4-23-2018 1:15 PM

A Framework for Modelling User Activity Preferences

Roberto Barboza Junior, *The University of Western Ontario*

Supervisor: Miriam A.M. Capretz, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

The availability of location data increases every day and brings the opportunity to mine these data and extract valuable knowledge about human behaviour. More specifically, these data may contain information about users' activities, which can enable, for example, services to improve advertising campaigns or enhance the user experience of a mobile application. However, several techniques ignore the fact that users' context other than location and time, such as weather conditions, influences their behaviour. Moreover, several studies focus only on a single data source, addressing either data collected without any type of user interaction, such as GPS data, or data spontaneously shared by the user, for instance, from *location-based social networks* (LBSNs), but not both.

This thesis proposes a framework that aims to predict *users' current activity preferences* (UCAP). UCAP handles data gathered from different sources. It takes into account users' historical data, their current context, and other external contexts such as weather conditions.

The framework was evaluated on five real-world datasets. The results demonstrated the accuracy of the proposed solution, which was on average 12.3% more accurate than a state-of-the-art technique. Moreover, the experiments evaluated the impact of the main components on the prediction results and showed that UCAP is not constrained by dataset size.

Keywords: Location-Based Services, Location-Based Social Networks, Location-Data Preprocessing, Context Aware, User Behaviour Prediction, Location-Based User Clustering

Acknowlegements

First, I would like to thank my supervisor Dr. Miriam Capretz. Dr. Capretz believed in my potential and gave me the opportunity to study under her guidance. Without her support, this thesis would not become a reality. Thank you for your patience, time and effort to make this whole experience possible.

I will never truly be able to express how thankful I am to my father, Roberto Barboza, and my mother, Cecilia Leite de Araujo Barboza, for being there whenever I needed it, and even when I thought I did not need it. Thank you for helping me find my own way. I would never have made it this far without you. I love you both. I would also like to extend my heartfelt appreciation to my brother Vinicius, who is always willing to talk with me about anything and supporting me no matter what. I am blessed to be a part of this family.

I would not be here without the support of my wife, Andrea Okuda, who left everything behind to join me in this challenge. You have inspired me to strive to be the best version of myself. Thank you for being with me. Love you.

My sincere thanks must also go to my former and current laboratory colleagues and friends: Wander Queiroz, Alex L'Heureux, Will Aguiar, José Miguel Alves, Rafael Aguiar, Daniel Berhane Araya, Dr. Wilson Higashino, Dr. Katarina Grolinger and Dr. Ahmed Eltahawi, who helped throughout this research. Special thanks to Dr. Hany ElYamany and Dr. Mahmoud ElGayyar for lending me their expertise to my research, spending countless hours reviewing my work, and for believing in my work even when I had questions about it. Thank you.

I cannot forget my old-time friends, Mauro Ribeiro and Dennis Bachmann, who did everything in their power to assist me since I decided to come to Canada. Thanks for being nearby and for all the discussions about life, the universe and everything.

Last but not least, thanks to my family, my friends, and all my colleagues who accompanied me during life, and for all of those that will never forgive me for not mentioning them here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Information about an individual's location can tell a lot about that person. For example, it can indicate what they like, what they are doing or what they are going to do. Using location information has always been part of human history. Over the years, it evolved from maps carved in stone to fully functional navigation systems that fit into our pockets. Some technological breakthroughs were essential to make these innovations a reality. First, the emergence of navigation systems such as the GPS[1] as stand-alone devices enabled applications that were focussed on finding a route to a location and also finding points of interest (POI) in an area. Later, GPS modules started to be embedded in other electronic devices (e.g., tablets and smartphones). These electronic devices coupled with the popularization of mobile Internet stimulated the development of a number of innovative applications. These new applications can gather personal location information either actively or passively. In active data gathering, users spontaneously share their location, for example through a check-in. In contrast, passive data collection gathers location data without user interaction.

In Location-Based Social Networks (LBSNs) [1] such as Facebook and Foursquare, users

[1]GPS stands for Global Positioning System and is a common embedded navigation system found in mobile devices. The term GPS will be used as a metonym for this type of system.

can share their current location with their friends by checking in to a certain POI. The check-in generates an event. An event includes a user identifier, a specific POI (e.g., a *Pizzeria*), and a timestamp. From these data, a current activity can be inferred [2, 3, 4], such as *eating pizza*. Aggregating users' check-ins can provide a better understanding of their spatial-temporal activity preferences. For example, an advertiser might have a campaign targeting people willing to eat pizza. By knowing that a user is interested in eating a pizza at a particular moment, the advertiser has more confidence that the campaign could convert into a sale. This understanding would also enhance the user experience because the advertisement would be relevant to the user.

Other applications depend on background services to capture user location without any type of interaction other than the user's permission. The collected location data can help enhance application usability. For instance, suppose that a user stores a shopping list in a mobile application. The application could detect that the user is near a grocery store, check whether any product on the list is on sale and push a notification to the mobile device.

Prior work using location data as input to prediction models uses either LBSN check-ins [5, 6] or GPS information [7, 8]. Each of these types of data sources presents its issues, and using the same approach for both types is even more challenging and has not been tried.

When the data come from user input, such as check-ins, duplicates may be generated [9]. For example, the user may click quickly on the check-in button multiple times, recording several events with the same user, the same timestamp, and the same location.

On the other hand, when the data source relies on GPS information, the data contain uncertainty[10]. The location information gathered from GPS devices is composed of the device's coordinates (latitude and longitude) and the precision of these coordinates. For instance, Fig. 1.1 depicts a hypothetical scenario where the circle represents the GPS coordinates and their precision. Note that the user could be anywhere within the circle radius. The user can be reported being either in the bank or the grocery store. The background process could record events with the same user and the same timestamp, but different locations.

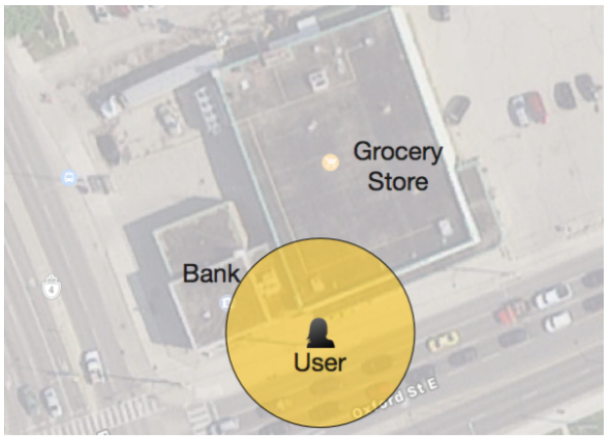

Figure 1.1: A scenario showing a user and two different locations.

Moreover, the context surrounding an individual is usually restricted to the location, the timestamp at which an activity happens, and the time between activities. It does not take into consideration events such as severe weather conditions, which are not unusual, or even more special occasions such as the Olympic Games. These external contexts can influence human behaviour in the location they occur and should be incorporated into the models.

1.2 Contribution

The main contribution of this thesis is a framework to predict **u**sers' **c**urrent **a**ctivity **p**references (UCAP). The prediction is based on users historical event patterns, their current context, and other external contexts such as weather conditions. This thesis explains all the framework components in detail, including their functions and relationships. The framework components accommodate a set of novel ideas that can be considered as additional research contributions.

First, the UCAP framework can handle location data collected from LBSN check-ins or background GPS services. It solves the *duplicated entries* and *GPS uncertainty* issues. Second, UCAP integrates external contexts such as weather conditions and special occasions into its prediction model.

Finally, UCAP uses clustering techniques to group users with similar behaviour to enhance

individual predictions. Intuitively, a cluster has more data than an individual. Accordingly, a cluster-based prediction model will perform better than one trained with individual data.

The proposed framework was evaluated in three experiments using five real-world datasets: two LBSN check-in datasets and three datasets gathered through GPS modules. The first experiment investigated the impact of the main UCAP components on the prediction results. The second experiment compared UCAP with a state-of-the-art approach. It is important to underline that the same datasets and scenario were used to make the comparison valid, not to undermine the results of the other approach. The third experiment evaluated the performance of UCAP on distinct datasets that had different data sources and sizes.

The results obtained highlight the following observations:

- The main UCAP components have a direct impact on prediction results.
- UCAP outperforms a state-of-the-art approach under the evaluated metric.
- UCAP is not constrained by dataset size.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 provides background information that is useful in understanding this work as well as a literature review of related studies. This chapter first provides an introduction to the technical terms and concepts that are used throughout this thesis. Second, the chapter presents a review of current studies that attempt to model user behaviour based on location data. Finally, it contrasts the contribution of this thesis with existing work.
- Chapter 3 discusses the layers of the UCAP framework. The UCAP framework consists of four layers: the *Preprocessing Layer*, the *Feature Engineering Layer*, the *Prediction Layer*, and the *Presentation Layer*. This chapter illustrates the components of each layer.

In addition, this chapter describes the relationships and interactions among these components.

- Chapter 4 presents an evaluation of the *UCAP* framework. It starts with a description of the evaluation environment. Next, the characteristics of the datasets used are presented, analyzed, and discussed. Then the chapter presents three experiments that evaluated the UCAP framework. Finally, a discussion of the experiemental results is provided.

- Chapter 5 presents the conclusions of this work, as well as a discussion of areas for future work involving the *UCAP* framework.

Chapter 2

Background and Literature Review

The objective of this chapter is two-fold: first, it introduces the background terminology and concepts related to the topics discussed in this thesis; second, it gives an overview of existing research done on topics related to modelling user behaviour based on location data.

2.1 Background

This section defines and discusses the concepts of data preprocessing and machine learning, which are the foundation for understanding the work presented in this thesis.

2.1.1 Data Preprocessing

Real-world datasets are highly influenced by negative factors such as the presence of noise, missing values, inconsistencies, and redundancies. These imperfections affect data quality, which influences the performance of data mining algorithms [11, 12]. The goal of data preprocessing is to improve data quality according to algorithm requirements. If data are not preprocessed, the algorithm may not work correctly, or in the best-case scenario, it will work, but the results will not be accurate. Another aspect of data preprocessing is that it can give the dataset a structure that enables more than one machine learning algorithm to be executed.

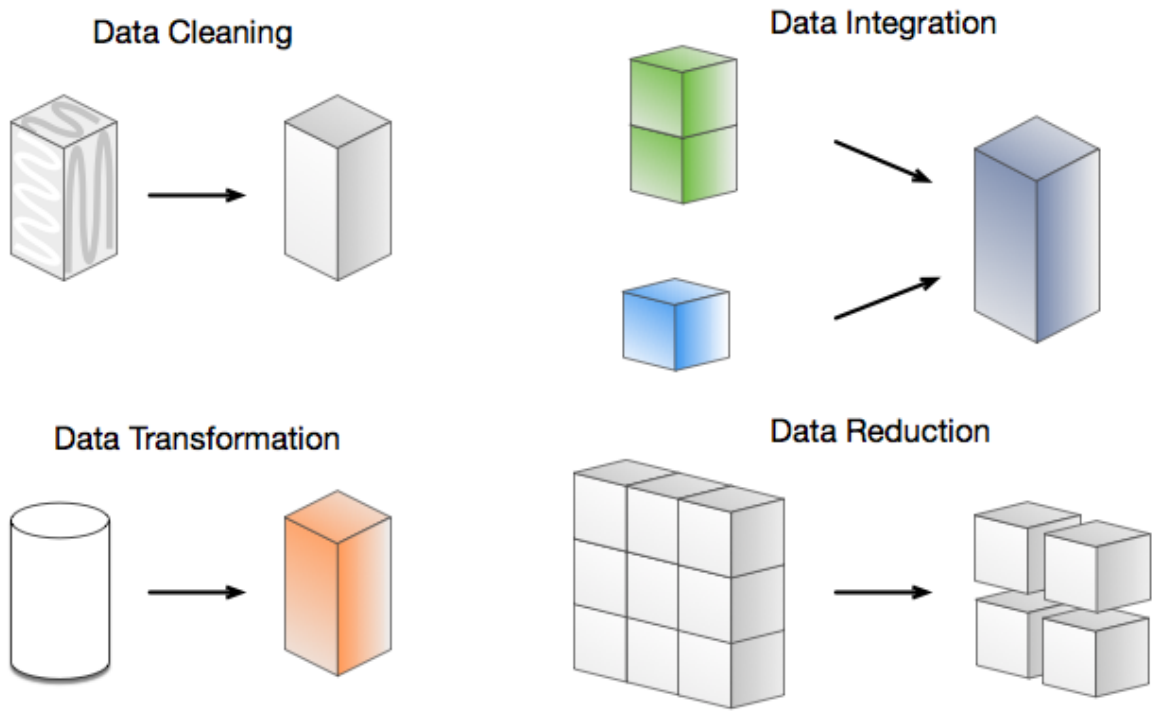

Figure 2.1: Data Preprocessing Categories

However, not all techniques may need to be used, depending on the data domain. Data preprocessing can be grouped into four categories, as shown in Figure2.1: *data cleaning*, *data transformation*, *data integration*, and *data reduction*.

Data Cleaning

Data cleaning focusses on handling missing, noisy, and inconsistent data. These characteristics are typical of real-world datasets. Missing values can occur for several reasons. For instance, an operator filling out a customer information form could skip some fields to make the process quicker; some fields are simply not applicable to all customers; or an equipment malfunction could lose a couple of records. Other malfunctioning equipment can record wrong values, or an issue may happen during data transmission, including some outliers in the dataset. Furthermore, discrepancies in code or naming conventions can result in inconsistencies in the dataset. Duplicate entries must also be “cleaned”.

Data cleaning techniques try to fill in missing values, smooth noisy data, identify and remove outliers, and resolve inconsistencies. Overall, machine learning algorithms can deal with some level of "dirty" data, but the methods are not very robust, nor do they address all types of issues. Some of the methods used to address each of these issues are described below.

1. *Missing values*: Not addressing this issue may lead to inaccurate models [13]. Various approaches can be used to handle missing values [14]. For example:

   - *Manually fill in missing values*: this can be feasible if there are few missing values;
   - *Eliminate or ignore missing values* [15]: this is straightforward to implement, but impractical unless most attributes of the entries are missing;
   - *Estimate missing values*: a constant can be used, or the mean of all entries. Another option is to use the expected value based on statistical methods [16] or other prediction approaches.

2. *Noise*: this is a random error that modifies the original value. Noisy data can generate outliers as well. This issue has an impact on model accuracy and needs to be tackled [17]. In the case of noisy data, the following methods may be used:

   - *Inspection*: a human operator can check the values after they are detected;
   - *Binning*: sorting all the data values and splitting them into bins with the same number of entries. Next, the bins are smoothed using the bin values or the values around them (e.g., using the mean);
   - *Clustering*: clustering the data into normal and anomalous values and removing the anomalous ones;
   - *Regression*: fitting the data with regression functions to smooth them.

3. *Inconsistency*: some inconsistencies can be corrected manually. For example, data gathered from scanned documents can be compared with the paper version of the documents.

If there are known constraints, automated tools may find data that do not respect these constraints.

Data Integration

Data integration is the process of merging data from multiple data sources. This process can be tricky and must be performed carefully to avoid inconsistency and redundancy. The main tasks of data integration are identifying and matching attributes, analyzing their correlation, and detecting any conflict between the various data sources.

Data Transformation

Data transformation is the process of transforming or consolidating data into an appropriate format for more efficient use. Data transformation techniques can be grouped into:

1. *Aggregation*: data aggregation can involve merging multiple attributes into a single attribute or combining two or more entries into a single entry. For example, individual transactions can be aggregated to daily sales;

2. *Normalization*: Normalization means adjusting attribute values measured on different scales to a common scale. Normalization attempts to give equal weights to all attributes. Some machine learning algorithms will not work properly without normalization. Min-Max, expressed in Eq. (2.1), is a popular normalization method:

$$min - max = \frac{x - X_{min}}{X_{max} - X_{min}} \tag{2.1}$$

   where x is the entry value, X_{min} is the minimum value in the dataset, and X_{max} is the maximum value in the dataset;

3. *Generalization*: involves replacing attributes by higher-level concepts. For example, annual income could be replaced by social class.

Data Reduction

Data reduction includes various methods to obtain a reduced representation of the original dataset while maintaining its properties and any existing intrinsic knowledge. Extracting the same insights from the reduced dataset should also be possible. Usually, data reduction comes with some degree of loss, and the trade-off must be taken into consideration. The main techniques used are:

1. *Feature Selection*: not all data features have the same importance, and feature selection techniques try to select the ones that best represent the data [18];

2. *Instance Selection* [19]: instead of selecting a subset of the features, instance selection tries to select a subset of instances to represent the full dataset;

3. *Discretization* [20]: this technique projects continuous attributes into a discrete domain, using non-overlapping ranges, for example.

2.1.2 Machine Learning

This section provides an overview of machine learning, focusing on the algorithms used in the UCAP framework. Initially, a high-level overview of machine learning will be presented. Later on, the concepts of the *K-Means* and the *Gradient Boosting* algorithms are discussed.

Overview

Machine learning is a part of artificial intelligence that explores algorithms that can receive input data and use statistical analysis to predict an output value within acceptable accuracy.

Machine learning techniques usually share a similar process. It starts with a dataset, which can be seen as a table where the rows represent observations and the columns are the values of the observed attributes. The dataset is split into at least two subsets, the training dataset and the test dataset. A validation dataset may also be created either at this moment or later in the

process. Other dataset splitting schemes [21, 22] exist, but will not be discussed here because they are out of the scope of this thesis.

In machine learning, the training process involves optimizing the parameters of a function, called the loss or cost function. The function and the parameters vary depending on the algorithm, but the goal is to minimize or maximize the loss function output when the model is applied to the training dataset. Once the model is optimized, it is then evaluated using the test dataset.

Machine learning algorithms are usually classified into three categories, according to the learning process used: supervised learning, unsupervised learning, and reinforcement learning.

In *supervised learning*, the algorithm learns from a training set that contains the desired output values (labels) [23]. The objective is to build a model that can predict the correct output for unseen data. This is analogous to the process in which a student learns from examples provided by a teacher. The student must generalize the examples into rules or functions that are applied when new examples arise.

Supervised learning is used primarily for two types of problems: classification problems and regression problems. In classification problems, the data have discrete labels. Each label can be seen as a category or a class, and the goal is to build a model that can assign new data entries to the most appropriate class. On the other hand, regression problems involve data with continuous labels, rather than the discrete classes in classification problems. For example, forecasting the energy consumption of a building is a regression problem.

In *unsupervised learning* [22], the training dataset does not contain any of the desired outputs. The algorithm's objective is to find patterns in the data by itself. It tries to find some intrinsic logic or rules that structure the training data. This is analogous to an individual who tries to find similarities between objects or events to organize them into classes or categories based on the individual's perception rather than any pre-established categories. These types of algorithms generate a better understanding of the data and provide insights into them. They can also generate new and useful inputs for supervised learning algorithms, which is how the

UCAP framework benefits from unsupervised learning.

Reinforcement learning is a type of learning in which the algorithm works as an agent that interacts with the environment, receiving feedback and adjusting its output accordingly. It is used mainly in decision-making problems, where the decisions cause consequences. It is analogous to learning by trial and error.

Next, two algorithms used in this work are detailed.

K-Means Algorithm

The K-means [24] is an unsupervised learning algorithm used to cluster unlabeled data. Its goal is to assign the data into k clusters based on the available features.

The algorithm's input is the data itself and the number of clusters k. The data samples are d-dimensional vectors $(x_1, x_2, \ldots, x_n)$ that will be assigned to $k \leq n$ clusters $C_1, C_2, \ldots, C_k$, with n being the number of samples. The algorithm starts by choosing k centroids for the clusters $(c_1, c_2, \ldots, c_k)$, which are also d-dimensional vectors. The centroids could be, for example, randomly selected data from the dataset. Then the algorithm iterates between the following two steps:

1. Data samples are assigned to the cluster with the nearest centroid. Equation (2.2) describes this step. The *dist* function computes the distance between two instances of the data, for example by using Euclidean distance.

$$C_i = \{x_j | dist(x_j, c_p) \geq dist(x_j, c_i), 1 \leq p \leq k\}. \tag{2.2}$$

2. The centroids are updated by taking the average of all data samples assigned to each cluster according to Eq. (2.3).

$$c_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \tag{2.3}$$

The algorithm runs until a stopping criterion is met. The stopping criterion could be, for

instance, when no data points change clusters or when some maximum number of iterations is reached.

Gradient Boosting

Gradient boosting [25] is a boosting algorithm that uses gradient descent [26] to update its predictors. Boosting is an ensemble approach in which the predictors are built sequentially. The idea is that subsequent predictors learn from the mistakes of their predecessors. Ensemble learning is a machine learning technique that uses multiple learners to solve a problem.

Gradient boosting algorithms use a loss function that varies according to the problem. For instance, the mean squared error can be used as a loss function. Such algorithms also need an ensemble of learners to make predictions, typically decision trees. Gradient boosting starts with a single learner. Subsequent models predict the residual error of the previous models. The prediction is given by the sum of all the models.

2.2 Literature Review

This section provides a literature review of research done on topics related to modelling user behaviour based on location data.

The increasing amount of available location data has increased interest in building prediction models around these data. One fundamental idea is that human mobility is predictable. Song *et al.* [27] explored the limits of this predictability and found in their test population that 93% of human mobility was predictable. To reach this conclusion, they measured user entropy from 45,000 mobile users.

Some studies have focussed on building models for next location prediction [3, 28, 29, 30, 31, 32].

Ye *et al.* [3] proposed a framework based on mixed Hidden Markov Models (*HMMs*) to (1) predict the next POI category, and (2) predict the next-visited POI based on the distribution of

the predicted category. They clustered users based on the frequency of visits using the K-Means algorithm and built one prediction model per cluster. The category level captured the semantic meaning of the user's activities. After getting the predicted category from the *HMM*, they used a ranking scheme to predict the next-visited POI. They evaluated the impact of clustering on the results and concluded that the clusters improved the results.

Noulas *et al.* [28] explored the problem of predicting the next-visited POI within 24 hours using check-in data. They proposed a set of features including user information, such as the visits that a user has performed and visits performed by the user's friends; global information that aggregates data related to the POIs themselves, such as the distance between POIs and existing transitions; and finally, they also used temporal features. These features were used to evaluate a linear regression model and an M5 tree model with a dataset consisting of 35 million check-ins from Foursquare. The M5 trees performed best in the evaluated metrics.

Preoţiuc-Pietro and Cohn [29] studied the check-in patterns in LBSNs. Their features consisted of the transitions that users make when they go from one POI to another. They clustered the users based on their POI category transitions using the $K-Means$ algorithm. They also built a model to predict the next POI category, inputting the same set of features to different Markov models (*MMs*), and developed methods using the most frequent category over a period. Their results showed that higher-order *MMs* did not achieve improvement over lower-order models. However, they did not merge the clustering with the prediction.

Trasarti *et al.* [30] proposed a system, called *MyWay*, to predict users' exact future position based on mobility profiles, which are based on the users' trajectories. These trajectories were constructed using the users' raw position. However, the raw position data do not need to be shared to make the prediction, reducing privacy risks. Moreover, these authors also proposed a collective strategy to consider more individuals' data, but this would be applied only with data shared by users. Besides, they proposed a hybrid approach that mixes both strategies. *MyWay* was evaluated with a dataset gathered for insurance purposes and consisting of 9.8 million car trips. The hybrid strategy was the one that performed best.

Nguyen *et al.* [31] applied a variety of machine learning methods to the next-visited POI problem, such as Markov models, Support Vector Machines (*SVM*), and decision trees. In their work, they built an Android app to collect GPS data from users. They collected each user's position every five minutes. According to their results, the *SVM* algorithm performed best.

Chen *et al.* [32] presented three approaches based on Markov models: a global model, based on all trajectories; a personal model based on a specific user; and a regional model in which trajectories were clustered to build cluster models. Their data were collected from the traffic system and represented cars passing specific locations. Their evaluation showed that their approach outperforms two other methods, *VMM* and *WhereNext*.

In contrast with these studies [3, 28, 29, 30, 31, 32], the proposed research focusses on users' current activity preferences, not on future locations. Moreover, the UCAP framework clusters users based on the places they have been and uses this information as a feature for the prediction model, rather than building one individual prediction model per cluster. Other studies have focussed either on LBSN data or other types of data, whereas this study proposes a framework that works with both LBSN and GPS data and uses external context to improve the prediction model.

Other studies have explored various ideas such as predicting where a user will be at a specific time with datasets other than LBSN data [6, 33, 34].

Cao *et al.* [6] presented a framework to predict the probability of a user visiting a specific location at a given time. They first evaluate the possibility of a user visiting a location and then checked whether the user would visit a specific location. They evaluated their approach using data from three different LBSN datasets.

Liu *et al.* [33] proposed a method to predict where a user will go next at a specific time. Their method is based on recurrent neural networks, which usually uses a single transition matrix. However, they used two matrices: a time-specific and distance-specific transition matrices. They have evaluated their model with data from a LBSN dataset and also a dataset of terrorist incidents. Their method outperformed eight other methods presented.

Lv *et al.* [34] presented two models: (1) a spatial-temporal model and (2) a next-visited POI prediction model based on Hidden Markov Models. Their dataset was periodically gathered from LTE control-plan traffic generated by user equipment, such as mobile phones and other electronic devices. They clustered nearby locations to reduce oscillation and identified the most important POIs from users to use in their models. They grouped users into four previously established clusters based on their daily patterns. Their models outperformed the baseline methods in the evaluation.

Unlike [6, 33, 34], this study does not try to predict where the user is going at a specific time and also considers data sources other than LBSNs. Furthermore, it is not restricted by any sampling periodicity scenario. The UCAP framework can use data with either static or variable sampling frequency. Also, the number of clusters is optimized when building a new model, instead of using a previously established value. Moreover, other studies [6, 33, 34] do not consider the external context, which may influence user behaviour.

Studies of building location recommender systems can be found in [35, 36, 37, 38]. Yao [35] proposed a POI recommendation model based on Poisson factorization. The model first profiles the temporal popularity of the venues within one day and then matches the POI profiles with users' preferences and routines. The users' preferences and routines are gathered from a LBSN dataset. The model can achieve improved evaluation over other factor-based models.

Yao *et al.* [36] used a tensor factorization check-in representation, with users, locations, and time slots as dimensions, to build a recommendation framework. They added social and spatial regularization terms. The first of these was related to the users' friends, whereas the latter was related to the distance between the users' check-ins. Yao *et al.* evaluated their work using two datasets from two LBSNs and achieved better results than 12 other methods which were used as the baseline.

He *et al.* [37] presented a model to handle location group recommendations. They took individual preferences into consideration and then used several factors to select recommended locations for the group. The process was inspired by an economics principle, the Pareto Op-

timality. They evaluated their idea using two datasets gathered from LBSNs, and their results showed improvements over baseline models.

Capdevila *et al.* [38] presented a hybrid recommender system for geolocated data. They gathered data from Foursquare using a parallel version of the Quadtree algorithm, which was a clever way to overcome the limitations of the API. They used a hybrid approach, with textual sentiment analysis for the collaborative-filtering recommendation part and aggregated reviews by users for the content-based recommendation part.

The UCAP framework could be extended to perform recommendations, but this was not the main goal of this work, unlike the studies mentioned above [35, 36, 37, 38]. In addition, these studies did not consider any context other than social relations, location, and time.

Lian *et al.* [39] presented a problem consisting of two tasks. First, they predicted whether users are willing to explore unvisited POIs. They evaluated a logistic regression and a classification and regression tree for this problem. Then, the result is used to choose the approach taken to predict of the next-visited POI. If the users were willing to explore a POI they have never visited, the researchers used a recommender system to find the best matches. If the user was more likely to go to a POI where he/she had already been, a Markov Model and a temporal regularity model were input into the Hidden Markov Model. Their system considered only LBSN data, whereas the system proposed here is not restricted to such data.

Finally, Yang *et al.* [2] considered spatial and temporal dimensions for modelling user preferences (*STAP*). They built models for these dimensions separately and then merged them using a fusion framework. Furthermore, they used a tensor factorization model to exploit the similarities among users. Their study was similar to this one, and their dataset and results were used to evaluate the UCAP framework. In addition, UCAP considers not only spatial and temporal information, but also the external context, such as temperature and wind conditions, because it may affect user preferences. Moreover, their evaluation was performed based on LBSNs datasets, whereas this study also considers datasets gathered without any user interaction. *STAP* outperformed 10 baseline approaches presented in [2], including sequential pattern

mining, temporal, spatial, and spatial-temporal approaches.

2.3 Summary

This chapter provides an overview of the concepts related to various topics that assist in understanding the UCAP framework. More specifically, an introduction to data preprocessing terminology has been presented. In addition, an introduction to machine learning and the algorithms that are used in this research has been provided. Finally, current studies on various topics related to modelling user behaviour based on location data were discussed and contrasted to the methods used in the UCAP framework, which is the approach presented in this thesis. These methods are discussed in more detail in the following chapters.

Chapter 3

UCAP Framework

This chapter describes the proposed framework. The primary goal of the UCAP framework is to build a model for predicting users' current activity preferences based on their current context. Figure 3.1 shows the data flow inside the UCAP framework.

The UCAP framework consists of four layers: the *Preprocessing Layer*, the *Feature Engineering Layer*, the *Prediction Layer* and the *Presentation Layer*. Feature engineering can be seen as part of the data preprocessing, but as UCAP adjusts the features along with the prediction model, the feature engineering process is represented as a separate layer. The four layers ensure better separation of responsibilities and help to decouple the functionalities from each other.

The following sections explain each layer and its components and how each fits into the framework.

3.1 Preprocessing Layer

The *Preprocessing Layer* is the framework's input interface and the first step in training the prediction model. Two datasets serve as inputs to this layer: a *User Event* dataset and an *External Context* dataset.

The *User event* dataset contains the following elements:

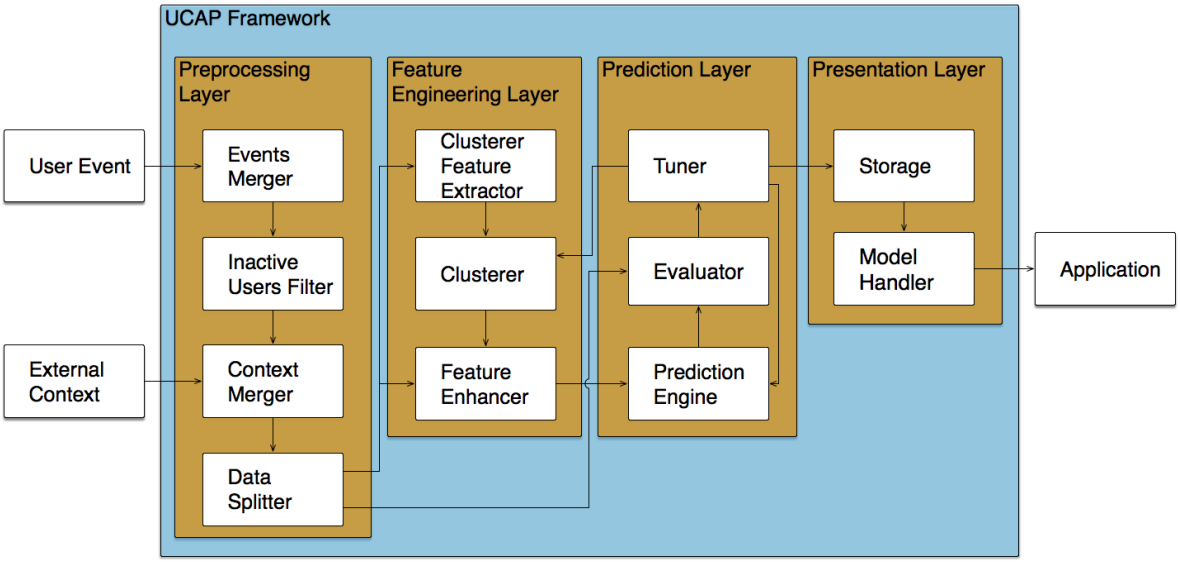

Figure 3.1: Data flow of the UCAP Framework

- Unique user identifier (collected according to privacy policies)
- Location
- Timestamp
- Activity

The exact elements of the *External context* dataset are not enforced by the UCAP framework as it cannot know *a priori* which and how many external contextual attributes will be used, but it should contain:

- Location
- Timestamp
- Context 1
- ...
- Context n

User Event Dataset

User ID	Location	Timestamp	Activity
1	(42.9849, -81.2453)	Apr 10, 2017 - 18:00:42	Pizzeria
2	(54.0076, -114.6569)	Apr 21, 2017 - 09:50:24	Hotel
...	...	...	...
1	(-32.088, -51.7412)	May 09, 2018 - 11:35:38	Gym

External Context Dataset

Location	Timestamp	Temperature	Hurricane
(34.6466, -78.4039)	Mar 27, 2017 - 09:39:53	25.6	1
(28.18461, -3.0031)	Sep 6, 2017 - 10:02:20	15.4	0
...	...	...	...
(14.0282, 104.5221)	Oct 24, 2017 - 14:09:15	-5.2	0

Figure 3.2: UCAP inputs example.

Figure 3.2 shows two example datasets: a *User Event* dataset and an *External Context* dataset. The *External Context* dataset contains two contextual attributes: the temperature and whether a hurricane is present in an area or not, represented as 1 and 0, respectively.

The *User Event* dataset will most likely be gathered from online interaction with digital content, either as user input, such as the check-ins or without any user interaction, for example, by a background process that stores the user location from time to time. The data gathered with user interaction may contain duplicates, while the data gathered from background location services may contain uncertainty, as it was shown in Fig. 1.1 and explained in Chapter 1. Both scenarios are handled by the *Preprocessing Layer*.

The *context* from an *External Context* dataset may consist of numbers or labels. Each data type is processed differently by the *Feature Engineering Layer*. The *Preprocessing Layer* is concerned only with merging the context with the events according to the location and timestamp.

Moreover, this layer filters inactive users based on their usage pattern. This layer is made

up of four components: the *Events Merger*, the *Inactive Users Filter*, the *Context Merger*, and the *Data Splitter*. These components are detailed below.

3.1.1 Events Merger

The *Events Merger* component is a data aggregation component data allows that the UCAP framework to accept as a *User Event* dataset both LBSN check-in datasets and datasets gathered using background location services. This component is responsible for dealing with duplicates generated by user input and multiple entries generated by GPS uncertainty.

Figure 3.3 (a) shows an example of a *User Event* dataset with both issues. The first two entries have the same user, location, timestamp, and activity. They represent duplicates generated by user input. The last three entries have the same user, location, and timestamp, but different activities. They depict entries generated by GPS uncertainty.

The *Events Merger* component merges all the events from a user with the same location and the same timestamp into a single event by changing its data representation. First, this component takes the set of all activities performed by a user in a specific location and at a specific timestamp. There are no duplicates in a set, which solves the user input duplication problem.

Next, the *Events Merger* component changes the activity data representation. The *User Event* dataset uses labels to represent activities performed by a user at a specific location and timestamp, such as the example given in Fig. 3.3 (a). Multiple entries may exist in the dataset due to GPS uncertainty. Probabilities are a good representation of uncertainty, and the *Events Merger* component uses them to represent multiple activities performed by the same user with the same location and timestamp. All the activities performed in an event have the same probability, and their sum is one. Fig. 3.3 (b) depicts the output of this component, given Fig. 3.3 (a) as input. This new data representation solves the issue generated by GPS uncertainty.

Finally, a new issue arises with this representation. An event may become irrelevant if the probabilities are spread among many activities. To handle the issue, if an event has more

User ID	Location	Timestamp	Activity
u_1	l_1	t_1	a_1
u_1	l_1	t_1	a_1
u_1	l_2	t_2	a_1
u_1	l_2	t_2	a_2
u_1	l_2	t_2	a_3

(a)

User ID	Location	Timestamp	$P(a_1)$	$P(a_2)$	$P(a_3)$
u_1	l_1	t_1	1	0	0
u_1	l_2	t_2	1/3	1/3	1/3

(b)

User ID	Location	Timestamp	$P(a_1)$	$P(a_2)$	$P(a_3)$	$P(a_\kappa)$
u_1	l_1	t_1	1	0	0	0
u_1	l_2	t_2	0	0	0	1

(c)

Figure 3.3: User Events dataset before and after the Events Merger

activities than a particular threshold n, the component uses a unique activity κ to represent all the activities. Only this activity is performed in this case. The threshold n is a parameter of the *Events Merger* component. Fig. 3.3 (c) gives the output of this component, given Fig. 3.3 (a) as input and $n = 2$ as the threshold.

3.1.2 Inactive Users Filter

In this study, users were considered to have two kinds of status: active and inactive. Active and inactive users were differentiated based on the frequency of each user's events during a period. Inactive users provide no value from the application perspective and affect negatively the dataset as "dirty" data.

The UCAP framework is flexible regarding this frequency because the definition of an active user may vary depending on the business domain. For example, an application that shows the daily news can expect to be used daily by active users, but a sports results application could define a user as active who used it once a week.

The *Inactive Users Filter* is a component that performs data cleaning by removing events related to inactive users from the *User Event* dataset. First, the time during which events happen is split into time intervals of the same duration, building a set $T = \{t_0, t_1, \ldots, t_l\}$, where the duration can be one day or one month, for example. Next, given E, the set containing all events; U, the set of all users; and $E_{u,t}$, the set of events that a user $u \in U$ performed during the time window $t \in T$, the set of active users U' can be built according to Eq. (3.1). A user is considered to be active if the user generates m or more events per defined time window. For example, to apply the equation to the news application example, the time window would be set to one day, and $m = 1$. The *Inactive Users Filter* will remove any event from a user who is not in U'.

$$U' = \{u \in U \mid |E_{u,t}| \geq m, \forall t \in T\} \tag{3.1}$$

Location	Timestamp	Temperature	Condition
l_1	t_1	15.1	Thunderstorm
l_2	t_2	18.2	Light Fog

(a)

User ID	Location	Timestamp	$P(a_1)$	$P(a_2)$	$P(a_3)$	$P(a_K)$
u_1	l_1	t_1	1	0	0	0
u_1	l_2	t_2	0	0	0	1

Location	Timestamp	Temperature	Condition
l_1	t_1	15.1	Thunderstorm
l_2	t_2	18.2	Light Fog

User ID	Location	Timestamp	$P(a_1)$	$P(a_2)$	$P(a_3)$	$P(a_K)$	Temperature	Contidion
u_1	l_1	t_1	1	0	0	0	15.1	Thunderstorm
u_1	l_2	t_2	0	0	0	1	18.2	Light Fog

(b)

Figure 3.4: Example of: (a) External Context dataset; (b) Context Merger being performed

3.1.3 Context Merger

The UCAP framework assumes that users' activity preferences are not only related to spatial-temporal attributes, but also to the external context. For example, severe weather conditions may make outdoor activities challenging to perform. Moreover, special occasions such as the Olympic Games influence the human behaviour in the place they occur and therefore should be taken into account. For example, Fig. 3.4 (a) depicts an *External Context* dataset with information related to weather.

The *Context Merger* component integrates the *External Context* dataset with the *User Event* dataset, which has already been preprocessed. It performs a data integration process, merging them into a unique dataset. For instance, Fig. 3.4 (b) shows this component merging the *User Event* dataset from Fig. 3.3 (c) with the *External Context* depicted by Fig. 3.4 (a).

3.1.4 Data Splitter

The last component of the *Preprocessing Layer* involves splitting of the data into training and validation datasets. The training dataset is used to train the prediction model, whereas the

validation dataset is used to evaluate this model and help select its parameters.

The *Data Splitter* orders the data chronologically to represent the most current user behaviour and then uses the first $q\%$ as training dataset and the last $(100 - q)\%$ as validation dataset, with $0 \leq q \leq 100$. The split is done this way because the prediction model will be used along with data that are more recent than the training dataset. Hence, this scenario is replicated with the validation dataset.

3.2 Feature Engineering Layer

The results of prediction techniques are often improved by using newly generated features based on existing ones. The *Feature Engineering Layer* transforms the dataset received from the *Preprocessing Layer* by encoding it properly and merging it with newly generated features. It has two outputs:

- the training dataset with the new set of features themselves; and
- the data transformers that generated these features, i.e., the operations used to generate the features. Other layers may apply these data transformers in other datasets.

The *Feature Engineering Layer* is composed of three components: the *Clusterer Feature Extractor*, the *Clusterer*, and the *Feature Enhancer*. These components are described below.

3.2.1 Clusterer Feature Extractor

The first feature that the UCAP framework generates is related to the cluster to which each user belongs. The clustering process is described in the next subsection. However, the critical point, for now, is the fact that the clustering is based on the weighted frequency of activities performed by each user. This information is not available in the *User Event* dataset and must be generated. Hence, the training dataset must go through a data transformation process before

being used by the clustering algorithm. The *Clusterer Feature Extractor* is the component that performs this task.

The *Clusterer Feature Extractor* faces two challenges. The first challenge is not knowing how many events a user u performed. Counting may result in, for example, 5 events from user u_1 and 5,000 from user u_2. Because this unknown gap between results can influence the model training, the numbers will be scaled to a known interval. The second challenge involves the uncertainty of how often a user performs an activity. For instance, let activity a_1 be "having lunch", and activity a_2 be "going to school". Almost every user would perform a_1, whereas only a fraction of users would perform a_2. Hence, a_1 is not a useful feature for distinguishing users, whereas a_2 could distinguish "students" from "non-students".

To deal with the gap discrepancy between the total number of events and the fact that some activities may be better metrics than others for distinguishing users, the UCAP uses a weighting scheme. This weighting scheme is inspired by a widely used technique in information retrieval and text mining, *TF-IDF* (*Term Frequency-Inverse Document Frequency*) [40]. In text classification, documents have different lengths, and each text is written both with common words, such as "the", "and", "or", and words that are specific to a topic, for example, "algorithm", which would be related to "computer science". This idea was adapted as follows: each user u was considered as a document, and each activity a as a word. An event performed by a user at a specific timestamp $e_{u,t}$ contains a set of activities, E_u representing the set of events generated by user u with cardinality $|E_u|$. Equation (3.2) defines the tf function, which can be described as the ratio of the number of times that user u performed an activity a over all the activities u has performed. This equation deals with the first challenge described above. The second challenge is addressed by the function $idf(a)$, shown as Eq. (3.3). In this equation, U is the set of all users, and A_u is the set of activities performed by user u. The tf-idf function, shown in Eq. (3.4), combines the two previous equations. This equation is used to build the dataset used by the *Clusterer* component to segment the users based on their activity preferences.

$$tf(a,u) = \frac{|\{e_{u,t} \in E_u \mid a \in e_{u,t}\}|}{|E_u|} \tag{3.2}$$

$$idf(a) = \log \frac{|U|}{|\{u \in U \mid a \in A_u\}|} \tag{3.3}$$

$$tf\text{-}idf(a,u) = tf(a,u) \times idf(a) \tag{3.4}$$

Suppose that user u_1 from the example performed activity a_1 twice and a_2 three times, whereas user u_2 performed activity a_2 in all 5,000 events. In this case, the tf function is calculated as $tf(a_1,u_1) = \frac{2}{5}$, $tf(a_2,u_1) = \frac{3}{5}$, $tf(a_1,u_2) = 0$, and $tf(a_2,u_2) = 1$. Next, the idf function is computed as $idf(a_1) = \log \frac{2}{1} = 0.301$ and $idf(a_2) = \log 1 = 0$. Finally, the $tf\text{-}idf$ function results are $tf\text{-}idf(a_1,u_1) = 0.0602$ and $tf\text{-}idf(a_1,u_2) = 0$. Notice that $idf(a_2) = 0$ means that this activity is not useful to differentiate users.

3.2.2 Clusterer

In addition to the issues regarding the number of events that a user may have performed, there is the fact that these events may be sparse in time and also may present a data scarcity problem because the number of activities performed per user is usually only a fraction of all the possible activities [41]. Time is a fundamental feature of the UCAP framework, and there can be many gaps in users' daily routines in the datasets. To fill these gaps, users with similar preferences are considered to have a similar routine. By clustering these users, their events can be considered under a single entity, the "group of users", rather than as individuals. For example, "parents", "university students", and "hikers" could each cluster several individuals. Another advantage of this approach is that new users can fit into a cluster after a few events and use a model that is already built and trained.

The *Clusterer* component groups users based on the weighted frequency with which they

perform the existing activities weighted by the *Clusterer Feature Extractor* component. Users in the same cluster are supposed to exhibit similar behaviour regarding activities, and for this reason, the cluster itself is a feature used by the *Prediction Layer*. This component builds and outputs a set of tuples with the users and the cluster to which each belongs.

3.2.3 Feature Enhancer

This component receives as input the data from both the *Clusterer* and the *Preprocessing Layer*. The data may contain features that are not encoded appropriately.

The data provided by the *Clusterer* are composed of tuples with a unique user identifier and an integer representing the cluster to which this user belongs. The user identifier is used to merge these data into the feature set, but it is not a feature by itself. The number that identifies the cluster is a feature to be included in the feature set.

Each cluster represents an entirely independent category of users. This kind of feature is called a categorical feature. If an integer is used to represent each cluster, the *Prediction Engine* can assume, for example, that cluster 1 and 2 are more similar than clusters 1 and 5. Moreover, it assumes some ordering between clusters, which is not true. The approach taken is to create one feature per cluster. This way, if there are k clusters, they will be represented by k features. This representation is called *1-of-K* scheme [42], and it is used to encode k mutually exclusive states, the clusters in this case. For example, if there are three clusters and a user belongs to cluster 3, this user's cluster is represented by $(0, 0, 1)$.

The context from the *External Context* dataset may consist of numeric values or labels. Numerical values do not need any modifications. Labels are categorical features and use the *1-of-K* representation. The *Feature Enhancer* does not care about what each context is but simply adjusts the data representation if needed.

Spatial features such as latitude and longitude are numeric values and do not need any adjustment. These features represent the location where an event occurs and are fundamental to the functioning of the framework.

Table 3.1: Features with domain

Feature	**Domain**
Cluster	Categorical
External Context	Categorical or Numerical
Latitude	Numerical
Longitude	Numerical
DayTime{sin, cos}	Temporal
WeekDay{sin, cos}	Temporal
Week{sin, cos}	Temporal
Month{sin, cos}	Temporal
Weekend	Categorical

Temporal features, based on the events' timestamps, receive a different treatment. Several features are generated based on them. The periodic temporal features are projected onto a unit circle, using their sine and cosine as their representation, to maintain their properties. This transformation is done using Eq. (3.5). In this equation, t represents the original temporal feature, and ω_t is its frequency.

$$\hat{t} = \left\{ \sin\left(2\pi t\omega_t\right), \cos\left(2\pi t\omega_t\right) \right\} \tag{3.5}$$

This approach helps deal with frequency boundaries. For example, a day has 1,440 minutes. By using Eq. (3.5), the distance between minute 1,439 and minute 0 is the same as the difference between minute 0 and minute 1. If the original feature t were used, there would be a considerable gap between these numbers.

Besides the periodic temporal features, there is a categorical feature indicating whether the timestamp is on a weekday or a weekend. The set of features is summarized in Table 3.1.

3.3 Prediction Layer

The *Prediction Layer* deals with training a prediction model, evaluating it, and running the validation process to tune its parameters. This layer has two types of inputs: data and data transformers.

The *Prediction Layer* receives the feature set generated by the *Feature Engineering Layer* (based on the training dataset) and the validation dataset from the *Data Splitter* component. The validation dataset is encoded using the data transformers received from the *Feature Engineering Layer*. These data transformers are the same as those used on the training dataset.

This layer is composed of three components: the *Prediction Engine*, the *Evaluator*, and the *Tuner*. These components are described in the following subsections.

3.3.1 Prediction Engine

The *Prediction Engine* trains a model to predict which activities users will most likely perform based on their current context. Its output is the trained prediction model. It receives the feature dataset built by the *Feature Enhancer* component and uses it along with the activity probabilities computed by the *Events Merger* component.

This trained model outputs predicted activity probabilities $\hat{a}$ based on the objective function shown in Eq. (3.6), where l is the loss function, a is the ground truth, i.e., the probability vector computed by the *Events Merger*, and Ω is the regularization function, which represents the complexity of the model. The index n is the size of the training dataset. The function $\mathcal{L}$ is to be minimized.

$$\mathcal{L}(model) = \sum_{i=1}^{n} l(a_i, \hat{a}_i) + \Omega(model) \tag{3.6}$$

3.3.2 Evaluator

The trained model must be evaluated to tune the training parameters. The *Evaluator* uses the data transformation processes generated by the *Feature Engineering Layer*, the model trained by the *Prediction Engine*, and the validation dataset received from the *Data Splitter* to evaluate the model. Various evaluation metrics are computed in the experiments for study purposes, but the one that is used to tune the models in the validation step is the *Accuracy*. This metric checks whether the predicted activity with the highest probability corresponds to the ground truth and

is computed as the ratio of correct predictions to the total number of predictions made.

3.3.3 Tuner

The *Tuner* coordinates the prediction model paramater tuning process. This process, depicted in Fig. 3.5, involves components from both the *Feature Engineering Layer* and the *Prediction Layer*.

The UCAP framework takes into consideration that clustering techniques may have parameters that can be tuned. This ends up including both the *Clusterer* component and the *Feature Enhancer* in the tuning process. This process starts when the *Clusterer Feature Extractor* sends the clusterer features (step 1). Next, clusters are created by the *Clusterer* (step 2). The features are then encoded by the *Feature Enhancer* (step 3). The enhanced features are sent from the *Feature Engineering Layer* to the *Prediction Layer* and used in the training process by the *Prediction Engine* (step 4). The model is evaluated by the *Evaluator* using the validation data (step 5). The *Tuner* component keeps track of the evaluation results. If the stopping criteria were not met, it sends new parameters either to the *Clusterer* (step 6) or the *Prediction Engine* (step 7). A stopping criterion can be, for example, the number of iterations or the convergence of a function whose input is the evaluation results. When the process is done, the *Tuner* sends the trained model to be stored (step 8). This process is also listed as pseudo-code in Algorithm 1. Notice that this component is generic and does not specify the tuning algorithm to be used, which is specific to the implementation.

3.4 Presentation Layer

The *Presentation Layer* stores the trained model and all the data transformation processes needed by the prediction model; it can also run the prediction model on new data. This layer is made up of two components: the *Storage* and the *Model Handler*, which are described below.

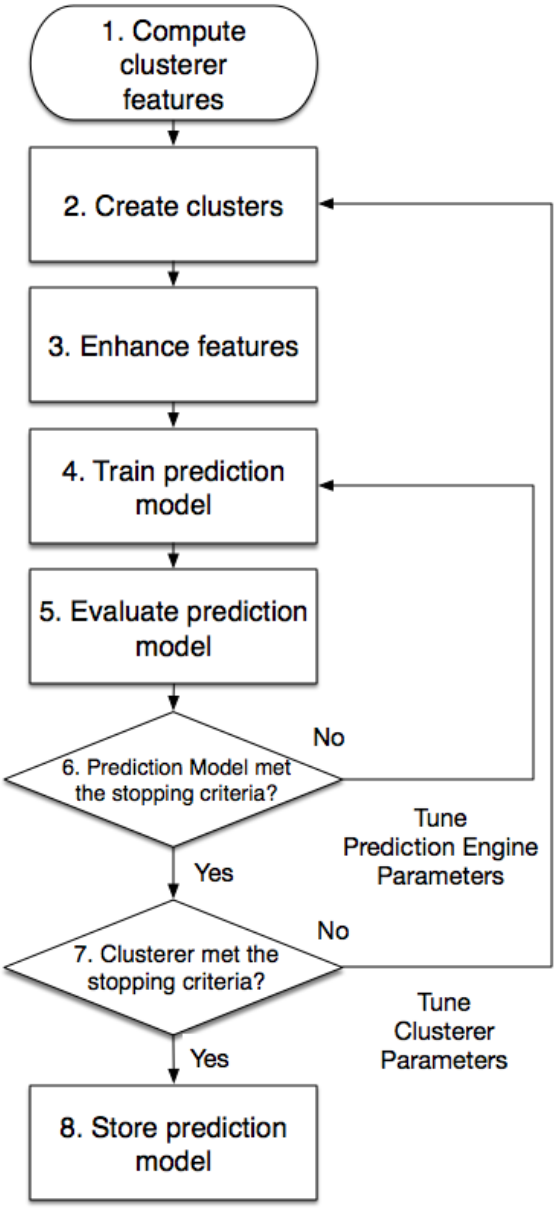

Figure 3.5: Tuning process flow.

Algorithm 1: Simplified Tuning Process

```
   input  : trainData, validData
   output: Clusterer Feature Extractor Data Transform
   output: Trained Clusterer
   output: Prediction Feature Extractor Data Transform
   output: Trained Prediction Model
   output: Training Results
 1 o ← Tuner();
 2 clFE ← ClusterFeatureExtractor(trainData);
 3 clData ← clFE.getData();
 4 while o.clusterStoppingCriteria() is false do
 5     cl ← Cluster(o.clParams(), clData);
 6     PredFE ← PredictionFeatureExtractor(trainData, clData, cl);
 7     predData ← PredFE.getData();
 8     while o.predStoppingCriteria() is false do
 9         pred ← PredictionEngine(o.predParams(), predData);
10         eval ← Evaluator(validData, clFE, cl, PredFE, pred);
11         o.EvalResults(eval);
12     end
13 end
14 o.storeModel(clFE, cl, PredFE, pred)
```

3.4.1 Storage

The *Storage* component receives and stores the results, data transformations, and trained models generated by the *Feature Engineering Layer* and the *Prediction Layer*. The data transformations and the trained models will later be used by the application to provide predictions.

The *Storage* component must be implemented in such a way that when a data transformation or a trained model is requested, only the most recent version is retrieved.

3.4.2 Model Handler

The *Model Handler* is the interface between the application and the UCAP framework. It retrieves the trained data transformations and model and runs them to provide a prediction.

3.5 Summary

This chapter has introduced the UCAP framework and discussed each of its layers: the *Preprocessing Layer*, the *Feature Engineering Layer*, the *Prediction Layer*, and the *Presentation Layer*. The discussion explained the responsibilities of each layer and the components of the framework and also explained how they interact.

Chapter 4

Case Studies

This chapter presents the UCAP implementation and the experiments used to evaluate UCAP. It starts with a description of the experimental environment and implementation details. Next, the five datasets used in all the experiments are presented, analyzed, and discussed. Then this chapter presents three experiments that evaluate the UCAP framework. Each experiment has its objectives detailed, as well as the metrics being used. Moreover, this chapter provides a discussion of the experimental results. This chapter is organized into two sections. The first will present the implementation and the experiments, and the second will present a discussion of the experimental results.

4.1 Implementation and Experiments

The UCAP framework was implemented using Python. The experiments were run on a server with 24 Intel Xeon CPUs at 2.60 GHz and 96 Gb RAM.

This section describes the shared components; experiment-specific component implementations are described in the subsection on that experiment.

Table 4.1: XGBoost Parameters

Parameter	Value
Minimum Child Weight	50
ETA	0.1
Colsample by Tree	0.9
Maximum Depth	6
Subsample	0.9
Lambda	1.0
Booster	gbtree
Objective	binary:logitraw
Gamma	0

Clusterer

The *Clusterer* component uses the *K-Means* implementation from Scikit-learn [43] with *Euclidean distance*. The *K-Means* algorithm was used due to its simplicity and effectiveness.

Other distances could have been used, but studies have shown that for high-dimensional data spaces, results are similar [44]. Different clustering techniques were initially considered, but preliminary tests showed that their performance was similar.

Prediction Engine

A gradient-boosting technique called eXtreme Gradient Boosting (XGBoost) [45] was used as the prediction engine. To the best of the author's knowledge, this is the first time that this algorithm has been used in this problem domain.

Overall, tree-based methods are highly accurate and easy to use and to interpret. In particular, XGBoost was involved in 17 out of 29 challenge-winning solutions at Kaggle[1] in 2015 [45]. This algorithm was used to minimize the objective function given in Eq. (3.6). Table 4.1 shows the parameters used by the XGBoost model.

The XGBoost algorithm does not output the required predicted activity probabilities, but rather scores. These scores are related to how likely each activity is to be performed by the user, given the context described by the input feature set. To use these scores z as probabilities,

[1] https://www.kaggle.com

they are input to the softmax function[42] using σ as described by Eq. (4.1). In this equation, a is an activity, A the set of all existing activities, and z_a is the score of activity a. The softmax result for each entry in the dataset is the output of the prediction model. The *Prediction Engine* output is the output of Eq. (4.1).

$$\sigma(z_a) = \frac{e^{z_a}}{\sum_{a' \in A} e^{z_{a'}}} \tag{4.1}$$

Tuner

A grid-search approach was implemented as the *Tuner* component, and it helped choosing the number of clusters for the *K-Means* algorithm.

4.1.1 Data Analysis

The experiments used five datasets, which are described in Table 4.2. The *LDN Small* dataset, the *MTL* dataset, and the *LDN Large* dataset were provided by a multi-media company specialized in weather-related content and technology. All the data contained anonymized pseudo-identifiers and were collected in accordance with privacy policies. No personally identifiable information about users was used. The *NYC* and the *TKY* datasets were gathered as described in [2].

The *External Context* dataset consisted of temperature, wind speed, and weather condition, such as "Thunderstorm", "Clear", or "Snow Showers". For London, ON, these data were extracted from the Canadian Historical Climate Data[2]. For New York and Tokyo, these data were obtained through the Weather Undergroung API[3].

Figures 4.1, 4.2 and 4.3 were plotted to depict the datasets more clearly. Figure 4.1 shows the probability distribution of the number of events per user in the datasets. The shape of the curves is very similar for all datasets, where most users have very few events. A successful

[2]http://climate.weather.gc.ca/historical
[3]https://www.wunderground.com/weather/api/

Table 4.2: Datasets Description

Dataset	LDN Small	LDN Large	MTL	NYC	TKY
Location	London, ON Canada	London, ON Canada	Montreal	New York	Tokyo
Data Source	GPS	GPS	GPS	LBSN	LBSN
Start Date	March 19th 2017	June 12th 2017	March 19th 2017	April 12th 2012	April 12th 2012
End Date	May 2nd 2017	October 19th 2017	May 2nd 2017	February 16th 2013	February 16th 2013
Number of Events	2,481,358	76,098,559	11,782,099	227,428	573,703
Unique Users	40,781	129,609	196,100	824	1,939
Unique Activities	77	84	72	251	251

approach to any problem that uses location data needs to handle this scenario. Figure 4.2 displays the number of events as a time series. Highlighting the different timeframes between the datasets is good, whereas the *NYC* and *TKY* datasets covered about ten months of data, the other two contained only about two months of data and may seem smoother, although they are not. Finally, Fig. 4.3 gives the plot of the number of activities per event. Each event is related to a unique user in a specific timestamp. The difference between the LBSN dataset, which presents a single activity per event for all the events, and the GPS datasets are noticeable.

4.1.2 Experiment 1: Impact of the UCAP Components

The objective of this experiment was to evaluate the impact of the main components used by the UCAP framework. The evaluation process used five variations of the framework: UCAP without the *Events Merger*, UCAP without the *Inactive Users Filter*, UCAP without *External Context*, UCAP without the *Clusterer*, and finally *UCAP* with all its components. Each variation of the framework used in this experiment was trained independently.

The set-up used the *LDN Small* dataset. The value used for the parameter m from the *Inactive Users Filter* was 5, and all the data were placed in the same time window, which

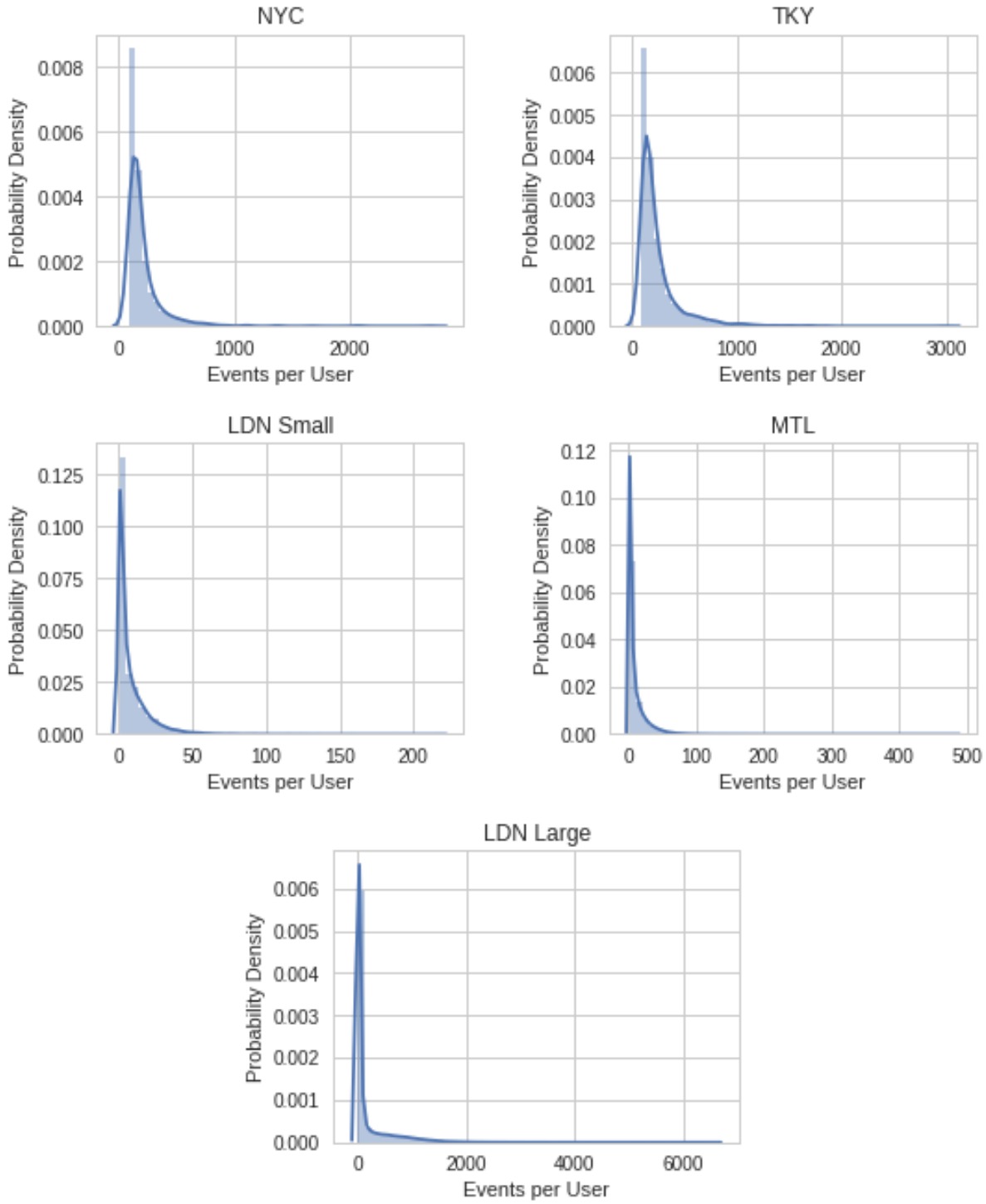

Figure 4.1: Probability distribution of the events per user

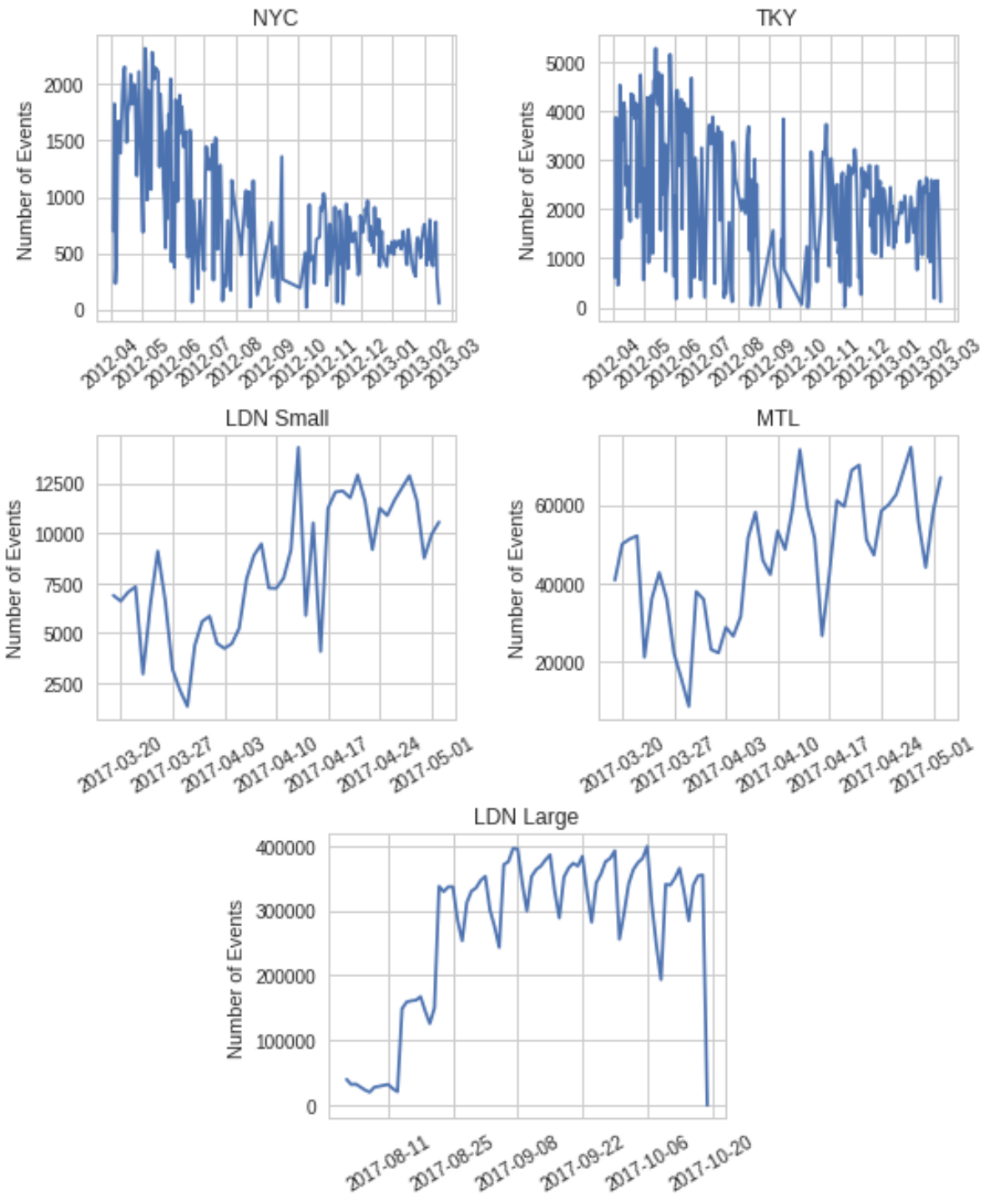

Figure 4.2: Number of events along time

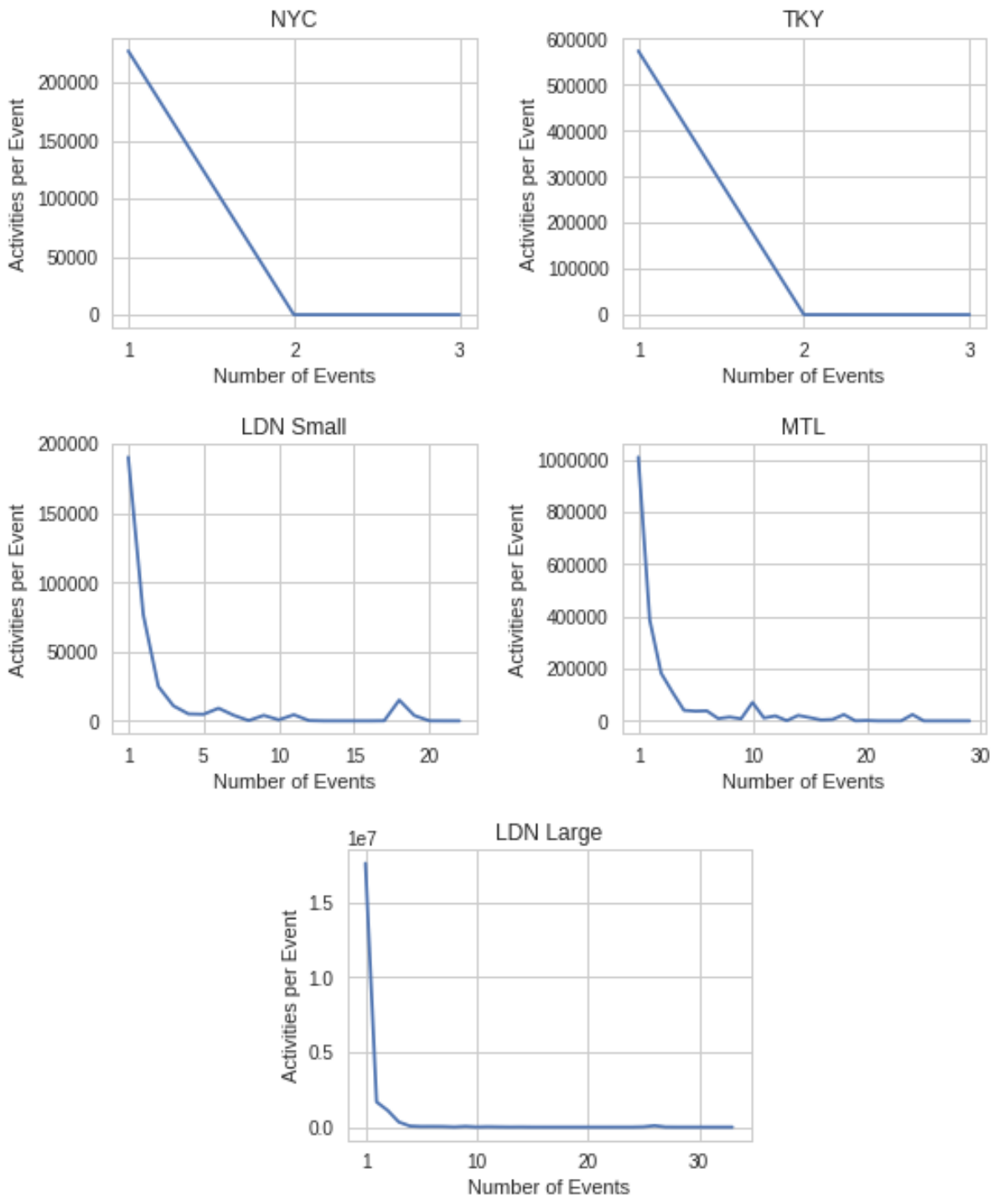

Figure 4.3: Number of activities per event

Table 4.3: Experiment 1 Results

Model	Accuracy	RMSE	MAE
No Inactive Users filter	0.95	0.09	0.087
No Events Merger	0.89	0.17	0.032
No External Context	0.96	0.08	0.0083
No Clusterer	0.91	0.09	0.0082
UCAP	0.97	0.08	0.0079

meant that any user who performed five or more events was considered active.

The data were ordered chronologically, the first 80% of the data were used as the training dataset, and the last 20% were used as the test dataset. The *Data Splitter* separates the training dataset into a training dataset and a validation dataset during the training process. The test dataset is used only for the evaluation.

To compare the performance of the models, the following metrics were used: (a) *Accuracy*, used by the *Evaluator component*; (b) the Root Mean Square Error (*RMSE*), according to Eq. (4.2); and (c) the Mean Absolute Error (*MAE*), according to Eq. (4.3).

For *Accuracy*, the higher the number, the better, and 1 is the highest score. For *RMSE* (Eq. (4.2)), the lower the score, the better, and 0 is the best possible value. In this equation, A is the dataset with all the ground truths, and $\hat{A}$ is the dataset with the predictions, $a \in A$, $\hat{a} \in \hat{A}$, and $n = |A| = |\hat{A}|$. The *MAE* (Eq. (4.3)) follows the same logic as *RMSE*: the lower the score, the better. Both equations used the same parameters.

$$RMSE(A, \hat{A}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(a_i - \hat{a}_i)^2} \tag{4.2}$$

$$MAE(A, \hat{A}) = \frac{1}{n}\sum_{i=0}^{n-1} abs(a_i - \hat{a}_i) \tag{4.3}$$

Figure 4.4 shows the results of this experiment, which are summarized in Table 4.3.

The UCAP framework with all the components and features presented a better performance than the other models with the missing components in this experiment. It is clear that the *Events Merger* is crucial to the result, as would be for a dataset based on *GPS* data, with a 9.8%

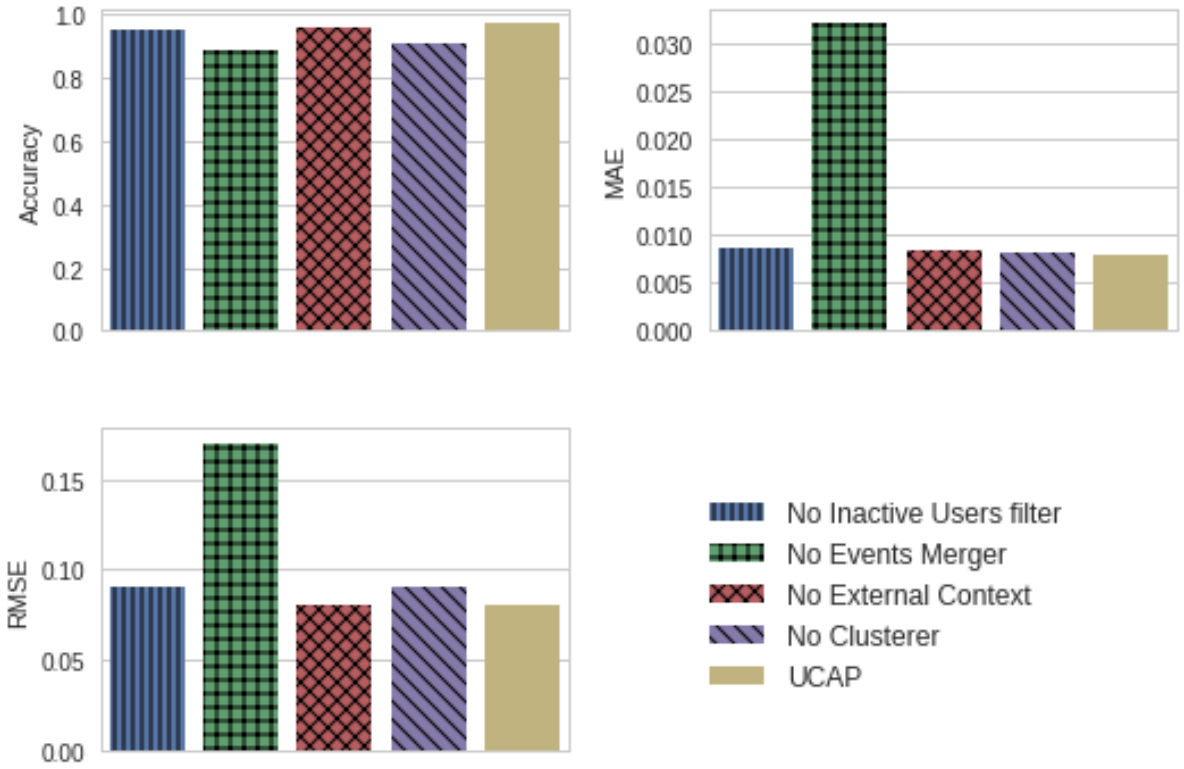

Figure 4.4: UCAP variations prediction results

improvement in *Accuracy*. Other components also provided relatively smaller improvements to the result, which gives evidence of their importance to the proposed approach.

4.1.3 Experiment 2: Comparison with a Different Approach

The objective of this experiment was to compare the proposed framework with a current technique that models the same problem to evaluate its effectiveness. Two datasets from [2] were used, which were the *LBSN* datasets described in Table 4.2. The UCAP framework was compared with their model, called *STAP*, by replicating their experimental set.

The value used for the parameter *m* from the *Inactive Users Filter* was 3, and the time window was one week, meaning that users were considered active if they generated at least three events per week. The parameters used in the XGBoost model were the same as those in *Experiment 1*, as shown in Table 4.1.

The data were ordered chronologically; the first eight months were considered as the train-

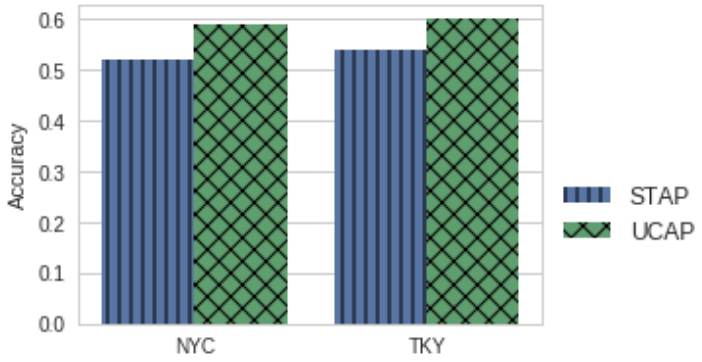

Figure 4.5: UCAP and *STAP* comparison

Table 4.4: Experiment 2 Results

Model	**NYC**	**TKY**
STAP [2]	0.52	0.54
UCAP	0.59	0.60

ing dataset, the ninth month was used as the validation dataset, and the tenth month was used as the test dataset. The test dataset is used only for the evaluation.

To compare the performance of the models, the *Accuracy* measure was used. Table 4.4 and Fig.4.5 summarize the results of this experiment. The framework proposed here outperformed the STAP model by 13.5% on accuracy on the NYC dataset and 11.1% on the TKY, or 12.3% better by averaging these two values.

4.1.4 Experiment 3: Comparison between Different Datasets

The objective of this experiment was to compare the performance of the proposed framework when applied to different datasets. This experiment used all five datasets described in Table 4.2. It is good to highlight the *LDN Large* dataset, which could be considered *Big Data* concerning its volume because the number of entries is large enough that the dataset does not fit entirely in memory [46]. To address the size issue, Dask [47] was used in the *Preprocessing Layer*.

For the datasets used in the previous experiments (*LDN Small*, *NYC*, and *TKY*), the entire set-up was kept the same. For the other two datasets, the *MTL* dataset and the *LDN Large* dataset, the value used for *m* from the *Inactive Users Filter* is 5, and all the data were placed in

Table 4.5: Experiment 3 Results

Dataset	Accuracy	RMSE	MAE
LDN Small	0.96	0.08	0.0079
LDN Large	0.93	0.11	0.026
MTL	0.98	0.08	0.0088
NYC	0.59	0.06	0.0079
TKY	0.60	0.06	0.008

the same time window, as in *Experiment 1*. The data were ordered chronologically, and the first 80% of the data were used as the training dataset and the last 20% as the test dataset. The *Data Splitter* separates the training dataset into a training dataset and a validation dataset during the training process. The test dataset is used only for the evaluation.

To compare model performance, *Accuracy*, *RMSE* (Eq. (4.2)) and *MAE* (Eq. (4.3)) were used. The results of this experiment are shown in Fig. 4.6 and summarized in Table 4.5. The results were very similar among datasets with the same data source type.

The *LDN Large* dataset presented a similar $RMSE$ but a higher MAE, which means the errors are larger, but the variance is smaller. This could be related to the bias-variance trade-off [48]. To avoid overfitting, the *Prediction Engine* ended up with a relatively simpler model when compared with the other datasets. Even though the *LDN Large* dataset is considerably larger than the other datasets, the model provided similar accuracy, which is a positive characteristic of the chosen algorithm.

4.2 Discussion

An essential component of the UCAP framework is the *Events Merger*. In Fig. 4.3, the plots from the LBSN datasets (*NYC* and *TKY* datasets) show that in these datasets, there was only one activity per recorded event, whereas the others presented a different distribution. An event with more than one activity is evidence of *GPS uncertainty* because it is not sure which of the activities the user was performing at that moment. In the *LDN Large* dataset, this affected a small fraction of the data, but in both *LDN Small* and *MTL* datasets, more events presented

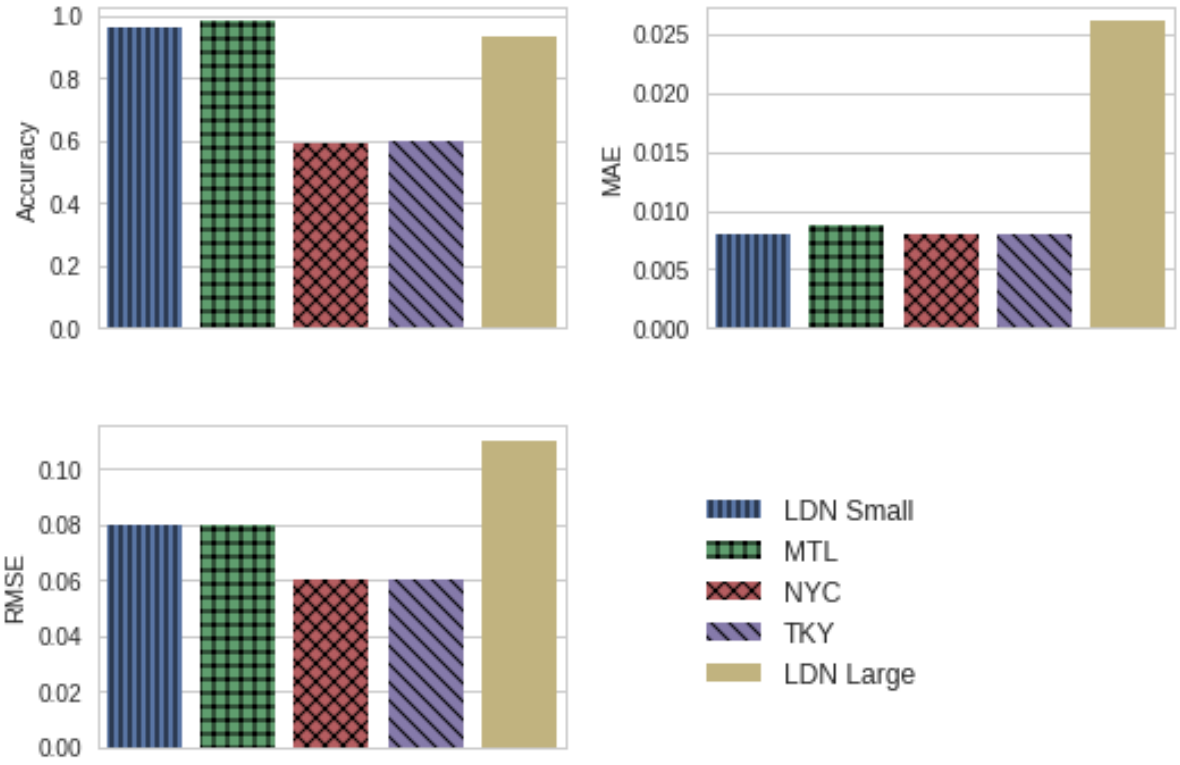

Figure 4.6: UCAP prediction results on different datasets

this characteristic. In the author's view, this makes the training process much harder. Usually, this happens in commercial areas with a large number of venues. By using a probability vector to represent activities and merging the locations under a new activity, the predictor results can be improved. The *Events Merger* contributed to a noticeable improvement in the evaluation result. Furthermore, the *Clusterer* also improved the results of the proposed framework, which may indicate that the *Preprocessing Layer* is on the right track in preparing the data for any modelling performed later in the process.

Moreover, the comparison with both *STAP* and different datasets could be considered as evidence that this work is not biased to a specific type or size of dataset. Finally, because the *STAP* approach is different from the one presented here, it is impossible to evaluate if the difference in the results is related to a group of specific UCAP components. It is good to point out that *STAP* was compared to several baselines[2], outperforming all of them. The decision to compare the UCAP framework only with *STAP* reflects the fact that it was the most closely

approach found.

The framework presented here would be useful for any location-based service that displays venues or activities for users because it could highlight what the user is willing to do, instead of a list of things to do in the vicinity based on distance. It could be helpful to more specific applications as well, enhancing their usability. For example, a grocery shopping list could notify the user about the list itself only when the user is within a context he/she is willing to buy groceries, instead of doing it every time the user is near a grocery store, without any set-up.

The author asserts that the main strengths of this framework are in the *Preprocessing Layer* and the *Feature Engineering Layer* and that these layers could be used in a set of related problems by changing the features used and the target modelled by the *Prediction Engine*. The UCAP is flexible enough to allow variations to be implemented with few changes according to the desired goal.

4.3 Summary

This chapter has presented an evaluation of the framework described in Chapter 3. Moreover, the implementation details of the UCAP framework used in the experiments as well as the results of the three case studies were discussed. Furthermore, the datasets that were used were analyzed and discussed. In the first case study, variations of UCAP were compared to evaluate the impact of different components on the final results. In the second case study, UCAP was compared with a state-of-the-art model. The third case study compared the performance of the UCAP framework with datasets of different sizes and sources. The results showed that UCAP preprocessing improves the accuracy of the trained model. In addition, UCAP outperforms a state-of-the-art model. Finally, it does not lose accuracy with larger datasets.

Chapter 5

Conclusions and Future Work

This chapter presents a concluding summary based on the contributions of the proposed framework for modelling *users' current activity preferences* (UCAP). In addition, a description of possible future research involving UCAP and its components is provided.

5.1 Conclusions

This thesis has presented UCAP, which is a framework for predicting a user's current activity preferences based on historical event patterns and current context, including time and location. This framework is suitable both for data gathered by user interaction, such as check-ins, and for location data collected by background processes. Moreover, an approach to solve the GPS uncertainty issue was also presented and incorporated into the framework. This was achieved through the *Events Merger*, a data aggregation component. An overview of the architecture has been presented, including its components, their roles, and their relationships.

To demonstrate the applicability of UCAP, this thesis presented three case studies using five real-world datasets. The first case study showed the impact of UCAP components by removing individual components and evaluating the performance loss compared to the complete framework. The second case study compared UCAP with a state-of-the-art approach. The third case study evaluated the framework's performance on different datasets. The results supported the

assertion that all the components are relevant to UCAP's performance. They also showed that UCAP outperforms the state-of-the-art modelling technique by 12.3% on average. Finally, the results show that UCAP is not limited to a specific dataset.

5.2 Future Work

The scope of this thesis took into consideration the limited time available for its development. Because of this, many aspects were left open and can be explored in future work.

The UCAP implementation presented in this thesis used a gradient boosting technique as the *Prediction Engine*. It would be possible to use other machine learning techniques in its place. For example, it would be intesting to evaluate the use of deep learning, adapting the surrounding components as necessary and comparing the results to the work presented in this thesis.

The same can be done with the *Clusterer* and the *Tuner*. For the *Clusterer*, instead of using the *K-Means*, it would be interesting to evaluate the performance of other clustering algorithms and clustering approaches, such as fuzzy clustering, as the users could belong to more than one cluster at the same time, with a different weight. Relating to the *Tuner*, a gradient-based or Bayesian approach could be taken instead of the grid-search used.

The *Preprocessing Layer* can be enhanced by using other components. For instance, a component could check for inconsistencies in the data regarding how distant subsequent events are in time and space: events with several kilometres of distance in a matter of seconds should be removed from the dataset. Evaluate the time performance of the framework, studying if there are relations between time and dataset size, and time and number of features. This could indicate whether UCAP would benefit from feature or instance selection techniques.

Privacy and security challenges were left out of the scope of this thesis. However, they certainly must be considered. Access to the datasets and the trained models should be restricted. Communication between the application and the framework must also be performed through

secure means.

The UCAP implementation evaluated in this thesis used only historical data. Extensions would be necessary to use this framework in a production environment. Production data come as a continuous stream that never stops being gathered, bringing several new challenges. For example, a new layer could be created to handle the interface with the input data stream. There are also clustering techniques on streaming data that could be evaluated. In this way, the UCAP framework could be used as presented in this thesis. Moreover, it would be necessary to decide when a new model should be trained. Furthermore, the amount of data that would be stored would have to be evaluated.

In this thesis, the problem explored was predicting users' current activity preferences. The framework can be extended to cope with other business objectives related to location data. For example, it could predict the user's next location. Moreover, the prediction layer could be leveraged to perform recommendation tasks.

Bibliography

[1] Y. Zheng, "Tutorial on location-based social networks," in *Proceedings of International Conference on World Wide Web*, May 2012.

[2] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, pp. 129–142, Jan 2015.

[3] J. Ye, Z. Zhu, and H. Cheng, "What's your next move: User activity prediction in location-based social networks," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 171–179, 2013.

[4] T. H. Silva, P. O. V. de Melo, J. M. Almeida, M. Musolesi, and A. A. Loureiro, "A large-scale study of cultural differences using urban data about eating and drinking preferences," *Information Systems*, vol. 72, pp. 95 – 116, 2017.

[5] V. Kounev, "Where will i go next?: Predicting future categorical check-ins in location based social networks," in *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pp. 605–610, Oct 2012.

[6] J. Cao, S. Xu, X. Zhu, R. Lv, and B. Liu, "Efficient fine-grained location prediction based on user mobility pattern in lbsns," in *2017 5th International Conference on Advanced Cloud and Big Data (CBD)*, pp. 238–243, Aug 2017.

[7] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "Nextplace: A spatio-temporal prediction framework for pervasive systems," in *Pervasive Computing* (K. Lyons, J. Hightower, and E. M. Huang, eds.), (Berlin, Heidelberg), pp. 152–169, Springer Berlin Heidelberg, 2011.

[8] A. Roy and E. Pebesma, "A machine learning approach to demographic prediction using geohashes," in *Proceedings of the 2nd International Workshop on Social Sensing*, SocialSens'17, (New York, NY, USA), pp. 15–20, ACM, 2017.

[9] J. Sang, T. Mei, and C. Xu, "Activity sensor: Check-in usage mining for local recommendation," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, pp. 41:1–41:24, Apr 2015.

[10] G. McKenzie and K. Janowicz, "Where is also about time: A location-distortion model to improve reverse geocoding using behavior-driven temporal semantic signatures," *Computers, Environment and Urban Systems*, vol. 54, pp. 1 – 13, 2015.

[11] D. Pyle, *Data preparation for data mining*, vol. 1. Morgan Kaufmann, 1999.

[12] J. Van Hulse, *Data Quality in Data Mining and Machine Learning*. PhD thesis, Boca Raton, FL, USA, 2007.

[13] H. Wang and S. Wang, "Mining incomplete survey data through classification," *Knowledge and Information Systems*, vol. 24, no. 2, pp. 221–233, 2010.

[14] J. Luengo, S. García, and F. Herrera, "On the choice of the best imputation methods for missing values considering three groups of classification methods," *Knowledge and Information Systems*, vol. 32, no. 1, pp. 77–108, 2012.

[15] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*, vol. 333. John Wiley & Sons, 2014.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[17] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[18] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[19] H. Liu and H. Motoda, *Instance selection and construction for data mining*. Springer US, 2001.

[20] S. Garca, J. Luengo, J. A. Sez, V. Lpez, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 734–750, April 2013.

[21] T. Fushiki, "Estimation of prediction error by using k-fold cross-validation," *Statistics and Computing*, vol. 21, pp. 137–146, Apr 2011.

[22] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[23] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence: A Modern Approach*, vol. 2. Prentice Hall Upper Saddle River, 2003.

[24] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, (Berkeley, Calif.), pp. 281–297, University of California Press, 1967.

[25] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[26] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[27] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[28] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "Mining user mobility features for next place prediction in location-based services," in *2012 IEEE 12th International Conference on Data Mining*, pp. 1038–1043, Dec 2012.

[29] D. Preoţiuc-Pietro and T. Cohn, "Mining user behaviours: A study of check-in patterns in location based social networks," in *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13, (New York, NY, USA), pp. 306–315, ACM, 2013.

[30] R. Trasarti, R. Guidotti, A. Monreale, and F. Giannotti, "Myway: Location prediction via mobility profiling," *Information Systems*, vol. 64, pp. 350 – 367, 2017.

[31] B. T. Nguyen, N. V. Nguyen, N. T. Nguyen, and M. H. T. Tran, "A potential approach for mobility prediction using gps data," in *2017 7th International Conference on Information Science and Technology (ICIST)*, pp. 45–50, Apr 2017.

[32] M. Chen, X. Yu, and Y. Liu, "Mining moving patterns for predicting next location," *Information Systems*, vol. 54, pp. 156 – 168, 2015.

[33] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 194–200, AAAI Press, 2016.

[34] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, "Big data driven hidden markov model based individual mobility prediction at points of interest," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 5204–5216, Jun 2017.

[35] Z. Yao, "Exploiting human mobility patterns for point-of-interest recommendation," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, WSDM '18, (New York, NY, USA), pp. 757–758, ACM, 2018.

[36] L. Yao, Q. Z. Sheng, X. Wang, W. E. Zhang, and Y. Qin, "Collaborative location recommendation by integrating multi-dimensional contextual information," *ACM Transactions on Internet Technology*, vol. 18, pp. 32:1–32:24, Feb 2018.

[37] M. He, W. Gu, and Y. Kong, "Group recommendation: By mining users' check-in behaviors," in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17, (New York, NY, USA), pp. 65–68, ACM, 2017.

[38] J. Capdevila, M. Arias, and A. Arratia, "Geosrs: A hybrid social recommender system for geolocated data," *Information Systems*, vol. 57, pp. 111 – 128, 2016.

[39] D. Lian, X. Xie, V. W. Zheng, N. J. Yuan, F. Zhang, and E. Chen, "Cepr: A collaborative exploration and periodically returning model for location prediction," *ACM Transactions on Internet Technology*, vol. 6, pp. 8:1–8:27, Apr 2015.

[40] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[41] Y. Liu, T.-A. N. Pham, G. Cong, and Q. Yuan, "An experimental evaluation of point-of-interest recommendation in location-based social networks," *Proceedings of the VLDB Endowment*, vol. 10, pp. 1010–1021, Jun 2017.

[42] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[43] 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*, vol. 12, pp. 2825–2830, 2011.

[44] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, (New York, NY, USA), pp. 1232–1237, ACM, 2004.

[45] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016.

[46] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[47] Dask Development Team, *Dask: Library for dynamic task scheduling*, 2016.

[48] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

Curriculum Vitae

Name:	Roberto Barboza Junior
Post-Secondary Education and Degrees	The University of Western Ontario London, ON 2016 - 2018 MESc
	University of Campinas Campinas, SP - Brazil 2004-2008 BSc Computer Engineering
Related Work Experience:	Teaching Assistant The University of Western Ontario 2016 - 2018
	Field Application Engineer Arrow Electronics (Brazil) 2014 - 2016
	Software Developer Embraer Defense (Brazil) 2011 - 2014
	Digital Systems Engineer Idea! Electronics (Brazil) 2009 - 2011