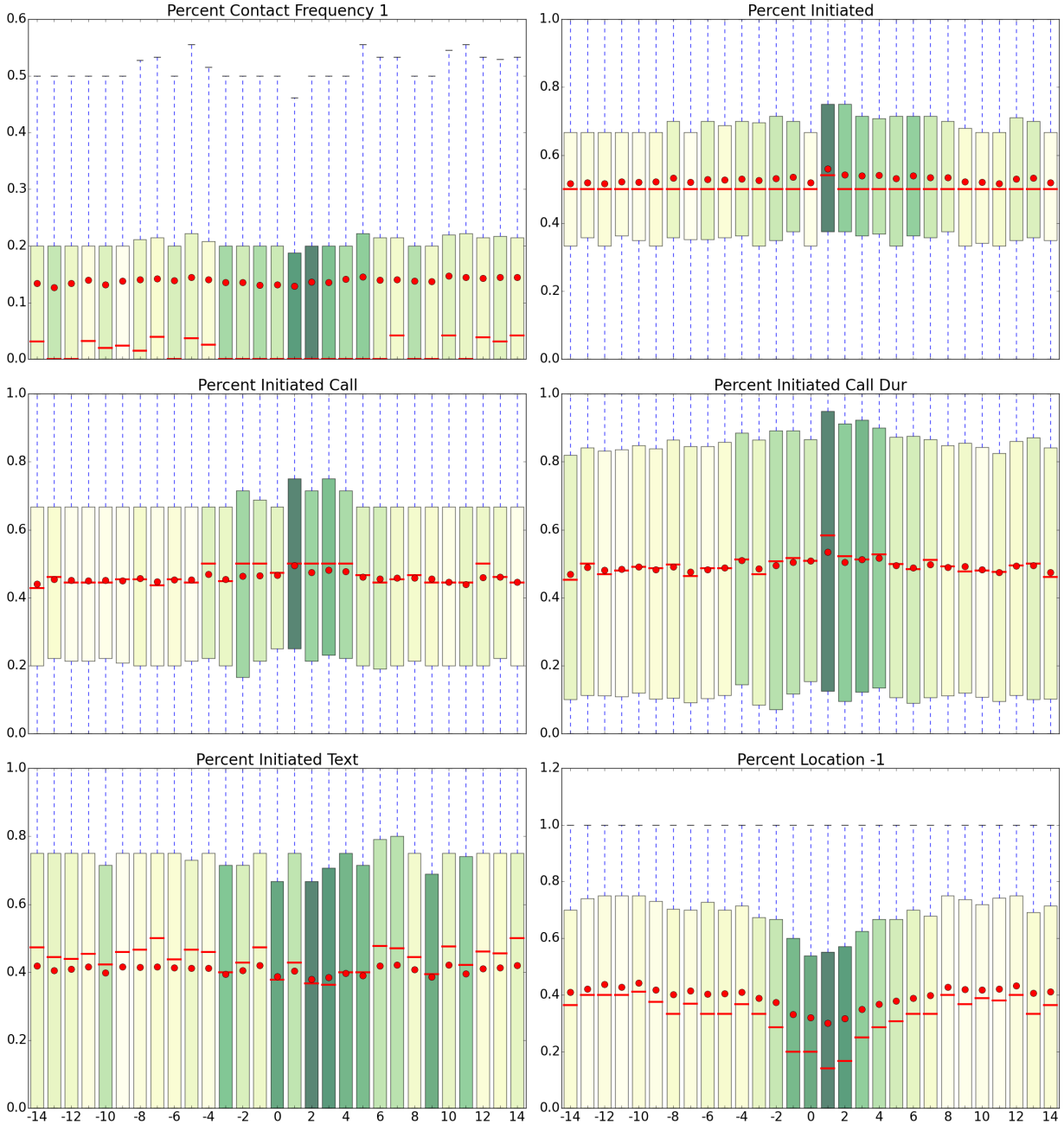


Figure A.7: Plots for percent interacted with second top contact(1), percent of initiated events(call, call duration, text, all), percent visiting strange location (-1)

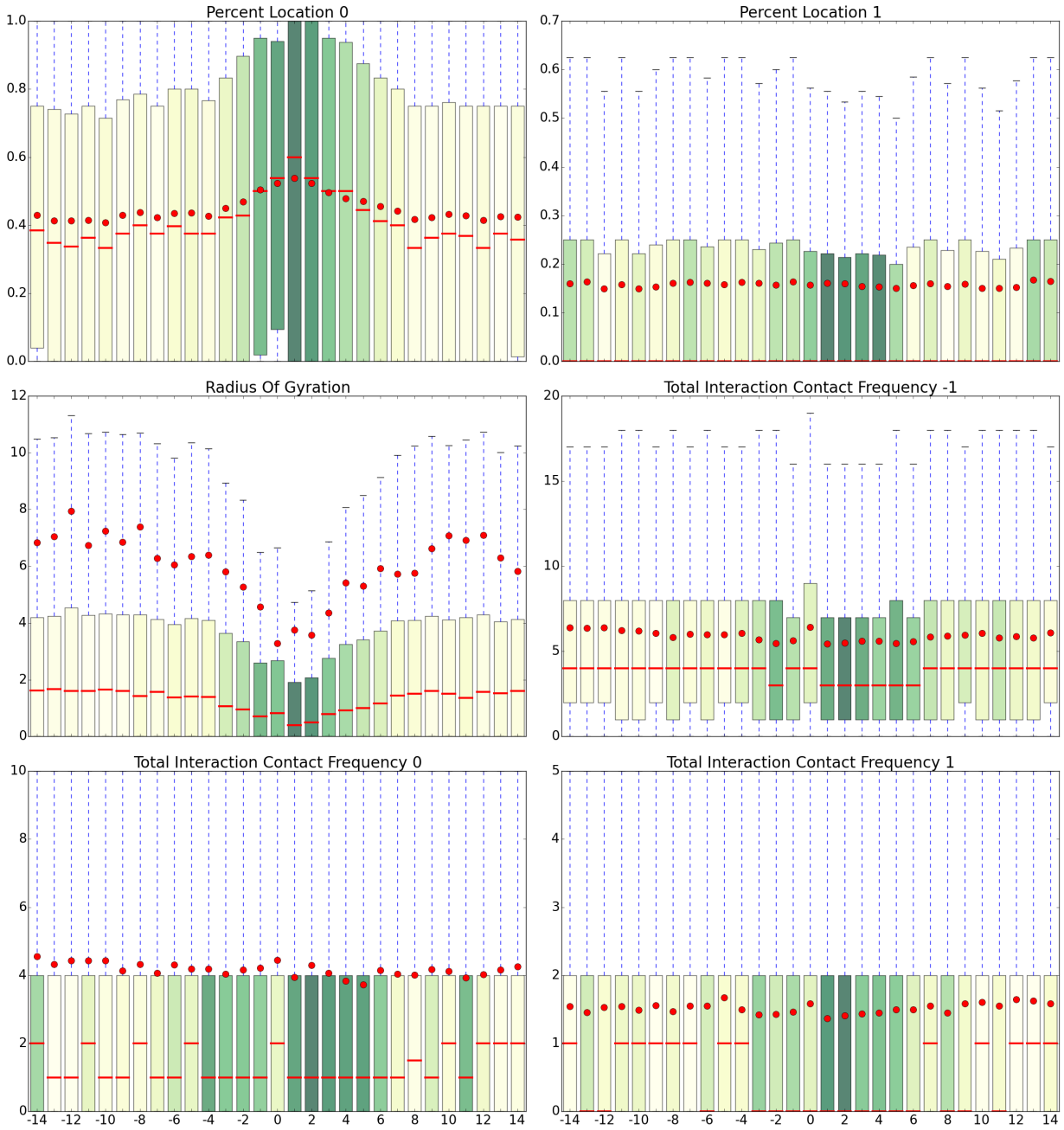


Figure A.8: Plots for percent visiting top location (0), percent visiting second top location (1), radius of gyration, total count of interactions with stranger(-1), top contact(0) and second top contact(1)

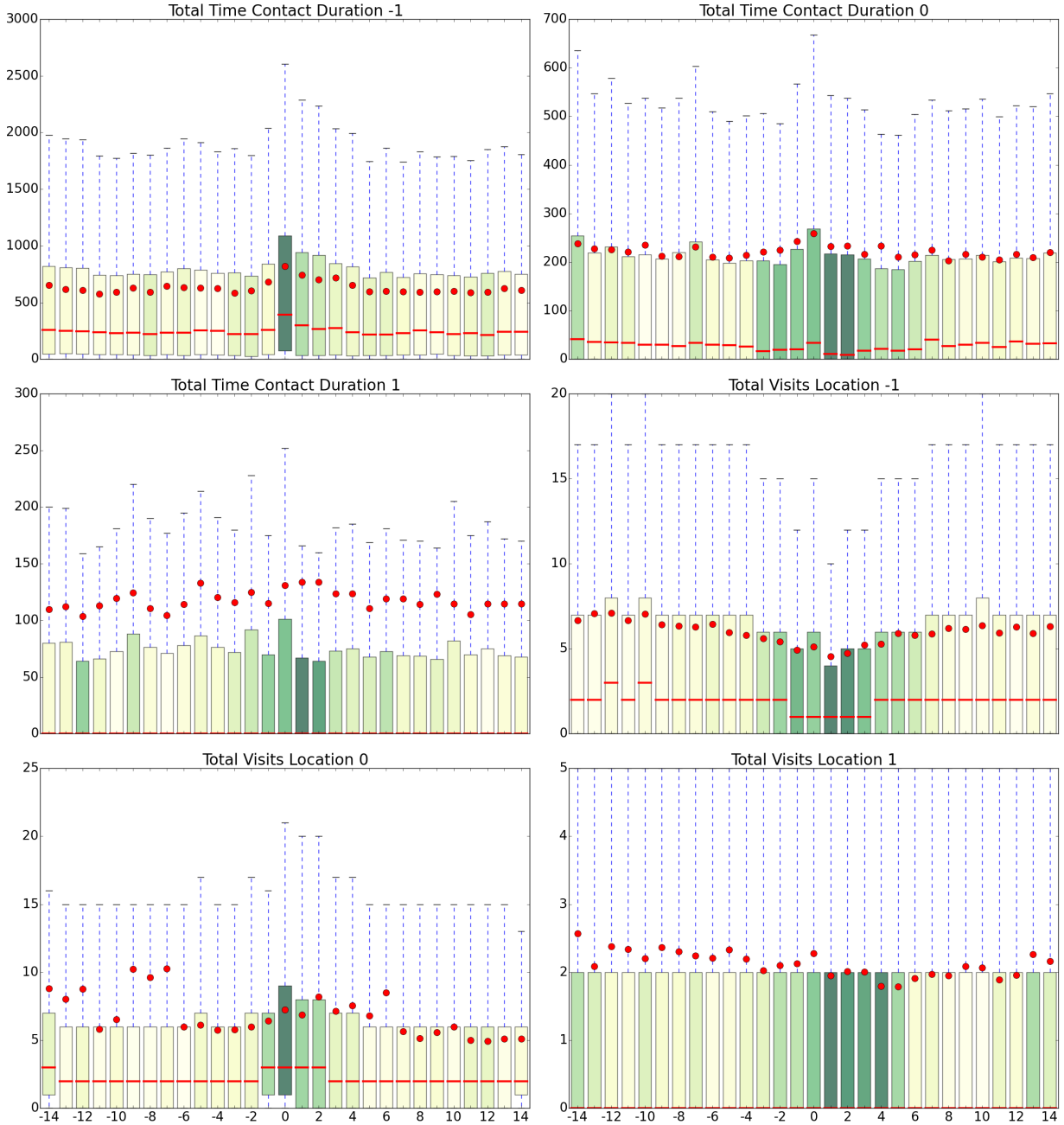


Figure A.9: Plots for total duration talked to strangers(-1), top contact(0), second top contact(1), total visits of strange location(0), top location(0) and second top location(1)

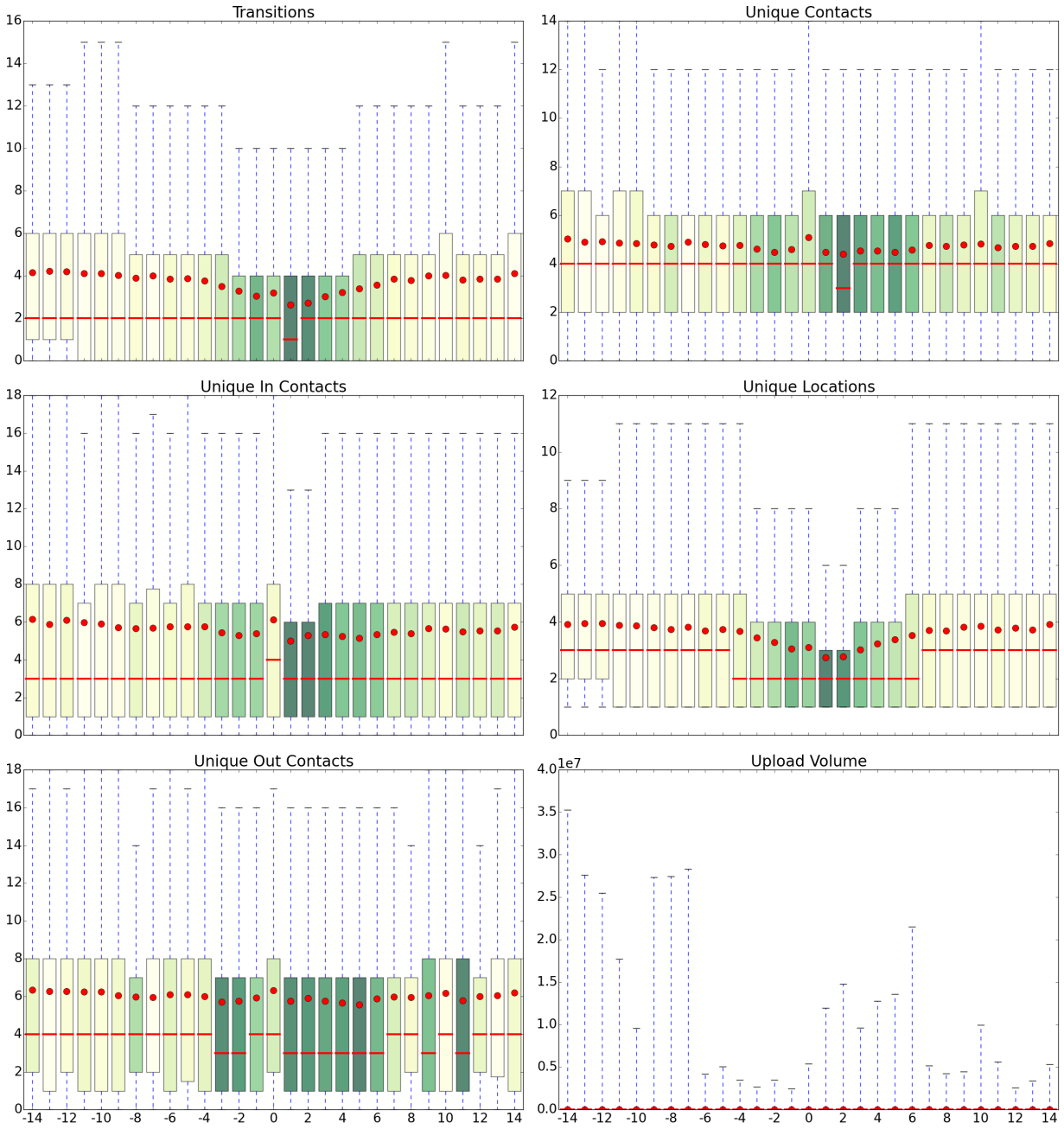


Figure A.10: Plots for number of transitions, number of unique contacts (incoming, outgoing, all), number of unique location visited and upload volume