République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER DE BISKRA
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique

N°d'ordre :…….
N°de Série :……

Thèse
En vue de l'obtention du diplôme de

Doctorat en Sciences en Informatique

Présentée par:
M. Abdessamed SASSI

THEME :

Vers des services Internet basés sur les profils de mobilité des utilisateurs

Soutenue publiquement le : 06/03/2022, devant le jury composé de :

M. Noureddine Djedi, Professeur, Université de Biskra, Président
M. Abdelmadjid Bouabdallah, Professeur, Université de Compiègne, France, Examinateur
M. Yacine Challal, Professeur, Ecole Supérieure d'Informatique, Alger, Examinateur
M. Abdelhamid Djeffal, Professeur, Université de Biskra, Examinateur
M. Walid Bechkit, Maitre des conférence, INSA-Lyon, France, Co Directeur de thèse
M. Abdelmalik Bachir, Professeur, Université de Biskra, Directeur de thèse,

ABSTRACT

Nowadays, mobility prediction models play an important role in many location-based services, such as food delivery, transportation planning, and advertisement posting. Most previous studies on predicting mobility have worked on computer generated data and focused on mathematical modeling principally due to the lack of a real mobility data. Such studies have limited ability to capture human mobility accurately. However, with the democratization of mobility data and the availability of large data sets, numerous research activities turned toward predicting mobility based on examining real mobility data traces with the aim of building realistic models that can capture and understand human's mobility behaviors as well as making accurate mobility prediction.
In this thesis, we present the methods we proposed to predict spatial and temporal behaviors of mobile users. Our first work focuses on predicting the next location of mobile users by analyzing large data sets of the history of their movements. We make use of past location sequences, also called location history, to train a classification model that will be used to predict future locations. Contrary to traditional mobility prediction techniques based on Markovian models, we investigate the use of modern deep learning techniques such as the use of Convolutional Neural Networks (CNNs). Inspired by the word2vec embedding technique used for the next word prediction, we present a new method called loc2vec in which each location is encoded as a vector whereby the more often two locations cooccur in the location sequences, the closer their vectors will be. Using the vector representation, we divide long mobility sequences into several sub-sequences and use them to form Mobility Subsequence Matrices on which we run CNN classification which will be used later for the prediction. We run extensive testing and experimentation on a subset of a large real mobility trace database made publicly available through the CRAWDAD project. Our results show that loc2vec embedding and CNN-based prediction provide significant improvement in the next location prediction accuracy compared to state-of-the-art methods. We also show that transfer learning on existing pre-trained CNN models provides further improvement over CNN models build from scratch on mobility data. We also show that our loc2vec-CNN model enhanced with transfer learning achieves better results than other variants including our

other proposal onehot-CNN and existing Markovian models.

In the second work, we focus on predicting the temporal behavior, particularly the residence time, of mobile users at their relevant locations. In this work, we explored the joint use of location history, arrival time, and the previous residence time to accurately predict the residence time at the current location. We developed a model that integrates all these parameters and uses our modified Moving-Average and CDF time-aided algorithms that include the arrival time in the model. We run performance evaluation experiments on a subset of the same mobility trace collected by Dartmouth College. Our results show that adding high-granularity temporal information to the mobility model allows to significantly improve the residence time prediction compared to state-of-the-art methods. The prediction accuracy improvement for the dataset we work on has been consistent and of about 20% on the average.

We also presented two linear mobility models for residence time prediction, namely Linear Regression (LR), and Auto-Regression (AR). We run performance evaluation experiments on two different WiFi mobility traces datasets made available through the CRAWDAD project. Our results show that using linear regression-based learning algorithms significantly improve the residence time prediction accuracy compared to state-of-the-art methods, and achieve prediction errors in the order of seconds and minutes for a large number of users.

Keywords: *Location Prediction*, *Time Prediction*, *Location Embedding*, *Convolutional Neural Networks*, *WiFi Mobility Traces*.

Résumé

De nos jours, les modèles de prédiction de la mobilité jouent un rôle important dans de nombreux services basés sur la localisation, tels que la livraison de nourriture, la planification du transport et la publication d'annonces. La plupart des études précédentes sur la prédiction de la mobilité ont travaillé sur des données générées par ordinateur et se sont concentrées sur la modélisation mathématique principalement en raison du manque de données de mobilité réelles. Ces études ont une capacité limitée à capturer avec précision la mobilité humaine. Cependant, avec la démocratisation des données de mobilité et la disponibilité de grands ensembles de données, de nombreuses activités de recherche se sont orientées vers la prédiction de la mobilité basée sur l'analyse de traces de données de mobilité réelles dans le but de construire des modèles réalistes capables de capturer et de comprendre les comportements de mobilité humaine et aussi faire une prédiction précise de mobilité.

Dans cette thèse, nous présentons les méthodes que nous avons proposées pour prédire les comportements spatiaux et temporels des utilisateurs mobiles. Notre premier travail se concentre sur la prédiction du prochain emplacement des utilisateurs mobiles en analysant de grands ensembles de données de l'historique de leurs déplacements. Nous utilisons des séquences des emplacements visités dans le passé, également appelées historique des emplacements, pour former un modèle de classification qui sera utilisé pour prédire les futures emplacements. Contrairement aux techniques traditionnelles de prédiction de la mobilité basées sur les modèles markoviens, nous étudions l'utilisation de techniques modernes d'apprentissage en profondeur telles que l'utilisation de réseaux de neurones convolutifs (CNN). Inspiré par la technique d'intégration word2vec utilisée pour la prédiction du mot suivant, nous présentons une nouvelle méthode appelée loc2vec dans laquelle chaque emplacement est codé en tant que vecteur, de sorte que plus deux emplacements coexistent dans les séquences d'emplacements, plus leurs vecteurs seront proches. En utilisant la représentation vectorielle, nous divisons les longues séquences de mobilité en plusieurs sous-séquences et les utilisons pour former des matrices de sous-séquences de mobilité sur lesquelles nous exécutons la classification CNN qui sera utilisée plus tard pour la prédiction. Nous effectuons des tests et des expérimentations approfondis sur un sous-ensemble d'une grande base de données

de traces de mobilité réelle rendue publique via le projet CRAWDAD. Nos résultats montrent que l'intégration loc2vec et la prédiction basée sur CNN améliorent considérablement la précision de la prédiction du prochain emplacement par rapport aux méthodes d'état de l'art. Nous montrons également que le transfert d'apprentissage des modèles CNN pré-entranés existants fournit une amélioration significative par rapport aux modèles CNN construits à partir de zéro sur les données de mobilité. Nous montrons également que notre modèle loc2vec-CNN amélioré avec l'apprentissage par transfert obtient de meilleurs résultats que d'autres variantes, y compris notre autre proposition onehot-CNN et les modèles markoviens existants.

Dans le deuxième travail, nous nous concentrons sur la prédiction du comportement temporel, en particulier le temps de résidence, des utilisateurs mobiles à leurs emplacements pertinents. Dans ce travail, nous avons exploré l'utilisation conjointe de l'historique d'emplacement, de l'heure d'arrivée et du temps de résidence précédent pour prédire avec précision le temps de résidence à l'emplacement actuel. Nous avons développé un modèle qui intègre tous ces paramètres et utilise nos algorithmes Moving-Average et CDF modifiés et aidés par le temps qui incluent l'heure d'arrivée dans le modèle. Nous menons des expériences d'évaluation des performances sur un sous-ensemble de la même trace de mobilité collectée par le Dartmouth College. Nos résultats montrent que l'ajout d'informations temporelles de haute granularité au modèle de mobilité permet d'améliorer considérablement la prédiction du temps de résidence par rapport aux méthodes d'etat de l'art. L'amélioration de la précision des prédictions pour l'ensemble de données sur lequel nous travaillons a été cohérente et d'environ 20% en moyenne.

Nous avons également présenté deux modèles de mobilité linéaire pour la prédiction du temps de résidence, appelés la régression linéaire (LR) et l'auto-régression (AR). Nous menons des expériences d'évaluation des performances sur deux différents ensembles de données de traces de mobilité WiFi mis à disposition via le projet CRAWDAD. Nos résultats montrent que l'utilisation d'algorithmes d'apprentissage basés sur la régression linéaire améliore considérablement la précision de la prédiction du temps de résidence par rapport aux méthodes d'etat de l'art et permet d'obtenir des erreurs de prédiction de l'ordre de quelques secondes et minutes pour un grand nombre d'utilisateurs.

Mots clés: *Prédiction de l'Emplacement*, *Prediction de Temps*, *Intégration de l'Emplacement*,

Réseaux de Neurones Convolutifs, *Traces de Mobilité WiFi*.

الملخص

في الوقت الحاضر، تلعب نماذج التنبؤ الحركية دورًا مهمًا في العديد من الخدمات المرتكزة على الموقع، مثل توصيل طلبات الطعام ، وتخطيط النقل ونشر الإعلانات.

عملت معظم الدراسات السابقة حول التنبؤ الحركي على البيانات التي تم إنشاؤها بواسطة الكمبيوتر وركزت على النمذجة الرياضية بشكل أساسي بسبب نقص بيانات التنقل الحقيقية. مثل هذه الدراسات لديها قدرة محدودة لالتقاط تنقل الإنسان بدقة. ومع ذلك ، مع دمقرطة بيانات التنقل وتوافر مجموعات البيانات الضخمة، تحولت العديد من الأنشطة البحثية نحو تنبؤ التنقل المبنيً على فحص تتبع بيانات تنقل حقيقية بهدف بناء نماذج واقعية يمكنها التقاط وفهم سلوكيات تنقل الإنسان وكذلك جعل تنبؤ التنقل دقيق.

في هذه الأطروحة ، نقدم الطرق التي اقترحناها للتنبؤ بالسلوكيات المكانية والزمانية لمستخدمي الهواتف المحمولة. يركز عملنا الأول على التنبؤ بالموقع التالي لمستخدمي الهاتف المحمول من خلال تحليل مجموعات البيانات الكبيرة لتاريخ تحركاتهم. نحن نستخدم تسلسلات المواقع السابقة ، والتي تسمى أيضًا سجل الموقع ، لتدريب نموذج تصنيف الذي سيتم استخدامه للتنبؤ بالمواقع المستقبلية.

على عكس تقنيات التنبؤ التنقل التقليدية القائمة على نماذج ماركوفيان ، فإننا نتحرى عن استخدام تقنيات التعلم العميق الحديثة مثل استخدام الشبكات العصبية التلافيفية (CNNs). نقدم طريقة جديدة تسمى loc2vec مستوحاة من تقنية تضمين word2vec المستخدمة للتنبؤ بالكلمة التالية، يتم فيها ترميز كل موقع كمتجه حيث كلما زاد ظهور موقعين معا في تسلسل المواقع، كلما اقتربت متجهاتهما.

باستخدام التمثيل بالمتجه، نقسم تسلسلات التنقل الطويلة إلى عدة تسلسلات فرعية ونستخدمها لتشكيل مصفوفات تسلسلات التنقل التي نقوم بتشغيل مصنف CNN عليها والذي سيتم استخدامه لاحقًا للتنبؤ.

نجري اختبارات و تجارب مكثفة على جزء من قاعدة بيانات كبيرة لتتبع التنقل الحقيقي و المتاحة للعامة من خلال مشروع Crawdad. تظهر نتائجنا أن تضمين Loc2vec و التنبؤ القائم على شبكة CNN يوفر تحسنا كبيرا في دقة التنبؤ بالموقع التالي مقارنة بالطرق الحديثة. نظهر أيضا أن نقل التعلم على نماذج CNN الحالية والمدربة مسبقا يوفر مزيدا من التحسين في دقة التنبؤ على نماذج CNN المبنية من نقطة الصفر على بيانات التنقل. نظهر أيضًا أن نموذجنا loc2vec-CNN المحسن بنقل التعلم يحقق نتائج أفضل من المتغيرات الأخرى بما في ذلك اقتراحنا الآخر onehot-CNN ونماذج Markovian الحالية.

في العمل الثاني، نركز على تنبؤ السلوك الزمني ، وخاصة وقت الإقامة ، لمستخدمي الهاتف المحمول في مواقعهم الشعبية. في هذا العمل ، استكشفنا الاستخدام المشترك لسجل الموقع، وقت الوصول، ووقت الإقامة السابق للتنبؤ بدقة بوقت الإقامة في الموقع الحالي.

لقد طورنا نموذجًا يدمج كل هذه العوامل ويستخدم خوارزمياتنا المعدلة MA-AT وCDF-AT والتي تتضمن وقت الوصول في النموذج.

نجري تجارب تقييم الأداء على مجموعة فرعية من نفس مجموعة بيانات تتبع التنقل التي تم جمعها بواسطة كلية دارتموث. تظهر نتائجنا أن إضافة معلومات زمنية عالية الدقة لنموذج التنقل يسمح بشكل كبير بتحسين التنبؤ بوقت الإقامة مقارنة بالطرق الحالية.

تحسين دقة التنبؤ لمجموعة البيانات التي نعمل عليها كان ثابتًا وحوالي 20٪ في المتوسط.

قدمنا أيضًا نوعين من نماذج تنقل خطية للتنبؤ بوقت الإقامة ، وهما الانحدار الخطي (LR) والانحدار التلقائي (AR). نجري تجارب تقييم الأداء على مجموعتين مختلفتين من مجموعات بيانات تتبع التنقل WiFi المتاحة من خلال مشروع CRAWDAD.

تظهر نتائجنا أن استخدام خوارزميات التعلم القائم على الانحدار الخطي تحسن بشكل كبير دقة التنبؤ بوقت الإقامة مقارنة بالطرق الحالية ، وتحقق أخطاء التنبؤ بترتيب الثواني والدقائق لعدد كبير من المستخدمين.

الكلمات المفتاحية :تنبؤ الموقع، تنبؤ الوقت، تضمين الموقع ، الشبكات العصبية التلافيفية، آثار تنقل WIFI

LIST OF PUBLICATIONS

The following listed are the publications related to this thesis.

Published Journal Papers

1. Abdessamed Sassi, Abdelmalik Bachir, Walid Bechkit. Evaluating Regression Models for Temporal Prediction of Wi-Fi Device Mobility. Wireless Personal Communications, vol. 116 , pp. 2169-2186, 2020.

Published Conference Papers

1. Abdessamed Sassi, Salah Eddine Henouda, Abdelmalik Bachir. (2017), On predicting the residence time of mobile users at relevant places, in Networks, Computers and Communications (ISNCC), 2017 International Symposium on, IEEE, pp. 16.

2. Abdessamed Sassi, Mohammed Brahimi, Walid Bechkit, Abdelmalik Bachir. Location embedding and deep convolutional neural networks for next location prediction, in 2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium), IEEE, pp. 149157.

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, I wish to express my sincere appreciation and thanks to my supervisor **Prof. Abdelmalik Bachir** who has provided me all the detailed reviews, guidance and direction throughout the research, without his motivation and continuous encouragement, this research would not have been successfully completed.

I would also like to thank my co-supervisor, **Dr. Walid Bechkit** (INSA Lyon, France), for his helpful suggestions and ideas for pursuing my research.

I would like to extend my heartfelt gratitude to my colleagues at CITI Research Laboratory (INSA Lyon, France) for all the good times and collaboration.

Last but not the least, I would like to thank my mother and my wife for their constant encouragement and moral support along with patience and understanding.

TABLE OF CONTENTS

ABSTRACT i

Resumé iii

LIST OF PUBLICATIONS viii

ACKNOWLEDGEMENT ix

LIST OF FIGURES xiii

LIST OF TABLES xv

LIST OF TERMS AND ABBREVIATIONS xv

List of Terms and Abbreviations **xvi**

1 Introduction **1**

1.1 Motivation 2

1.1.1 Predicting the next location 2

1.1.2 Predicting the residence time 2

1.2 Problem Statement 2

1.3 Contributions 3

1.4 Outline of thesis 5

2 Background **6**

2.1 Location 6

2.2 Data collection 6

2.3 Location Prediction 7

2.4 Prediction algorithms 8

2.4.1 Markov Predictors 8

2.4.2 Moving Average Predictor 10

2.4.3 CDF Predictor 10

2.4.4 Linear Regression-Based Prediction 10

2.4.5 Auto Regression-Based Prediction 12
2.5 Minimizing Squared Error 14
2.5.1 Using Gradient Descent 14
2.5.2 Minimizing Square Error with Normal Equation 22
2.6 Prediction metrics 22
2.6.1 Next-location prediction 22
2.6.2 Time prediction 23
2.7 Conclusion 24

3 Next Location Prediction 25
3.1 Introduction 25
3.2 Related Work 28
3.3 Proposed Approach 31
3.3.1 Embedding Methods for Location Representations 33
3.3.2 Convolutional Neural Network for Next Location Prediction . . . 37
3.3.3 CNN layers 38
3.3.4 CNN Training 40
3.4 Performance Evaluation 43
3.4.1 Effect of loc2vec Parameters Choice 43
3.4.2 Evaluating the Performance of CNN-based Predictions 45
3.5 Conclusions 49

4 Residence Time Prediction 50
4.1 Introduction 50
4.2 Related Work 51
4.3 First System Model 55
4.3.1 An illustrative Example 57
4.3.2 Residence Time Prediction 58
4.3.3 Results 61
4.4 Second System Model 65
4.4.1 Regression-Based Residence Time Prediction 65
4.4.2 Minimizing Squared Error 70
4.4.3 Applying LR and AR on Selected Features 71

4.4.4 Results 72
4.5 Performance Evaluation 80
4.5.1 Evaluation Dataset 80
4.5.2 Eliminating the Ping-Pong Effect 81
4.6 Conclusions 82

5 Conclusion and future research directions **84**

REFERENCES 85

LIST OF FIGURES

2.1 The best fitting straight line after getting the optimal parameter $\hat{\boldsymbol{\theta}}$. . . . 12

2.2 Optimization of the function $h_{\boldsymbol{\theta_1}}$ by applying the gradient descent to the function $J(\boldsymbol{\theta_1})$ 19

2.3 Optimization of the function $h_{\boldsymbol{\theta_0},\boldsymbol{\theta_1}}$ by applying the gradient descent to the function $J(\boldsymbol{\theta_0},\boldsymbol{\theta_1})$ 20

2.4 The Learning rate λ 21

3.1 Overview of next location prediction based on CNN. 32

3.2 Architecture of the training loc2vec neural network. 36

3.3 Convolutional Neural Network (CNN) 38

3.4 Convolution layer. 38

3.5 ReLU activation function. 39

3.6 Max-pooling layer. 41

3.7 Comparison of the three variants of loc2vec-CNN model according to the average accuracy metric with various subsequence lengths s and embedding vector sizes m. 45

3.8 Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov, according to the average accuracy metric. 47

3.9 Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov according to the accuracy metric for every user. 47

3.10 Effect of pre-training on the performance of loc2vec-CNN. 48

4.1 Comparison of the four predictors according to the time difference metric. The k-CDF and k-moving have been first proposed in Song, Deshpande, Kozat, Kotz and Jain (2006). 61

4.2 Comparison of the four predictors according to the time difference metric. The values used for the context length k is 5 and the value used for the desired confidence p is 0.8. 62

4.3 Comparison of the two k-moving average and k-moving-average-AT predictors according to various context lengths ($k = 5$ and $k = 10$ . . . 63
4.4 Comparison of the two k-CDF and k-CDF-AT predictors according to various desired confidence values ($p = 0.2$ and $p = 0.8$) 64
4.5 The best fitting straight line after getting the optimal parameter $\hat{\boldsymbol{\alpha}}$. . . . 68
4.6 Cumulative Distribution of the fraction of users according to the prediction error achieved on them for the Dartmouth WiFi dataset and the Ile Sans Fils dataset. 73
4.7 Comparison of Autoregression, Linear Regression, NonLinear and Moving-Average models for the two datasets on different relevant places of a sample user's history. 75
4.8 Prediction Error variation for a sample users history. 76
4.9 Prediction Error of various LR models with input features including: H (Hour of the Day), D (Day of the Week), and W (Type of the Day). . . . 77
4.10 Comparison of the two autoregression models according to the time difference metric with various values of k. 78
4.11 Evaluating three variants of GD techniques: Batch GD, Mini-Batch GD, and Stochastic GD with both autoregression and linear regression models according to the prediction error metric. 79

LIST OF TABLES

2.1 A Sample of User Trace 7
2.2 Example of accuracy calculation for O(1) Markov predictor 23
3.1 Training examples extracted from a sub-sequence with the loc2vec neural network model 35
3.2 Generating Mobility Subsequence Matrices and their corresponding labels from a mobility sequence according to a sliding window of length $k = 3$ 43
3.3 Average accuracy with and without fine-tuning for five pre-trained models 48
4.1 Example of the visits history at Workplace when the next location is Home 57
4.2 Example of the visits history at Workplace when the next location is Cafeteria 57
4.3 Example of visit history taking into account the arrival time when the next location is Home. 58
4.4 Example of visit history taking into account the arrival time when the next location is Cafeteria. 58

List of Terms and Abbreviations

MA Moving-Average

MA-AT Moving-Average-Arrival-Time

CDF-AT cumulative distribution function-arrival-time

CDF cumulative distribution function

LR Linear Regression

AR Auto Regression

AP Access Point

WLAN Wireless Local Area Network

GD Gradient Descent

BGD Batch Gradient Descent

M-BGD Mini Batch Gradient Descent

SGD Stochastic Gradient Descent

LSTM Long Short Term Memory

RNN Reccurent Neural Networks

CNN Convolutional Neural Network

STS Spatial Temporal Semantic

CHAPTER 1

Introduction

Mobility is ubiquitous in people's daily life. An individual might move from one place to the other such as moving from home to workplace, from workplace to restaurant, etc., and spend different amounts of time at each place.

With the rapid growth of positioning technology coupled with the ubiquitous use of wearable devices such as smart phones, detecting and recording human movements have become possible almost anywhere and at anytime with various levels of accuracy. Since these movements usually contain spatial and temporal information, modeling user's mobility behaviors basing on this information would help a lot in making accurate mobility prediction models. Predicting user mobility become a critical issue for location based services. It is generally based on analyzing the history of their movements and identifying repeating mobility patterns. It is a fundamental requirement for a wide-range of application areas including urban management, location-based travel recommendation system Ravi and Vairavasundaram (2016), Noulas et al. (2012*b*), Rodriguez-Carrion et al. (2012), advertisement dissemination Aalto et al. (2004), leisure events reports and notifications Marmasse and Schmandt (2000), as well as intelligent HVAC systems Scott et al. (2011).

In this thesis, we focus on the prediction of mobility, and present our approaches to predict individuals next location as well as residence time at a particular location.

1.1 Motivation

1.1.1 Predicting the next location

One strong motivation that drives the research of mobility prediction is recommendation service. By knowing user's future location, many services related not only to their current location, but also to their future destinations can be suggested to the user, such as recommendation of new places or recommendation of nearby restaurants, shops, transportations, etc.

1.1.2 Predicting the residence time

The ability to predict the arrival and residence time of mobile users at a particular place is essential for the development of a wealth of new applications and services, such as smart heating control, transportation planning or urban navigation. Regarding residence time, it has been shown that users tend to spend most of their time in a few places with temporal regularity. In Chon et al. (2012), Montoliu et al. (2013), it has been shown that users spend 60% to 65% of their residence time in the top-1 place and between 80% and 85% of residence-time in the top-2 places. This indicates that, in order to predict the temporal behaviours, focus has to be put on predicting the residence time in places which represents the majority of users' time.

1.2 Problem Statement

Predicting user mobility accurately become a critical issue for location based services. The main goal of this thesis is to build a model able to predict human mobility accurately. In particular, we focus on the following two main research questions.

1. *Which place a user is gong to visit next?*

2. *How long the user will stay at a specific place?*

1.3 Contributions

In this section, we present an overview of the main contributions of this thesis to the mobility prediction.

- **The locations representation**

  Traditional next location prediction algorithms are based on a symbolic representation of locations in a way they consider each location as a different symbol. With such a representation, it is not easy to include more information that provides additional meaningful and helpful description for the location.

  We propose a new location embedding technique called loc2vec in which each location is encoded as a vector by taking into consideration several features. Loc2vec embedding ensures that locations that are likely to appear close to each other in location sequences (i.e. locations frequently seen the one next to the other) are embedded into similar vectors such that the distance between these vectors is small. A better prediction results can be achieved by integrating loc2vec in the prediction model.

- **The next location prediction**

  Traditional prediction models such as those based on Markov chains do not perform well with long sequences, and cannot build a robust prediction model that is not highly dependent on context length. Markovian models have been extensively used in the literature to predict next locations of user. The assumption is that the probability of the next location of a user depends only on a sequence of limited previous locations visited by the user. Usually, lower order Markov model, i.e. 1-order or 2-order is a popular configuration for such model. However, certains people have a complex mobility behaviors and basing on a low sequences length to predict the next location may not be sufficient.

  We propose an innovative representation of mobility subsequences which we call

Mobility Sequence Matrices which allows having a two-dimensional representation of mobility subsequences and thus can be used as inputs for a modern deep learning techniques. The *Mobility Sequence Matrices* could be also seen as similar to images and thus allow us to make use of Convolutional Neural Networks (CNNs) classifiers particularly those which have been pre-trained on image datasets such as ImageNet. With the Mobility Subsequence Matrices representation, we propose two variants of CNN-based location prediction algorithms called onehot-CNN and loc2vec-CNN which are based on onehot and loc2vec location representations respectively.

- **Residence time prediction**

  We focus on predicting the residence time at the current location of a particular user. In the first part, we developed two new models named k-moving-average-arrival-time (k-MA-AT) and k-CDF-arrival-time (K-CDF-AT) by combining the location history, the arrival times and the previous residence time at each location, and tested them against existing models such as k-moving-average (k-MA) and k-CDF which do not take into consideration the arrival time in their model Song, Deshpande, Kozat, Kotz and Jain (2006). Our work differs from Song, Deshpande, Kozat, Kotz and Jain (2006), Scellato et al. (2011) by the way we use the joint temporal and spatial information to predict the residence time. In the second part, we used regression-based learning algorithms to predict the residence time of a particular user at the current location. Previous techniques based on probabilistic models have not been able to perform such prediction accurately. We specifically build models using Linear Regression (LR) and Auto Regression (AR) by considering both linear combination of previous residence times and other spatial or temporal features as well.

1.4 Outline of thesis

The remainder of this thesis is organized as follows:

1. Chapter 2 contains background information about basic concepts, and presents a study of several techniques on mobility prediction.

2. Chapter 3 presents a deep learning methodology as a classification model for next location prediction. It also provides an overview on the most relevant contributions concerning the next location prediction topic.

3. Chapter 4 demonstrates our approaches and models for predicting the residence time of mobile user at particular location.

4. lastly, we conclude our work and further work in Chapter 5

CHAPTER 2

Background

In this chapter, we first define what we mean by location, then we introduce the traces that we used to drive our simulations, next we present several prediction algorithms that allow: (i) to predict future locations, i.e, where will a user go next, and (ii) to estimate the residence time of a user at a particular location, i.e how long a user will stay at a particular location. We also discuss the metrics that we used to evaluate the performance of our prediction algorithms.

2.1 Location

We assume in this work that, at any given time, a user resides at a given discrete location. We assume that the set of all possible locations are listed in a finite alphabet $\mathcal{L} = \{l_1, l_2, \ldots, l_i, \ldots, l_m\}$. We represent the sequence of the locations visited by a user, also called location history H, as a string of symbols. If the history has n locations, $H_{1:n} = l_1 l_2 \ldots l_n$ where $l_i \in \mathcal{L}$ for $1 \leq i \leq n$. In our data, the location is expressed as the access point (AP) with which the user device is associated (i.e., there are n different access points).

2.2 Data collection

The dataset used in this work is a subset of WLAN traces extracted from Dartmouth College Kotz and Essien (2005), Henderson et al. (2008) and made available through the CRAWDAD project Kotz et al. (2009). In this dataset, mobility sequence is expressed in the form of (time, location) pairs for each user where location is taken to be

Table 2.1 A Sample of User Trace

Timestamp	Location (AP)
1008253217	AcadBldg12AP2
1008253716	AcadBldg25AP4
1022867758	AcadBldg20AP1
1022868237	OFF

that of the access point (AP) to which the user is associated as shown in Table 2.1. This dataset contains more than 543 different access points resulting in more than 543 different locations. As users move around these locations, they generate different mobility sequences for different users which lengths vary widely from a user to another reaching several thousand movements for some users.

Table 2.1 also shows a special location named OFF that represents the users departure from the network. The timestamp granularity is one second and measured as UNIX timestamps which count the number of seconds since the epoch. It is to be noted that the location of a user does not necessarily reflect their exact geographical position. It rather indicates an approximation of that location to the one of the access point (AP) that was serving the user at that moment. In this dataset, mobility does not necessarily represent a physical movement of a user. In fact, it is possible that the users device associates and re-associates with a number of different nearby access points without physically moving. Typically, a user situated at boundary of the transmission range of two APs or more may change association with each one of in response to varying radio conditions even the user does not move.

2.3 Location Prediction

Location prediction has become an important task for many applications including urban management Lv et al. (2018), Jiang et al. (2018), Liu and Shoji (2019), transportation recommender systems Rodriguez-Carrion et al. (2012), smartphone energy optimization Chon et al. (2011), etc.

In general, location predictors can be classified into two categories: domain-independent

and domain-dependent predictors. The domain-independent predictors consider only the location history of a user to predict their next location whereas the domain-dependent predictors may include additional information into the location predictor such as time, geographical distance, social relationships, and check-in on location-based social network Noulas et al. (2012*a*), Gonzalez et al. (2008), Cho et al. (2011). In Song et al. (2004), the authors evaluated and compared the performance of several different location predictors by using two popular families of domain-independent predictors, named Order-k ($O(k)$) Markov Predictors, and LZ-based Predictors. The major advantage of this category of domain-independent predictors is that they can be performed online, i.e. by examining the already available history, extracting the k most recent locations, and predict the next location. The sequence of the k most recent locations in the location history is also called the current context.

2.4 Prediction algorithms

A large number of algorithms and techniques can be used to treat the prediction problems. In this thesis, we consider several prediction algorithms that we use for location and time prediction.

2.4.1 Markov Predictors

Here we introduce an overview of the most popular approach used to solve the prediction tasks. The $O(k)$ Markov predictor assumes that the probability of visiting a particular next location depends on the current context defined as the sequence of the k most recent locations in the location history. For instance, if we assume that the next location depends on the current location only, then we refer to this model as the order-1 $O(1)$ Markov model. If the next location now depends on the sequence of the current and the previous locations, we refer to the model as the order-2 $O(2)$ Markov model, and so on. The $O(k)$ Markov model consists of a finite set of states, and transitions from one state to another. The states represent the possible contexts, while the transitions represent the possible locations that follow each context with their corresponding prob-

abilities. More formally, in a location sequence $S_{1:m} = l_1 l_2 \dots l_m$, the $O(k)$ Markov predictor predicts the next location l_{m+1} based on the sequence of the k most recent locations $c = l_{m-k+1} \dots l_{m-1} l_m$ in the history $H_{1:n}$. The probability estimation for the next location to be $l \in \mathcal{L}$ can be calculated as follows:

$$\hat{P}_k(l) = \frac{N(cl, H)}{N(c, H)} \tag{2.1}$$

where $N(cl, H)$ denotes the number of times the sub-sequence cl occurs in the sequence H. Given this estimate, we can predict the location $l \in \mathcal{L}$ with the highest probability, that is, the location that most frequently followed the current context c in the history. If c has never occurred before, the above equation evaluates to 0/1 = 0 for all l, and $O(k)$ Markov predictor can not predict any location.

Example: Consider $H = l_1 l_2 l_3 l_4 l_1 l_2 l_4 l_1 l_2 l_1 l_2 l_3 l_1 l_2$ as the location history in this example. We observe that H contains four distinct relevant locations l_1, l_2, l_3, and l_4. We now want to derive $O(2)$ Markov predictor. The current context (last 2 locations of the history) $l_1 l_2$ has seen 5 times in the location history H. The probabilities would be 1/5, 2/5, 1/5 for l_1, l_3, l_4, respectively. Therefore, it predicts the location c with the highest probability.

The order-k $O(k)$ Markov model has many advantages as it is easy to implement and requires a relatively small memory space. In fact, after each movement to the next location, the predictor updates only one transition probability which make it so fast. The order-k $O(k)$ Markov model has however some limitations caused by the difficulty to find the best value for k a priori as it varies from a situation to another. Note that the order-k $O(k)$ Markov predictor might also be unable to make a prediction when a new pattern that has never been recorded before appears, mostly in case when a human detour from their normal mobility behaviour, e.g., visiting new places, or old places through new routes, etc. Hence, human's next location will not match any previous patterns, which would lead to make inaccurate prediction.

2.4.2 Moving Average Predictor

Moving Averages are most useful to predict a trend in a sequence of values. The order-k average predictor takes a sequence of previous values and predicts that the next value of the sequence is the average of the last k values in that sequence. Consider a sequence of values $v_1, v_2, \ldots, v_n$. The order-k average predictor estimates the next value to be as follows:

$$\hat{v}_{n+1} = \frac{1}{m} \sum_{i=1}^{m} v_{n-i+1} \tag{2.2}$$

where $m = \min\{k, n\}$.

2.4.3 CDF Predictor

The CDF predictor takes a set of values and computes the probability that the next value is less than (or greater than) a given value.

Consider a sequence of values $v_1, v_2, \ldots, v_n$. Assume that V is the random variable that outputs the actual values $v_1, v_2, \ldots, v_n$. The CDF predictor computes the probability that next value, i.e. v_{n+1}, is less than a given value v.

$$\begin{aligned} \hat{v}_{n+1} &= \arg\min_{v} \left(\Pr(V < v) \geq p\right) \\ &= \arg\min_{v} \left(\frac{1}{n} \sum_{i=1}^{n} I(v_i < v) \geq p\right) \end{aligned} \tag{2.3}$$

where I is the indicator function.

2.4.4 Linear Regression-Based Prediction

Linear regression assumes a linearity relation ship between data as shown in the Figure 2.1. In fact, it is a parametric model that computes a linear combination of the input variables using a vector of parameters. Given a set of n input variables $x_1, \ldots, x_n$, and an output value y, Linear Regression $h_{\boldsymbol{\theta}}$ aims at finding the set of parameters $\theta_0, \theta_1, \ldots, \theta_n$

so that:

$$y = h_{\boldsymbol{\theta}}(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{2.4}$$

where θ_j are the weights (also called parameters), and x_j are the features (also called variables) with $j = 1, \ldots, n$. By assuming that $x_0 = 1$ (this is the intercept term) and putting $\mathbf{x} = (x_0, x_1, ..., x_n)^T$, and $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_n)^T$, Eq. (2.4) can be rewritten as:

$$y = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} \tag{2.5}$$

The linear regression model presented in Eq. (2.4) can be used as a method to make prediction for a given set of input features x_j with $j = 1, \ldots, n$.
The model can be trained on a set of m instances $\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots, \mathbf{x^{(m)}}$ with the corresponding labels $y^{(1)}, y^{(2)}, \cdots, y^{(m)}$, respectively, where each vector $\mathbf{x^{(i)}}$ is defined as $\mathbf{x^{(i)}} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})^T$. The aim of the model is find the best value of parameters, called $\hat{\boldsymbol{\theta}}$ and based on the training data, which will be used to predict output values, e.g. $\hat{y}^{(m+1)}$, from a vector of input variables, e.g. $\mathbf{x^{(m+1)}}$, as shown in the following:

$$\hat{y}^{(m+1)} = h_{\boldsymbol{\theta}}\left(\mathbf{x^{(m+1)}}\right) = \hat{\boldsymbol{\theta}}^{\boldsymbol{T}} \mathbf{x^{(m+1)}} \tag{2.6}$$

Finding the best set of parameters $\hat{\boldsymbol{\theta}}$ for function $h_{\boldsymbol{\theta}}$ with m training examples, i.e. $(\mathbf{x^{(1)}}, y^{(1)}), (\mathbf{x^{(2)}}, y^{(2)}), \ldots, (\mathbf{x^{(m)}}, y^{(m)})$, can be obtained by minimizing the square error function $J(\boldsymbol{\theta})$ defined as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x^{(i)}}) - y^{(i)}\right)^2 \tag{2.7}$$

Finding the optimal parameter $\hat{\boldsymbol{\theta}}$ that minimizes $J(\boldsymbol{\theta})$ can be done with a Gradient Descent (GD) algorithm or using Normal Equation (NE) technique that we explain later in the Section 2.5.

Figure 2.1 shows an example of a Linear Regression model $h_{\boldsymbol{\theta}}(x)$ with one input

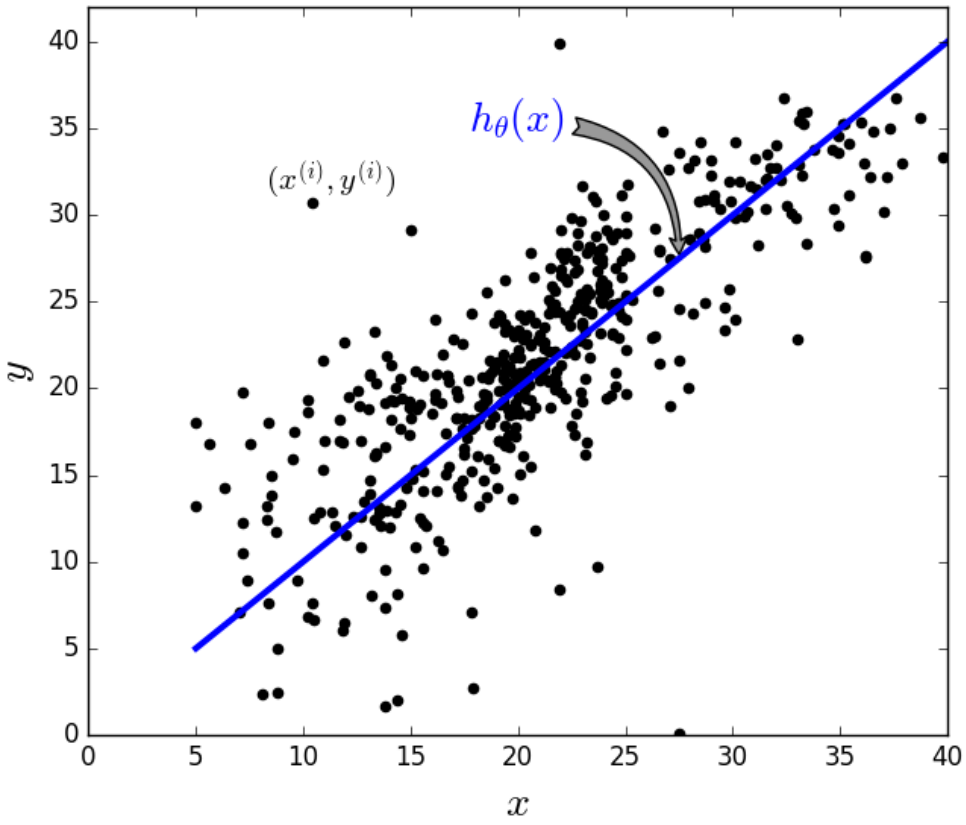

Fig. 2.1 The best fitting straight line after getting the optimal parameter $\hat{\boldsymbol{\theta}}$.

variable x after getting an optimal value of the parameter $\hat{\boldsymbol{\theta}}$ by training the model on a set of m training examples, i.e, $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$.

Figure 2.1 shows how the fitted line $h_{\boldsymbol{\theta}}(x)$ can be used as predictor by assigning a value y to each input value of the variable x.

2.4.5 Auto Regression-Based Prediction

The Auto Regression (AR)-based prediction model is only based on building relations between successive output values, the value to be predicted, say $\hat{y}_{n+1}$ is based on its previous values $y_n, y_{n-1}, \ldots, y_1$. Therefore, an order-$k$ autoregression model can be written as follows:

$$y_n = h_{\boldsymbol{\beta}}(y_{n-1}, y_{n-2}, \ldots, y_{n-k}) + \epsilon_n \tag{2.8}$$

where the output variable y at time n is defined by a function $h_{\boldsymbol{\beta}}$ of the k immediate past values plus an error term for time n. Assume that $h_{\boldsymbol{\beta}}$ is a linear function and consider the history of values $y_1, y_2, \ldots, y_n$. The order-k autoregression model $h_{\boldsymbol{\beta}}$ will be then written as:

$$h_{\boldsymbol{\beta}}(y_{i-1}, y_{i-2}, \ldots, y_{i-k}) = \beta_0 + \beta_1 y_{i-1} + \beta_2 y_{i-2} + \cdots + \beta_k y_{i-k} \tag{2.9}$$

for $i \in \{k+1, \ldots, n\}$. This means that the next value is a linear weighted sum of the k immediate past values.

By putting $\mathbf{y} = (y_{i-1}, y_{i-2}, \ldots, y_{i-k})^T$, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)^T$, Eq. (2.9) can be rewritten as:

$$h_{\boldsymbol{\beta}}(\mathbf{y}) = \boldsymbol{\beta}^T \mathbf{y} \tag{2.10}$$

Also, by defining $\mathbf{Y_{n-1}}$ and $\mathbf{y_n}$ as follows:

$$\mathbf{Y_{n-1}} = \begin{pmatrix} 1 & y_k^{(1)} & \cdots & y_1^{(1)} \\ 1 & y_{k+1}^{(2)} & \cdots & y_2^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_{n-1}^{(m)} & \cdots & y_{n-k}^{(m)} \end{pmatrix}, \mathbf{y_n} = \begin{pmatrix} y_{k+1}^{(1)} \\ y_{k+2}^{(2)} \\ \vdots \\ y_n^{(m)} \end{pmatrix}$$

we have:

$$\mathbf{y_n} = \mathbf{Y_{n-1}} \boldsymbol{\beta} + \epsilon_n \tag{2.11}$$

Given a history of m observations (we take $m = n - k$ for the sake of simplicity and without loss of generality), $\boldsymbol{\beta}$ may be estimated by minimizing the squared error

function $J(\boldsymbol{\beta})$ as follows:

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) \\
&= \arg\min_{\boldsymbol{\beta}} \left(\frac{1}{2m} \sum_{i=k+1}^{n} (y_i - h_{\boldsymbol{\beta}}(y_{i-1}, \ldots, y_{i-k}))^2 \right) \\
&= \arg\min_{\boldsymbol{\beta}} \left(\frac{1}{2m} \sum_{i=k+1}^{n} (y_i - \beta_0 - \beta_1 y_{i-1} - \cdots - \beta_k y_{i-k})^2 \right) \\
&= \arg\min_{\boldsymbol{\beta}} \left(\frac{1}{2m} (\mathbf{Y_{n-1}}\boldsymbol{\beta} - \mathbf{y_n})^T (\mathbf{Y_{n-1}}\boldsymbol{\beta} - \mathbf{y_n}) \right) \\
&= \arg\min_{\boldsymbol{\beta}} \left(\frac{1}{2m} \|\mathbf{Y_{n-1}}\boldsymbol{\beta} - \mathbf{y_n}\|^2 \right)
\end{aligned} \tag{2.12}$$

After estimating the parameters $\hat{\boldsymbol{\beta}}$ by using the Gradient Descent (GD) or the Normal Equation (NE) methods that we explain later in the Section 2.5, the predicted value can be computed by Eq. (2.9) as follows:

$$\hat{y}_{n+1} = h_{\boldsymbol{\beta}}(y_n, y_{n-1}, \ldots, y_{n-k+1}) = \hat{\beta}_0 + \hat{\beta}_1 y_n + \hat{\beta}_2 y_{n-1} + \cdots + \hat{\beta}_k y_{n-k+1} \tag{2.13}$$

Also, Eq. (2.13) can be rewritten, using the vector/matrix format, as follows.

$$\hat{y}_{n+1} = h_{\boldsymbol{\beta}}(\mathbf{y_{n+1}}) = \boldsymbol{\beta}^T \mathbf{y_{n+1}} \tag{2.14}$$

where $\mathbf{y_{n+1}} = (1, y_n, y_{n-1}, \ldots, y_{n-k+1})^T$, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k, \beta_{k+1})^T$.

2.5 Minimizing Squared Error

2.5.1 Using Gradient Descent

Gradient Descent (GD) is an iterative method that is generally used for solving a minimization problem for general functions Ng (2013). In our case, GD can be used to find and update values of a parameter $\boldsymbol{\theta}$ (which could be substituted with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$). . Algorithm 1 shows how GD can be applied to find optimal values of $\boldsymbol{\theta}$ ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$). The

Algorithm 1 Finding $\boldsymbol{\theta}$ using GD algorithm.

1: ▷ Let $\boldsymbol{\theta}^{\text{old}}$ be a given initial value
2: $\boldsymbol{\theta}^{\text{old}} \leftarrow \mathbf{0}$
3: ▷ Iterate until the difference between old value $\boldsymbol{\theta}^{\text{old}}$ and the new value $\boldsymbol{\theta}^{\text{new}}$ is smaller than the preset accuracy ϵ.
4: **repeat**
5: $\quad \boldsymbol{\theta}_j^{\text{new}} \longleftarrow \boldsymbol{\theta}_j^{\text{old}} - \lambda \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ (with $j = 1, \ldots, n$)
6: **until** $(|\boldsymbol{\theta}^{\text{new}} - \boldsymbol{\theta}^{\text{old}}| < \epsilon)$

GD method starts by initializing the vector of parameters $\boldsymbol{\theta}$ and iterates to update the values of $\boldsymbol{\theta}$ with the aim of minimizing the value of $J(\boldsymbol{\theta})$ as shown in the Figure 2.2 and Figure 2.3. The iteration continues until reaching a preset accuracy ϵ. For a given value θ_j (with $j = 1, \ldots, n$) of the vector $\boldsymbol{\theta}$, the algorithm GD operates as the following:

$$\theta_j^{\text{new}} := \theta_j^{\text{old}} - \lambda \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \tag{2.15}$$

where λ is a positive number called the learning rate. It basically controls how big a step will take with GD when updating $\boldsymbol{\theta}$ as showed in Figure 2.4. We have:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}) - y^{(i)} \right) x_j^{(i)} \tag{2.16}$$

Therefore, Eq. (2.15) can be rewritten as:

$$\theta_j^{\text{new}} := \theta_j^{\text{old}} - \lambda \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}) - y^{(i)} \right) x_j^{(i)} \tag{2.17}$$

Each time we change the parameters $\boldsymbol{\theta}$, we choose the gradient that reduces $J(\boldsymbol{\theta})$ the most possible. With each step of gradient descent as shown in the Figure 2.2(b) and Figure 2.3(b), the parameters θ_j come closer to the optimal values that will achieve the lowest cost $J(\theta)$ which corresponds the best fitting straight line as shows the Figure 2.2(a) and Figure 2.3(a).

Eq. (2.17) is the main instruction in the GD algorithm. The updating of $\boldsymbol{\theta}$ can be performed in batch, mini batch or incremental modes thereby resulting in three variants of GD algorithm: Batch GD, Mini-Batch GD, and Stochastic GD.

1. *Stochastic Gradient Descent*

   Instead of updating the parameter $\boldsymbol{\theta}$ based on the whole training set, Stochastic Gradient Descent (SGD) consider a single training example to update the parameter $\boldsymbol{\theta}$. It updates $\boldsymbol{\theta}$ sequentially with every randomly picked training example during the iteration until finding the best value of $\boldsymbol{\theta}$. The algorithm operates as described in Algorithm 2.

Algorithm 2 Finding $\boldsymbol{\theta}$ using SGD algorithm.

```
▷ Let θ be a given initial value
θ ← 0
▷ Iterate until the difference between old value θ and the new value θ^new is smaller than the preset accuracy ϵ.
repeat
    θ^new ← θ
    for i ← 1 to m do
        θ^new ← θ − λ (h_θ(x^(i)) − y^(i)) x^(i)
    end for
until (‖θ^new − θ‖ < ϵ)
```

2. *Batch Gradient Descent*

   In Batch Gradient Descent (BGD), the m training examples are considered at once for every iteration to update $\boldsymbol{\theta}$. Previous equations can be written using matrix notation as the following. Eq. (2.7) can be rewritten, using matrix format, as the following:

$$\begin{aligned} J(\boldsymbol{\theta}) =& \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathbf{i})}) - y^{(i)} \right)^2 \\ =& \frac{1}{2m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(\mathbf{i})} - y^{(i)} \right)^2 \\ =& \frac{1}{2m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \\ =& \frac{1}{2m} \left\| \mathbf{X}\boldsymbol{\theta} - \mathbf{y} \right\|^2 \end{aligned} \tag{2.18}$$

where $\mathbf{X}$ and $\mathbf{y}$ are defined as follows:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \cdots & x_n^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

Similarly, Eq. (2.15) can be rewritten, using the vector/matrix format, as follows.

$$\boldsymbol{\theta}^{\text{new}} := \boldsymbol{\theta}^{\text{old}} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.19}$$

where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is defined as:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \left(\frac{\partial}{\partial \theta_1} J(\boldsymbol{\theta}), \ldots, \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \right)^T \tag{2.20}$$

Hence, Eq. (2.17) can be rewritten as:

$$\boldsymbol{\theta}^{\text{new}} \leftarrow \boldsymbol{\theta}^{\text{old}} - \lambda \frac{1}{m} \left((\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \mathbf{X} \right)^T \tag{2.21}$$

Algorithm 3 Finding $\boldsymbol{\theta}$ using BGD algorithm.

1: ▷ Let $\boldsymbol{\theta}$ be a given initial value
2: $\boldsymbol{\theta} \leftarrow \mathbf{0}$
3: ▷ Iterate until the difference between old value $\boldsymbol{\theta}$ and the new value $\boldsymbol{\theta}^{\text{new}}$ is smaller than the preset accuracy ϵ.
4: **repeat**
5: $\quad \boldsymbol{\theta}^{\text{new}} \leftarrow \boldsymbol{\theta} - \lambda \frac{1}{m} \left((\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \mathbf{X} \right)^T$
6: **until** $(\|\boldsymbol{\theta}^{\text{new}} - \boldsymbol{\theta}\| < \epsilon)$

3. *Mini Batch Gradient Descent*

While the Batch Gradient Descent (BGD) method looks at every example in the whole training set on every step to do a single update for a parameter $\boldsymbol{\theta}$, in

Mini Batch Gradient Descent (M-BGD), on the other hand, only a subset b of the training examples from the entire training set m is considered for every step to update $\boldsymbol{\theta}$, cycling over the training set.

In this case, previous matrices $\mathbf{X}$ and $\mathbf{y}$ can be rewritten as follows:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}^{\{1\}} \\ \vdots \\ \mathbf{X}^{\{i\}} \\ \vdots \\ \mathbf{X}^{\{m-b+1\}} \end{pmatrix}, \mathbf{y} = \begin{pmatrix} \mathbf{y}^{\{1\}} \\ \vdots \\ \mathbf{y}^{\{i\}} \\ \vdots \\ \mathbf{y}^{\{m-b+1\}} \end{pmatrix}$$

where $\mathbf{X}^{\{i\}}$ and $\mathbf{y}^{\{i\}}$ are defined as follows:

$$\mathbf{X}^{\{i\}} = \begin{pmatrix} 1 & x_1^{(i)} & \cdots & x_n^{(i)} \\ 1 & x_1^{(i+1)} & \cdots & x_n^{(i+1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i+b-1)} & \cdots & x_n^{(i+b-1)} \end{pmatrix}, \mathbf{y}^{\{i\}} = \begin{pmatrix} y^{(i)} \\ y^{(i+1)} \\ \vdots \\ y^{(i+b-1)} \end{pmatrix}$$

Algorithm 4 Finding $\boldsymbol{\theta}$ using MBGD algorithm.

1: ▷ Let $\boldsymbol{\theta}$ be a given initial value
2: $\boldsymbol{\theta} \leftarrow \mathbf{0}$
3: ▷ Let $\boldsymbol{b}$ be a given mini-batch size
4: ▷ Iterate until the difference between old value $\boldsymbol{\theta}$ and the new value $\boldsymbol{\theta}^{\text{new}}$ is smaller than the preset accuracy ϵ.
5: **repeat**
6: $\quad \boldsymbol{\theta}^{\text{new}} \leftarrow \boldsymbol{\theta}$
7: $\quad$ **for** $(i \leftarrow 1; (m-b+1); b)$ **do**
8: $\quad\quad \boldsymbol{\theta}^{\text{new}} \leftarrow \boldsymbol{\theta} - \lambda \frac{1}{b} \left(\left(\mathbf{X}^{\{i\}} \boldsymbol{\theta} - \mathbf{y}^{\{i\}} \right) X^{\{i\}} \right)^T$
9: $\quad$ **end for**
10: **until** $(\|\boldsymbol{\theta}^{\text{new}} - \boldsymbol{\theta}\| < \epsilon)$

with $i \in \{1, b+1, 2b+1, 3b+1, \ldots, m-b+1\}$

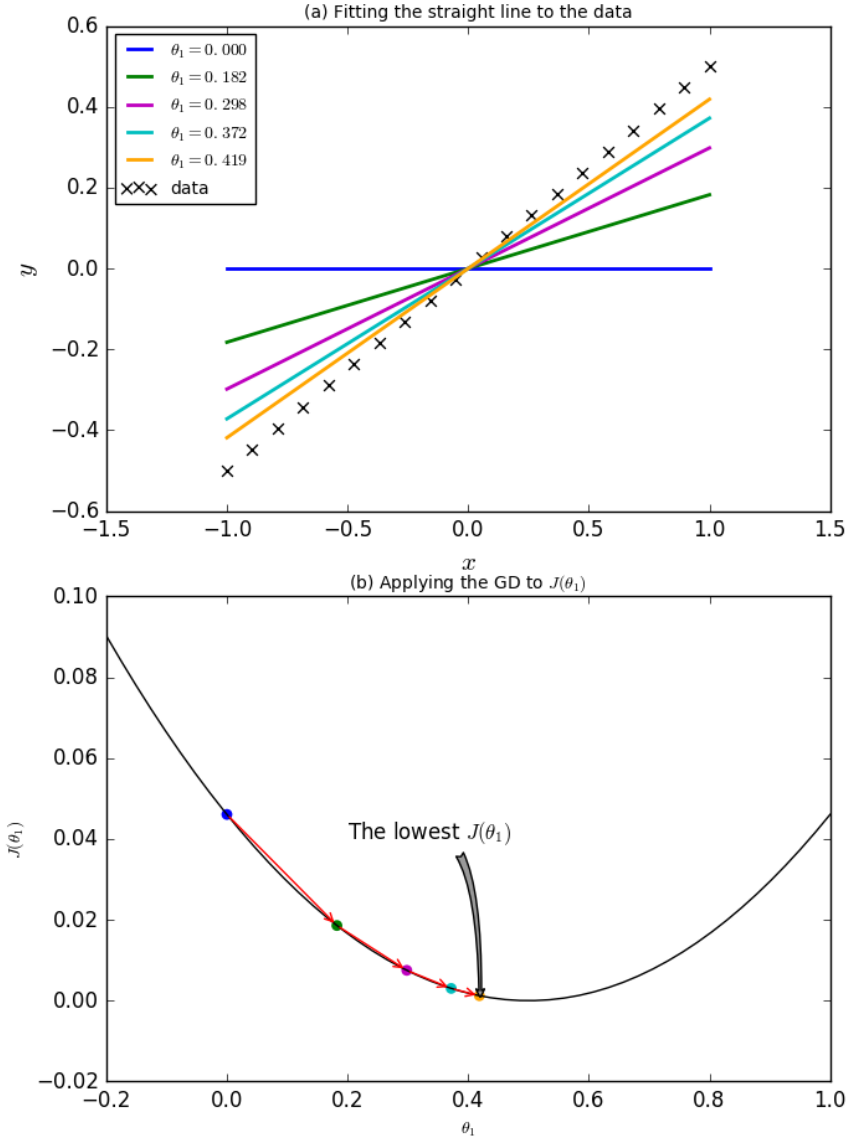

Fig. 2.2 Optimization of the function $h_{\boldsymbol{\theta_1}}$ by applying the gradient descent to the function $J(\boldsymbol{\theta_1})$

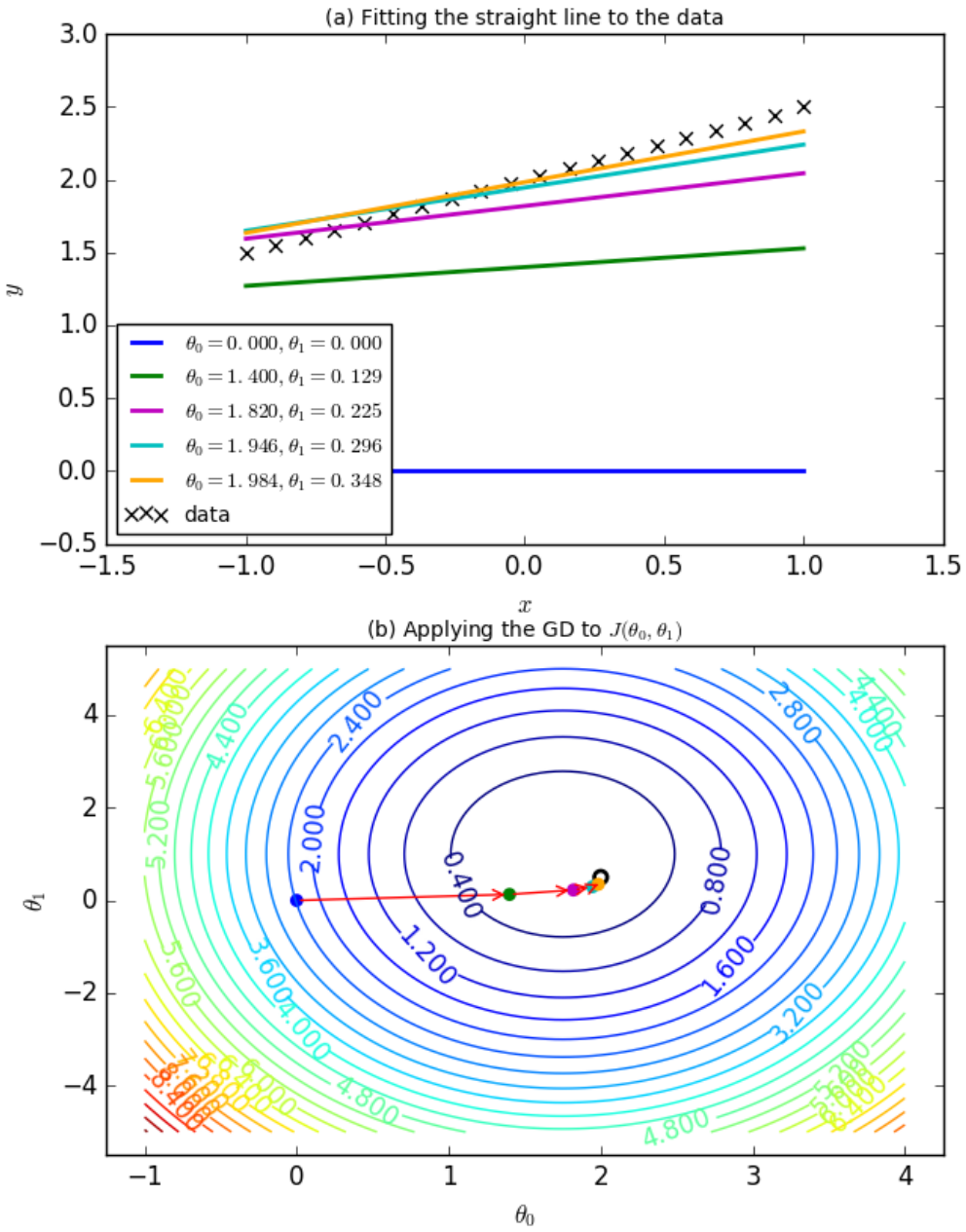

Fig. 2.3 Optimization of the function $h_{\boldsymbol{\theta_0},\boldsymbol{\theta_1}}$ by applying the gradient descent to the function $J(\boldsymbol{\theta_0}, \boldsymbol{\theta_1})$

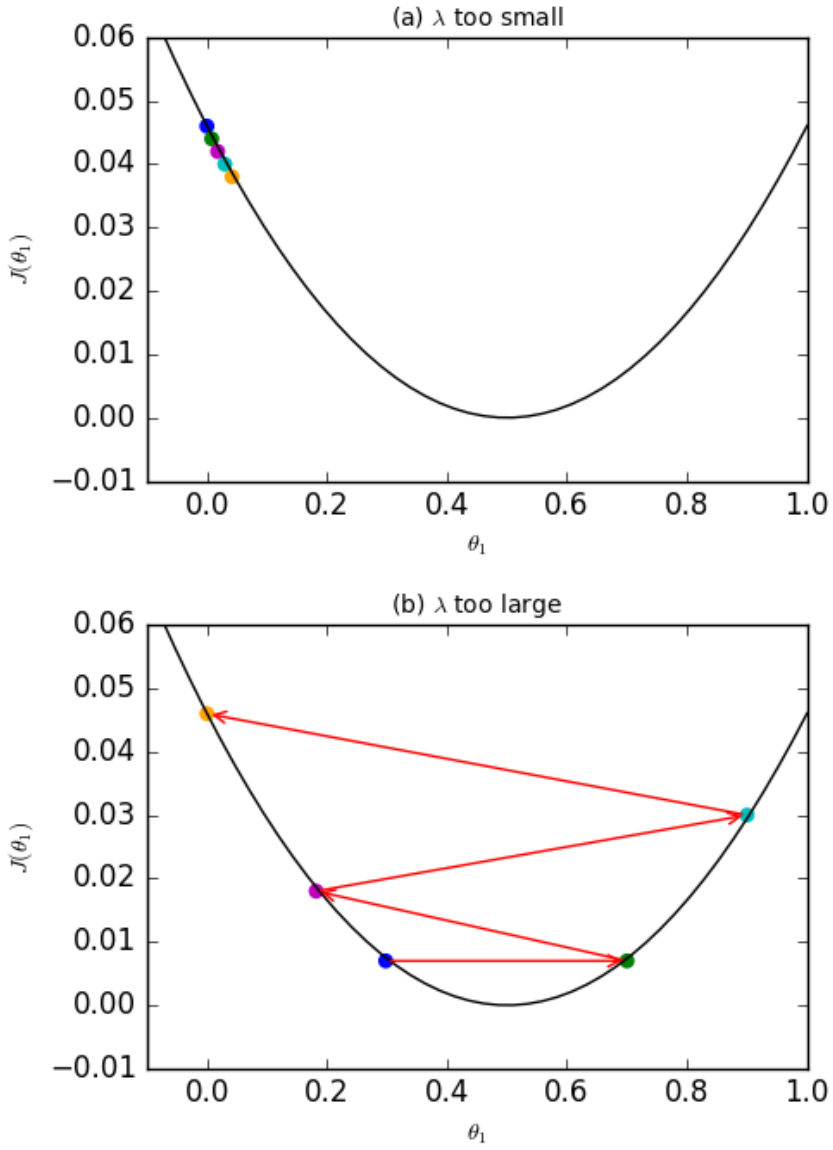

Fig. 2.4 The Learning rate λ

2.5.2 Minimizing Square Error with Normal Equation

Rather than needing to run an iterative algorithm that takes many steps, multiple iterations of GD to converge in to the global minimum, we can instead just solve for the optimal value for $\boldsymbol{\theta}$ analytically using the Normal equation method. So that in basically one step the optimal value of the parameters $\boldsymbol{\theta}$ can be easily computed analytically and is given by :

$$\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T y \tag{2.22}$$

Where $(X^T X)$ is an $(n \times n)$ symmetric matrix and assumed non-singular (invertible).

If $(X^T X)$ not invertible (i.e, singular), we may need to use a pseudo-inverse to compute the parameters θ (In python, numpy.linalg.pinv(a)).

2.6 Prediction metrics

The performance of the predictors listed in Section 2.4 can be evaluated using different performance metrics. According to the study conducted in this thesis, we use two popular metrics that measures the performance of location prediction and time prediction.

2.6.1 Next-location prediction

Predictor should predict the next location after analysing the location history. By comparing the actual location with the predicted one, three possible outcomes for the next location prediction:

- Correct location.
- Incorrect location.
- No prediction.

Predictors may encounter situations in which they can not make prediction especially for the first locations of the history and for new other locations that have not seen before in the history (see Table 2.2). In our evaluation, we consider cases when predictors are unable to make prediction as incorrect prediction. We define the *Accuracy* to be the ratio between the number of correct next location predictions and that of all next location predictions made as follows:

$$\textit{Accuracy} = \frac{\textit{Number of Correct Predictions}}{\textit{Number of Total Predictions}} \tag{2.23}$$

Table 2.2 shows an example of how we calculate the accuracy when applying the order-1 $O(1)$ Markov predictor to the sequence $ababcab$. The last value of the accuracy in the table depicts the overall accuracy.

Table 2.2 Example of accuracy calculation for O(1) Markov predictor

History	a	b	a	b	c	a	b
Prediction	NP	NP	NP	b	a	NP	b
Accuracy	$1/7$	$0/2$	$0/3$	$1/4$	$1/5$	$1/6$	$\mathbf{2/7}$

2.6.2 Time prediction

Since time is a continuous value, it is impossible to build a model able to predict the exact time event. To evaluate a such models, the idea is to measure how much the predicted time value differs from the actual time value. In this thesis, we evaluate the performance of our time predictors using the Prediction Error metric defined as the absolute value of the difference between the predicted time value $\hat{v_p}$ and the real time value v as follows.

$$\textit{Prediction Error} = |v - \hat{v_p}| \tag{2.24}$$

2.7 Conclusion

In this chapter, we have presented several location and time prediction algorithms used in this dissertation. We have also discussed the metrics that we used to evaluate the performance of our prediction algorithms. In the next chapters, our contributions for the next location prediction of users will be presented.

CHAPTER 3

Next Location Prediction

3.1 Introduction

The ability to accurately predict the future locations of mobile users has become a fundamental requirement for a wide range of location-based services in many areas including urban management, travel recommendation systems Ravi and Vairavasundaram (2016), Noulas et al. (2012*b*), Rodriguez-Carrion et al. (2012), advertisement dissemination Aalto et al. (2004), leisure event reporting Marmasse and Schmandt (2000), intelligent HVAC systems Scott et al. (2011), etc.

Predicting future locations of a user is generally based on analyzing the history of their locations and identifying repeating patterns Gonzalez et al. (2008). Many techniques have been used to achieve this. Markovian models are one of the models that have been extensively used in the literature Song et al. (2004), Song, Kotz, Jain and He (2006), Asahara et al. (2011), Gambs et al. (2012, 2010). Markovian models assume that the next location of a user depends on the current context defined as a sequence of the most recent locations visited by the user. According to the considered context length, various accuracy values have been obtained for different situations. Finding the optimal context length that achieves high accuracy values consistently has been one of the main limitations the Markovian models.

With the recent advances in machine learning, new next location prediction algorithms have been developed based on sequence modeling with deep neural networks Liu et al. (2016), Mikolov et al. (2010), Wu et al. (2017). In these models, the locations visited by a user are considered as a sequence of elements. Prediction of the next location is

thus seen as a pattern recognition problem which can be solved using a neural network. For example, in the work presented in Liu et al. (2016), the authors proposed the use of Recurrent Neural Networks (RNNs) for location modeling and solving the next location prediction problem. In their model, they take into account temporal information in addition to spatial information to further enhance their results.

With the recent advances in machine learning, new next location prediction algorithms have been developed based on sequence modeling with deep neural networks Liu et al. (2016), Mikolov et al. (2010), Wu et al. (2017). In these models, the locations visited by a user are considered as a sequence of elements. Prediction of the next location is thus seen as a pattern recognition problem which can be solved using a neural network. For example, in the work presented in Liu et al. (2016), the authors proposed the use of Recurrent Neural Networks (RNNs) for location modeling and solving the next location prediction problem. In their model, they take into account temporal information in addition to spatial information to further enhance their results.

Neural Networks have been generally used efficiently in sequence modeling and next element prediction in many application domains. For example, in Mikolov et al. (2010), the authors developed a neural networks model for the prediction of the next word in a text. In their approach Mikolov et al. (2013), they enhanced the performance of their neural network model by making use of a new embedding technique which consists in encoding words as vectors of real values. The proposed embedding technique called word2vec allowed to achieve significant performance improvement as it ensures that the distance between the vectors representing words reflects the closeness of these words in text documents, i.e. words that tend to be next to each other frequently have very close vector representations.

In our work, we follow a similar approach as word2vec and propose a new location embedding technique that we use on locations instead of words. Similar to word2vec philosophy, loc2vec embedding ensures that locations that are likely to appear close to each other in location sequences (i.e. locations frequently seen the one next to the other) are embedded into vectors such that the distance between these vectors is small.

Next, contrary to existing approaches that use recurrent networks in mobility sequence modeling, and seen that convolutional architectures outperform recurrent networks for a wide range of mobility sequence modeling tasks and datasets Bai et al. (2018), we make use of Convolutional Neural Network (CNN) to perform the prediction of the next location. In our solution, we rely on Convolutional Neural Networks (CNNs) and propose new prediction techniques called onehot-CNN and loc2vec-CNN which are built by converting mobility sequences into matrices which we call *Mobility Sequence Matrices*. These matrices could be seen as similar to images and thus allow us to make use of CNN classifiers particularly those which have been pre-trained on ImageNet datasets. The use of pre-trained CNN models has achieved significant results in various application domains such as sequence modeling Bai et al. (2018), malware binary detection Yue (2017), action recognition Laraba et al. (2017), Minh et al. (2018), sounds classification Boddapati et al. (2017), etc. compared to CNN models established from scratch on the specific data of the considered domain.

The main contributions of this work are the following:

- Traditional next location prediction algorithms are based on a symbolic representation of locations in a way they consider each location as a different symbol. With such a representation, it is not easy to include more information that provides additional meaningful and helpful description for the location. The proposed loc2vec location embedding technique, however, represents each location as a vector by taking into consideration several features. In our case we consider, to represent locations as vectors, the surrounding locations, i.e. previous and next location. A better prediction results can be achieved by integrating loc2vec in the prediction model.

- Traditional prediction models such as those based on Markov chains do not perform well with long sequences, and cannot build a robust prediction model that is not highly dependent on context length. We propose an innovative representation of mobility subsequences which we call Mobility Subsequence Matrix which allows having a two-dimensional representation of mobility subsequences and thus

can be used as inputs for a CNN model that has been pre-trained on large image datasets such as ImageNet Russakovsky et al. (2015). With the Mobility Subsequence Matrices representation, we propose two variants of CNN-based location prediction algorithms called onehot-CNN and loc2vec-CNN which are based on onehot and loc2vec location representations respectively.

- We conduct extensive simulations on large real mobility traces from the CRAWDAD project Kotz and Essien (2005), Henderson et al. (2008) and compare our results with those obtained by means of the prediction techniques based on Markovian models. Experimentation results show that we achieve a stable and higher prediction accuracy compared to those proposed in the literature.

The remainder of the chapter is organized as follows. In Section 3.2, we provide an overview of the main contributions on solving the next location prediction problem. In Section 3.3, we present our prediction model and algorithms. In Section 3.4, we evaluate the performance of our algorithms and discuss the obtained results. In Section 3.5, we conclude the chapter by summarizing our findings.

3.2 Related Work

A variety of algorithms for next location prediction have been proposed in the literature. Most algorithms focused on Markovian models (e.g. Song, Kotz, Jain and He (2006), Gambs et al. (2012)). These models, called Order-k ($O(k)$) Markovian models, assumed that the probability of visiting a particular next location depends on the current context defined as the sequence of the k most recent locations visited. The principle of such models is based on parsing the location history looking for locations that follow all occurrences of the current context (i.e., the sequence of the k most recent locations) in the history, then predict the most frequently one as the next location.

In Song et al. (2004), the authors evaluated and compared the prediction accuracy of several $O(k)$ Markovian location predictors with the goal of enhancing the initial Markovian model with a simple fallback mechanism by decrementing the order of the

model to $k-1$, then $k-2$, ..., in case the searched context has not been seen before in the location history. In their experiments, they found that lower order Markov predictors provide more accurate results compared to higher order ones. In particular, the $O(2)$ Markovian model with fallback has been shown to be the best overall predictor. In Song, Kotz, Jain and He (2006), the authors proposed a time-aided mechanism where each state of the predictor was a pair of location and arrival time (location, time) with the aim of improving the spatial-only Markov location predictor. They considered a quantized time (one minute and one-hour buckets) for the Markov time-aided model to obtain a finite set of states. They found that the prediction accuracy of one-hour time quantization is better than that of one-minute. They also found that the original Markovian model that does not include temporal information provided a better accuracy. They explained their findings by the fact that adding a temporal information in the Markov model increases the number of states. As the number of states increases, the number of the set of next location candidate decreases. This leads to increase the incorrect prediction cases.

In Gambs et al. (2012), the authors proposed a next location prediction algorithm called n-MMC, by using the Mobility Markov Chains (MMC) presented in Gambs et al. (2010). This model incorporates the n previously visited locations to predict the next location of users. The evaluation over different datasets showed that a high accuracy for the next location prediction is obtained with $n = 2$. Similar to the results obtained with Order-k Markovian models, the authors showed that having a context larger than 2 elements did not improve the accuracy of the prediction.

The previously mentioned models have many advantages. They are easy to implement and do not require large memory space as the predictor updates only one transition probability after each movement from one location to another. These models have however some limitations caused by the difficulty of a priori finding the best value for k as these vary from a situation to another. With a large value for k in a Markovian model, chances of encountering a pattern that has been seen before in the location history are slim and thus a prediction cannot be done without performing a fallback into lower

values for k. In Scellato et al. (2011), the authors used delay embedding technique to extract similar patterns from time series. The proposed algorithm, called Nextplace, used the previous arrival and residence times of a user at each of their relevant places to predict the next relevant place. In their proposal, they independently predict the arrival time and the residence time of a user at a given location. The prediction of arrival time (resp. residence time) is based on the similarity existing between the current arrival time pattern (resp. residence time pattern) and the previous arrival times patterns (resp. residence times patterns) of the same location in the location history. Then, they used the predicted values of arrival and residence times of a user at each of their relevant places to predict the next relevant place. Nextplace can only predict the user to be or not to be in one of his relevant places. Compared to Markov predictor, NextPlace model considers the similarity between sequences of arrival time (resp. residence time) without taking into account information about the previously visited locations. They considered Nextplace model as a location-independent predictor.

In Baumann et al. (2013*b*), the authors presented a prediction model in which they ran several parallel predictors and performed voting to select the best predictors. In their proposal they incorporated more information to the Markovian model (the time of the day, the day of the week, etc.). They derived several prediction algorithms by using different combinations of spatial and temporal features. The authors proposed an algorithm, called Major, which predicts the next location of the user based on a voting processes combining the outputs of the several prediction algorithms used.

In Wu et al. (2017), the authors treated the problem of next location prediction as a classification problem. They proposed a solution based on Long Short Term Memory (LSTM) neural networks which is a class of Reccurent Neural Networks (RNNs). The authors proposed a spatial-temporal-semantic neural network algorithm for location prediction called STS-LSTM. First, the algorithm generates a discrete location sequences from the whole trajectory by using a spatial-temporal-semantic feature extraction algorithm (STS). Then, a Long Short-Term Memory (LSTM) neural network model is constructed to make further prediction.

Our proposal differs from the aforementioned work on two aspects. First, we consider a CNN architecture as opposed to RNN, which allows us to take advantage of transfer learning from pre-trained CNNs. Second, our work makes use of location embedding according to the loc2vec technique which contributes in improving the quality of learning and prediction.

This work also considers the location prediction as a classification problem, but it differs from our method by the way we integrate a new location embedding algorithm to represent the locations. It also differs from our solution by the way we apply the CNN-classification based on a transformation of the embedded location sequences to images.

3.3 Proposed Approach

We assume in this work that, at any given time, a user resides at a given discrete location. We assume $\mathcal{L}$ is the set of all discrete locations where $\mathcal{L} = \{l_1, l_2, \ldots, l_i, \ldots, l_n\}$. In our data, the location is expressed as the access point with which the user device is associated (i.e., there are n different access points). Our aim is to answer the question: where will a user go next. First, we describe the proposed approach, illustrated in Figure 3.1, which contains three components as follows:

1. Representation phase: In this phase, we perform location embedding which consists in representing each location $l_i \in \mathcal{L}$ by a vector $\mathbf{v_i}$ of length l where $\mathbf{v_i}$ is defined as $\mathbf{v_i} = (v_{i1}, v_{i2}, \ldots, v_{il})^T$. For a sequence of successive user locations, we consider sequence embedding by replacing each location in the sequence by its corresponding vector. After that, we divide the embedded location sequence (i.e. a sequence of vectors $\mathbf{v_i}$) into multiple sub-sequences with fixed length k. Finally, each subsequence can be represented as a matrix and thus can be seen as an image. We give a label for each subsequence (or image), the label is the next vector in the location embedding sequence. For example: consider a location sequence of a given user as $l_1 l_2 l_3 l_4 l_5$, the corresponding embedding will be

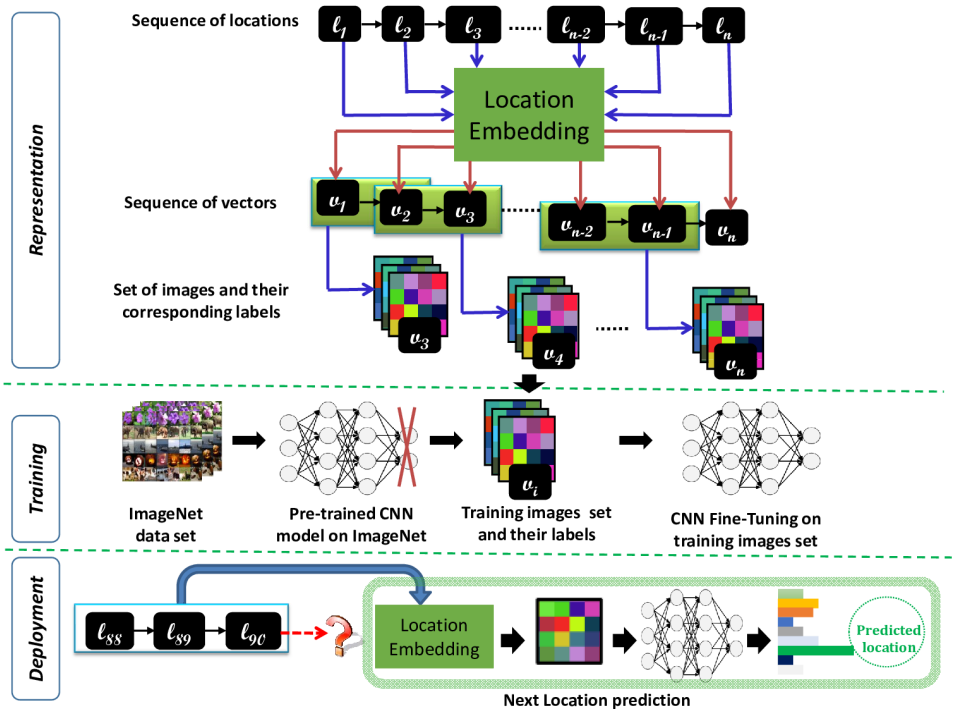

Fig. 3.1 Overview of next location prediction based on CNN.

$\mathbf{v_1v_2v_3v_4v_5}$. If we take a window length of $k = 2$ to construct sub-sequences, we get the following sub-sequences (in the case where we do not take overlapping sub-sequences) $\mathbf{v_1v_2}$ and $\mathbf{v_3v_4}$. We provide a label for each subsequence in the following way: the label of a given sub-sequence is the embedding vector representing the next location in the original sequence, i.e. for the sub-sequence $\mathbf{v_1v_2}$ the label will be $\mathbf{v_3}$ and for the other subsequence $\mathbf{v_3v_4}$ the label will be $\mathbf{v_5}$. Note that sub-sequencing can also generate overlapping sub-sequences as shown in Table 3.2.

The output of this phase is a set of images classified according to their next location labels. We divide the set of images assigned to each label into two parts. We use the first part to train the model and the second part to test it.

2. Training phase: we propose to use a CNN to train an image classification model that will be used to predict future locations.

   - Pre-training: rather than starting from a model with a random configuration

of neural network parameters, we can instead start with parameters of an already trained neural network on a specific task with a very large number of examples Yue (2017), Boddapati et al. (2017), Laraba et al. (2017), Singh et al. (2017). In this phase, deep architectures will be trained on a large dataset of images like ImageNet Russakovsky et al. (2015) using powerful machines with the aim of initializing the network weights to be used for the next phase. This pre-training phase is optional and we evaluate later in this work the improvement while using pre-training. The pre-training phase results in a neural network composed of an input layer, middle layers, and an output layer with the weights set for the connection between the elements of the neural network. The output layer represents the number of classes in the classification problem to be solved.

- Training (fine-tuning): by changing the number of classes of the output layer of the pre-trained model to match the number of next locations, we fine-tune the model by considering a part of images set, i.e the training images set.
- Testing: we test our model by considering the second part of images set.

3. Deployment: the obtained CNN can be used to predict the next location of users by finding the corresponding label (which represents the next location) from a given sub-sequence of user history locations.

3.3.1 Embedding Methods for Location Representations

Embedding methods, which correspond to representing a given piece of data by a vector of reals, are being extensively used as inputs to machine learning algorithms, especially in the deep learning community Mikolov et al. (2013), LeCun et al. (2015), Bengio et al. (2013). Several embedding methods have been proposed in the literature Camacho-Collados and Pilehvar (2018) with one-hot embedding and word embedding being the most popular ones. We consider both methods of embedding with our proposal based on pre-trained CNNs which we name one-hot and loc2vec.

3.3.1.1 One-Hot Representation

In the one-hot embedding, a given location $l_i \in \mathcal{L}$ is represented by a vector $\mathbf{v_i^{onehot}}$ which the dimension l is equal to n where l_1 is embedded as $\mathbf{v_1^{onehot}}$ with $\mathbf{v_1^{onehot}} = (1, 0, \ldots, 0)^T$, l_2 is embedded as $\mathbf{v_2^{onehot}}$ with $\mathbf{v_2^{onehot}} = (0, 1, \ldots, 0)^T$, and l_n is embedded as $\mathbf{v_n^{onehot}}$ with $\mathbf{v_n^{onehot}} = (0, 0, \ldots, 1)^T$. In general, a location l_i is represented by a vector $\mathbf{v_i^{onehot}} = (v_1, v_2, \ldots, v_j, \ldots, v_n)^T$ where:

$$v_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad j \in \{1, \ldots, n\}$$

It is a way to represent locations l_i as vectors $\mathbf{v_i}$ by taking all vocabulary of locations $L = l_1, l_2, \ldots, l_n$, put them in a certain order, and then use one-hot encoding to transform each location l_i to a vector $\mathbf{v_i}$ of length n where each vector $\mathbf{v_i}$ consisting of a single value of 1 at the $i\text{-}th$ index of the vector and zeros in all other indexes. For example, given n vocabulary of locations, the vector $\mathbf{v_1} = (1, 0, 0, \ldots, 0, 0)$ represents the first location and the vector $\mathbf{v_2} = (0, 1, 0, \ldots, 0, 0)$ represents the second location and so on.

3.3.1.2 loc2vec Representation

The previously described representation one-Hot does not take context into consideration. Therefore, we propose another representation inspired from word2vec Mikolov et al. (2013) to take context into account which we call loc2vec. In order to find the best embedding, loc2vec uses a similar approach as word2vec which consists in using a neural network with a single hidden layer as shown in Figure 3.2 to find the best vector corresponding to a given location. The basic idea behind loc2vec is to attribute to each location l_i a vector $\mathbf{v_i^{loc2vec}}$ such as locations that have the same neighboring locations in sequences will have similar embedding. For example, given a hypothetical sequence of locations $l_1 l_2 l_3 l_4 l_1 l_5 l_3$, the locations l_2 and l_5 are surrounded by the same locations l_1 and l_3. Therefore, the locations l_2 and l_5 will have similar vectors $\mathbf{v_2^{loc2vec}}$ and $\mathbf{v_5^{loc2vec}}$

respectively.

We propose to build our loc2vec neural network on locations and their surrounding locations (i.e previous and next locations in the mobility sequence). For a given location l_i the surrounding locations are all location l_j defined such that $j \in \{i-c, \ldots, i-2, i-1, i+1, i+2, \ldots, i+c\}$ where c is the length of the context, which delimits how long the surrounding context is taken around the location i. The loc2vec neural network is built by training it on various pairs (l_i, l_j). We show in Table 3.1 some of the training examples (location pairs) taken from location sub-sequence of length equal to 9 and a context c of length equal to 2.

Table 3.1 Training examples extracted from a sub-sequence with the loc2vec neural network model

Sub-sequence of locations	Training examples (l_i, l_j)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ l_4 l_5 l_6 l_7 l_8 l_9	(l_1, l_2), (l_1, l_3)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ $\mathbf{l_4}$ l_5 l_6 l_7 l_8 l_9	(l_2, l_1), (l_2, l_3), (l_2, l_4)
$\mathbf{l_1}$ $\mathbf{l_2}$ $\mathbf{l_3}$ $\mathbf{l_4}$ $\mathbf{l_5}$ l_6 l_7 l_8 l_9	(l_3, l_1), (l_3, l_2), (l_3, l_4), (l_3, l_5)
...	...
l_1 l_2 l_3 l_4 l_5 l_6 $\mathbf{l_7}$ $\mathbf{l_8}$ $\mathbf{l_9}$	(l_9, l_7), (l_9, l_8)

The main goal of loc2vec is to represent a function that find the best association for each input location l_i with its corresponding output location l_j. To construct the hidden layer of the loc2vec neural network, we use all pairs (l_i, l_j). To achieve this, we consider the one-hot representations $\mathbf{v_i^{onehot}}$ $\mathbf{v_j^{onehot}}$ of locations l_i and l_j.

In this work, we propose two ways of generating sub-sequences from the locations sequence: non overlapping and overlapping sub-sequences. In the overlapping case, we generate an overlapping sub-sequences of fixed length s from the locations sequence by sliding a window of length s across the locations sequence. For example, given a sequence of locations $l_1l_2l_3l_4l_5l_6l_7l_8$, a sub-sequence of length 5 with an overlapping of 2 locations can be generated as follows: $\{l_1l_2l_3l_4l_5, l_4l_5l_6l_7l_8\}$. In this work we focus on a sub-sequences of length $s = 32$ consecutive locations shifted by 15 locations at a time. In non-overlapping case, we generate sub-suequences of length s from the locations sequence while shifting the starting point by s consecutive locations at each

step. In this work, we focus on a sub-sequences of $s = day$ consecutive locations. This means that the length of sub-sequences is variable and depend on the number of locations visited at each a day period.

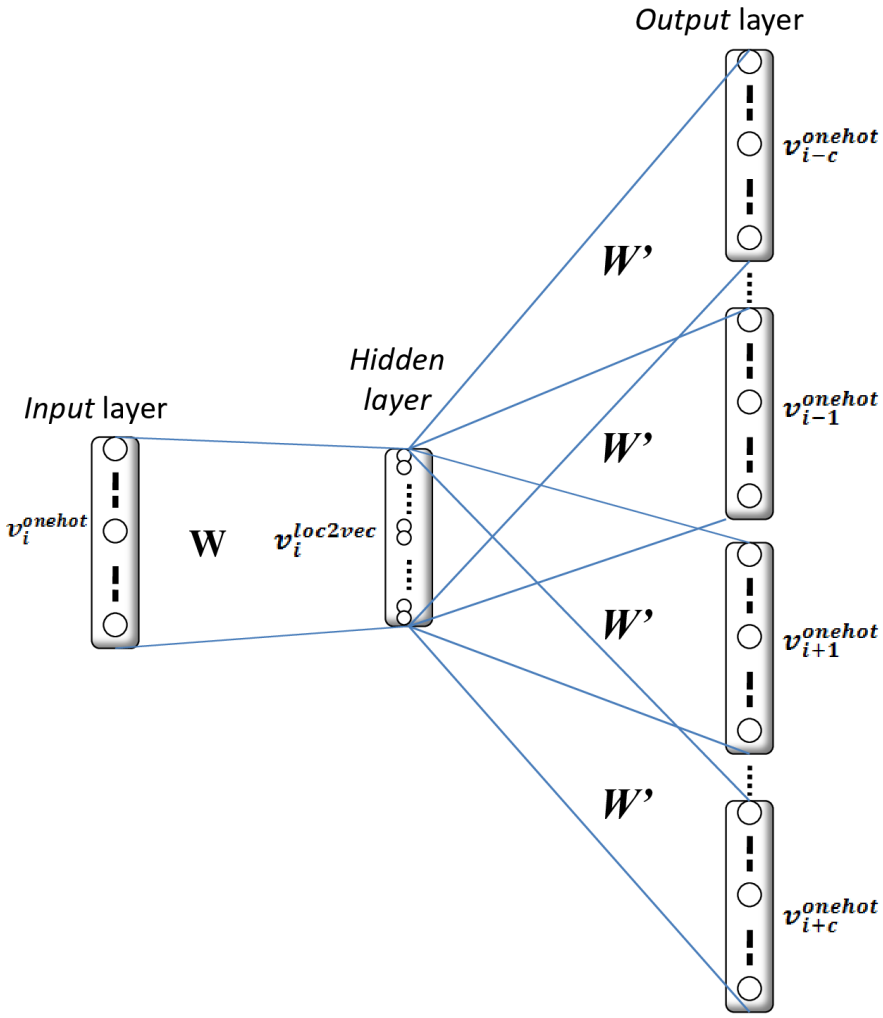

Fig. 3.2 Architecture of the training loc2vec neural network.

Figure 3.2 shows that loc2vec is a neural network with an input layer and a single hidden layer.

In that Figure, v_i represents the one-hot vector corresponding to the input location l_i in the training example and $\{v_{i-C}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+C}\}$ are the one-hot vectors corresponding to the output locations in the training example. Both the input and the outputs are represented as a one-hot vectors of length n.

The goal of loc2vec is to find a representation vector $\mathbf{v}_\mathbf{i}^{\text{loc2vec}}$ for each location l_i. By

contrast to one-hot where the representation vector $\mathbf{v_i^{onehot}}$ of location l_i is of length n, the length of the loc2vec representation vector $\mathbf{v_i^{loc2vec}}$ can be equal to a value m pre-set in advance and considered as a parameter which is the number of nodes in the hidden layer of the loc2vec neural network. To find the values of the vector $\mathbf{v_i^{loc2vec}}$, loc2vec consists in constructing a $n \times m$ matrix $\mathbf{W}$, called the weight matrix, where the i^{th} row of the matrix $\mathbf{W}$ represents the weights of location l_i. We have:

$$\mathbf{v_i^{loc2vec}} = \mathbf{W^T v_i^{onehot}} \tag{3.1}$$

where $\mathbf{W^T}$ is the transpose of matrix $\mathbf{W}$. The elements of the matrix $\mathbf{W}$ are constructed by applying back-propagation and stochastic gradient descent algorithms by considering the set of all inputs $l_i \in \mathcal{L}$ (represented by $\mathbf{v_i^{onehot}}$) and the corresponding outputs l_j (represented by $\mathbf{v_j^{onehot}}$) for each considered l_i.

3.3.2 Convolutional Neural Network for Next Location Prediction

Very good results are achieved using Convolutional Neural Networks (CNNs) in many areas in the literature Taigman et al. (2014), Krizhevsky et al. (2012), Szegedy et al. (n.d.), LeCun et al. (2015). In this work, we propose to use these powerful neural networks for next location prediction of users based on learning from previous mobility sequences. CNNs have a great power in learning patterns and can make correct prediction even with long sequences by generating images from Mobility sequences. We shift toward using images in our representation to exploit standard architectures of images which are already trained on a large dataset of images like ImageNet Russakovsky et al. (2015).

The general architecture of a CNN is presented in Figure 3.3. A CNN is typically composed of a Convolution layer, Max-pooling layer and two Fully Connected layers. Depending on the way these layers are superposed, various architectures can be constructed providing different performance results for different application domains.

A CNN takes various types of inputs and provide results as outputs. Depending on

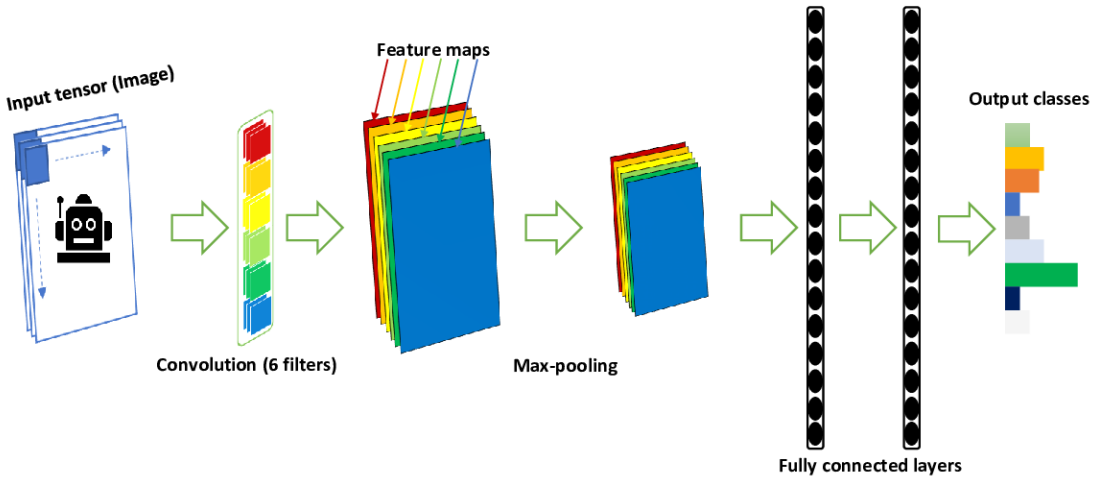

Fig. 3.3 Convolutional Neural Network (CNN)

the structure and the encoding of input, different results may be obtained.

The following subsection gives some details about these layers.

3.3.3 CNN layers

3.3.3.1 Convolution layer

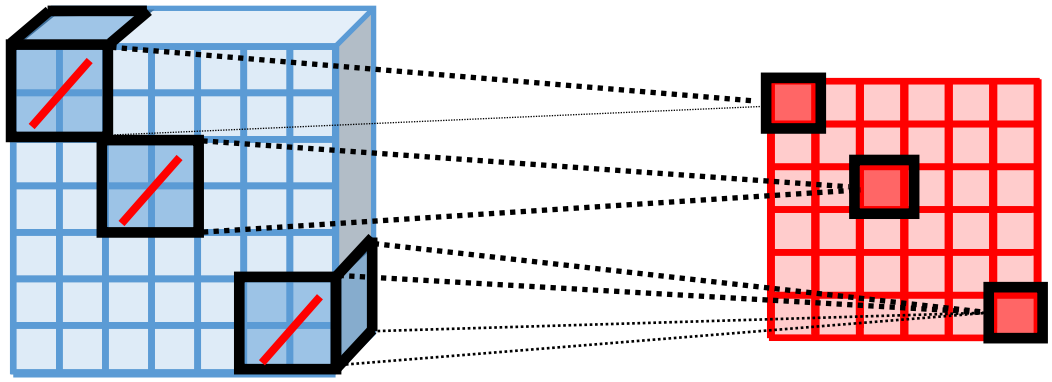

Fig. 3.4 Convolution layer.

The Convolution operation consists of sliding on image/tensor a small rectangular patch of learnable weights. These patches, called filters or kernel, are used to detect local features in images/tensors (see Figure 3.4). More formally, the convolution operation $*$ between an image I having C channels and a filter K having the dimension $k_1 \times k_2 \times D_{in}$ is given by the following formula:

$$(I * K)(i,j) = \sum_{n=0}^{k_1-1} \sum_{m=0}^{k_2-1} \sum_{d=0}^{D_{in}-1} I(i-n, i-m, d) K(n,m,d) \tag{3.2}$$

where $(I * K)(i,j)$ is a value that measures the similarity between the filter K and an image region during the filter sliding. Hence, the matrix $I * K$, called feature map, resumes all activations between the filter and image I. In practice, convolution layer applies D_{in} different filters on input tensor which yields an output tensor composed of D_{out} feature maps.

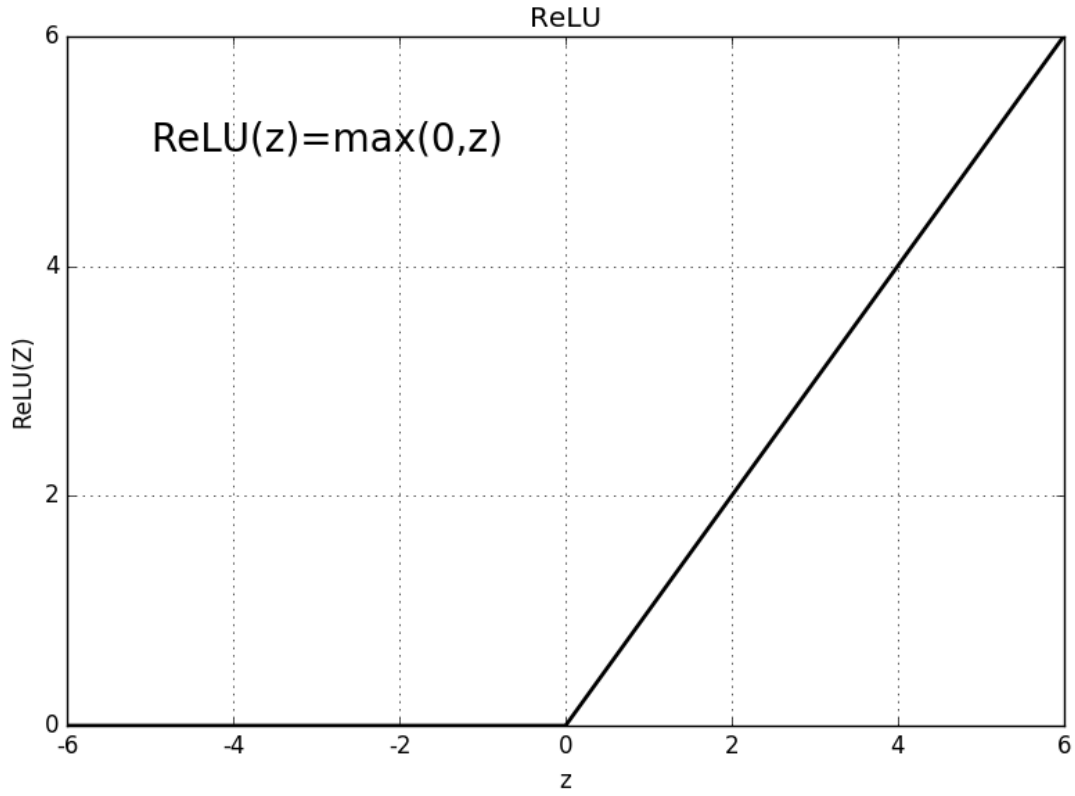

Fig. 3.5 ReLU activation function.

To increase the expressiveness of the CNN, an activation function is applied after the convolution Goodfellow et al. (2016). Recently, Rectified Linear Unit (ReLU) is used as the standard activation function on CNN standard architectures Krizhevsky et al. (2017). ReLU is defined by the following formula:

$$ReLU(z) = \max(0, z) \tag{3.3}$$

where z here is a value from the convolution output tensor. Therefore, the application of RelU function enforce all values calculated by the convolution to be positive or zero.

3.3.3.2 Pooling layer

The Max-Polling layer is used to reduce the spatial size of the inputs while preserving the depth of the input tensor ($D_{in} = D_{out}$). The pooling divides every feature map of the input tensor in non-overlapping rectangular regions. Then, values of each region are reduced to one value using a specific function. For example, max-pooling, used extensively recently, produce the maximum value of each region. In Figure 3.6, max-pooling is applied to a tensor having $D_{in} = 1$. However, if $D_{in} > 1$ then max pooling is applied for each feature map Krizhevsky et al. (2017), Szegedy et al. (2015), He et al. (2015).

3.3.3.3 Fully connected layer

After several stages of constructing features by alternating convolution and pooling layers, fully connected layers are used as the final layers in the CNN architecture Krizhevsky et al. (2017). The fully connected layers are used to implement the decision component by combining all input tensor values to produce a vector that estimates the class of the input image.

3.3.4 CNN Training

CNN is trained using an optimization algorithm in order to minimize a $Loss$ function in respect to the learnable weights W of the network function g. This Loss function

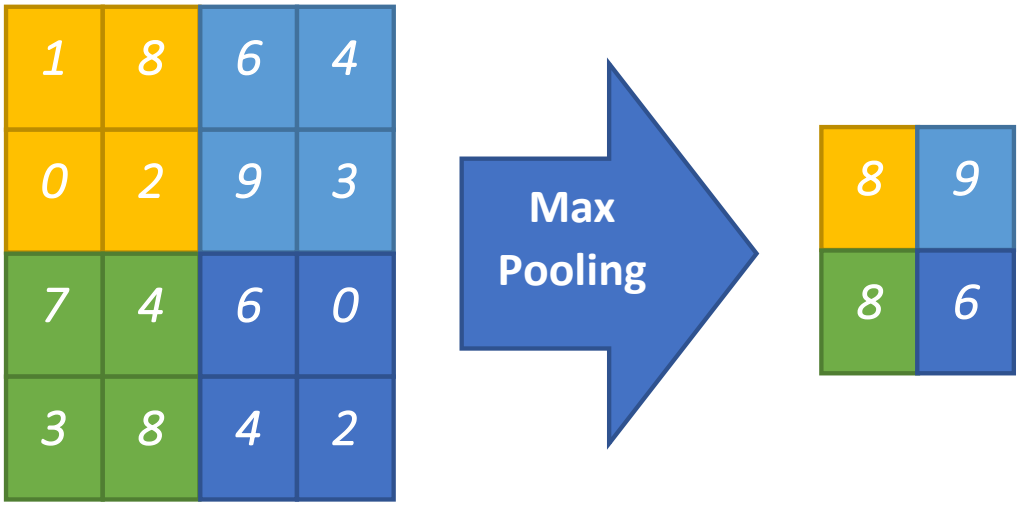

Fig. 3.6 Max-pooling layer.

represents the error of the network output $\hat{y}_i = g(I_i, W)$ compared to the ground truth label y_i for all N examples (I_i, y_i) in the training set. This loss can be written as following:

$$Loss = \sum_{i=1}^{N} Loss_i(\hat{y}_i, y_i) = \sum_{i=1}^{N} Loss_i(g(I_i, W), y_i) \tag{3.4}$$

$$Loss_i(\hat{y}_i, y_i) = -\sum_{i=1}^{C} y_i log(\hat{y}_i) \tag{3.5}$$

To minimize this loss function, gradient descent calculate the gradient $\nabla_W Loss$ of the loss function in respect to all learnable weights W in the network. This gradient is calculated using backpropagation algorithm. Then, network weights are updated as following:

$$W = W - \alpha \nabla_W Loss \tag{3.6}$$

Where α is a hyper-parameter called learning rate. α controls the size of each updating gradient descent step.

As shown in Table 3.2, we propose to encode inputs as Mobility Subsequence Matrices where each matrix is composed of a mobility subsequence of length k as explained in Section 3.3. Each element of the mobility sub-sequence (i.e. each vector $\mathbf{v_i}$ of the Mobility Subsequence Matrix) is represented by one-hot embedding and loc2vec embedding resulting in two methods that we refer to as one-hot-CNN and loc2vec-CNN respectively.

Finally, an image and its label are generated from each Mobility Subsequence Matrix to be used by CNN. Both images and their labels (see Table 3.2) are used to train a CNN model as well as making predictions. Hence, our model will take an image as an input and will output a label which correspond to the next location to be visited.

It is important to note that our loc2vec-CNN construction ensure a spatial locality in the image generation to get closer to the spatial structure of real images. Indeed, each row i of the image corresponds to the encoded vector of location l_i which is itself computed using a neural network based on the surrounding locations (see Section 3.3.1). The vectors corresponding to rows after and before the row i are computed based on neighbouring surrounding locations which guarantee the spatial locality.

The success of using a given CNN solution depends on the architecture and also on the availability of pre-training. It has been shown that a CNN with pre-training generally provides significantly better results than a randomly initialized CNN. We consider using existing pre-trained CNN models that have been constructed based on a large image dataset like ImageNet (e.g. Inception Szegedy et al. (2016), ResNet He et al. (2016), SequeezeNet Iandola et al. (2016), DenseNet Huang et al. (n.d.), etc.)) which showed significant results in various application domains Yue (2017), Boddapati et al. (2017), Laraba et al. (2017), Singh et al. (2017), Minh et al. (2018). The rationale of

relying on images is two-fold: (i) using a pre-trained network is generally better than a randomly initialized one even if the domain applications are different (e.g. environmental sound classification Yue (2017)), and (ii) there is a similarity between our Mobility Subsequence Matrices and images as there is proximity between neighboring pixels in images and neighboring location in a mobility subsequence.

In our solution, we choose to rely on SequeezeNet architecture Iandola et al. (2016) which is a CNN that has been trained on very large image datasets. In our solution, we use a pre-trained CNN model that we fine-tune by further training it on a subset of Mobility Subsequence Matrices (as inputs) and their corresponding labels (as outputs).

Table 3.2 Generating Mobility Subsequence Matrices and their corresponding labels from a mobility sequence according to a sliding window of length $k = 3$

Sequence of vectors	Mobility Sub-sequence Matrix	Label
$\mathbf{v_1 v_2 v_3 \dots v_n}$	$\mathbf{(v_1, v_2, v_3)}^T$	$\mathbf{v_4}$
$\mathbf{v_1 v_2 v_3 v_4 \dots v_{n-1} v_n}$	$\mathbf{(v_2, v_3, v_4)}^T$	$\mathbf{v_5}$
$\dots$	$\dots$	$\dots$
$\mathbf{v_1 \dots v_{n-3} v_{n-2} v_{n-1} v_n}$	$\mathbf{(v_{n-3}, v_{n-2}, v_{n-1})}^T$	$\mathbf{v_n}$

3.4 Performance Evaluation

3.4.1 Effect of loc2vec Parameters Choice

We wrote simulation code in Python and used Gensim Rehurek and Sojka (2010) module to implement our location embedding approach loc2vec. We start by taking a subset of our dataset which we use for the generation of loc2vec embedding for locations. On this subset, we run loc2vec training algorithm by considering all pairs extracted from each mobility subsequence by considering both overlapping and non overlapping subsequences. We set the length of the context $c = 3$, i.e. for each location in the mobility sequence we consider its one and three-hop neighboring locations.

In the overlapping case, we choose to reduce the length of the initial mobility sequences taken from the considered subset by splitting long sequences into subsequences of equal length $s = 32$. However, for the non-overlapping case, we reduce the length

of sequences by splitting long ones into subsequences of lengths covering mobility sequence of one day.

In Figure 3.7, we evaluate the effect of loc2vec construction on the performance of the final prediction algorithm that is based on CNN, called loc2vec-CNN. We use the accuracy metric defined as the ratio between the number of correct next location predictions and that of all next location predictions made. In the evaluation of the effect of loc2vec parameters choice, we consider the following parameters: (i) the embedding vector length m, and (ii) the length of the subsequences s obtained from the splitting of long sequences. We consider the following values (m, s) = (50, 32), (10, 32) and (10, 1 day), with the two first values concerning the overlapping case and the latter one concerning the non-overlapping one. We show that non-overlapping loc2vec-CNN with a variable subsequences length (s=1 day) provides the best performance over the other considered cases. The reason is that constructing subsequences according to a certain logic (activities during a day) captures better the relations between user movements and activities than taking fixed length subsequences.

We also show that having relatively shorter loc2vec vectors is likely to be more accurate than having longer ones as loc2vec-CNN (m=10, s=32) provides better results than a loc2vec-CNN (m=50, s=32). By reducing the length m of location vectors, each vector will contain more information and the mobility subsequence matrix generated from each k vectors may show more patterns.

We also show in Figure 3.7 that the accuracy of the combination of loc2vec with CNN is affected by the choice of parameter k which determines the length of the context that is taken into account for the prediction of the next location. We show that increasing the length of k decreases the accuracy of the prediction which is in concordance with early results on location prediction based on O(k) Markovian models.

The accuracy, however, decreases whenever the length of the window k increases for the three models. For example, with loc2vec-CNN(m=10, s=1 day) model the accuracy decreases by more than 9% between the two values of $k = 8$ and $k = 32$, and decreases of about 17% between the two values of $k = 2$ and $k = 32$.

This means that a high value of the window k does not necessarily improve the model because it becomes difficult to find repeating patterns that would help predicting the next location correctly which is a similar behaviour observed with $O(k)$ Markovian Predictor when k is large.

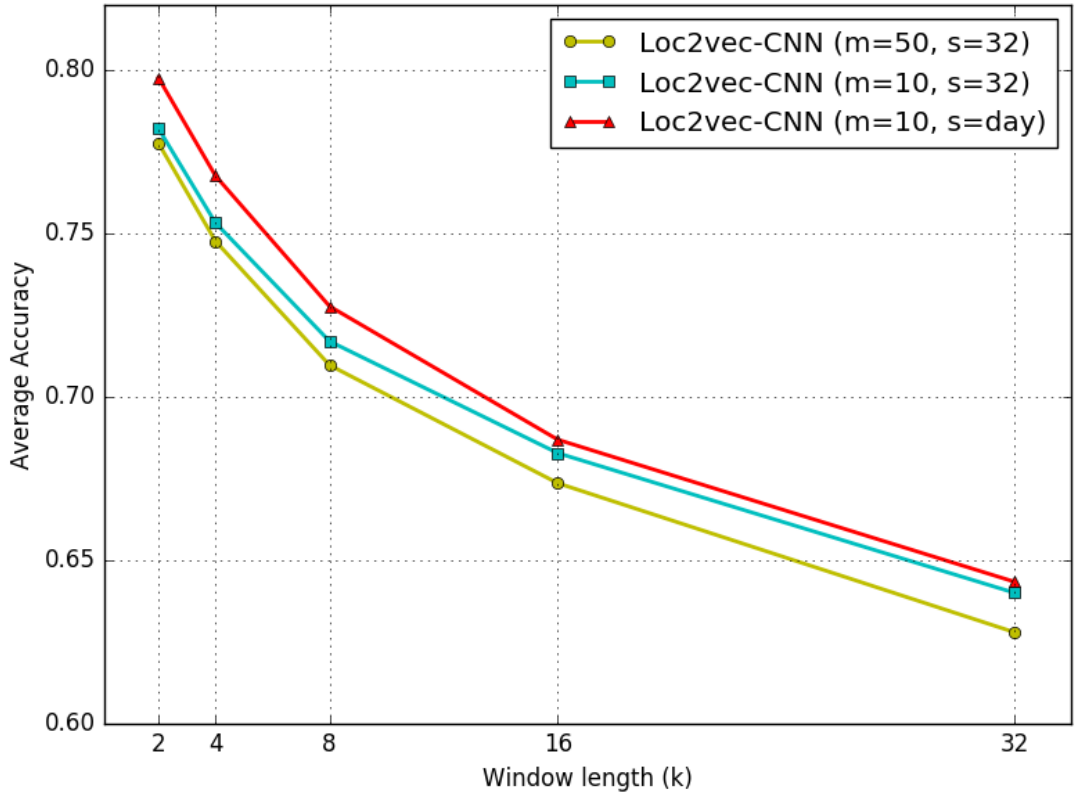

Fig. 3.7 Comparison of the three variants of loc2vec-CNN model according to the average accuracy metric with various subsequence lengths s and embedding vector sizes m.

3.4.2 Evaluating the Performance of CNN-based Predictions

We evaluate the performance of our CNN-based location prediction proposal, we choose two variants for location encoding: one-hot and loc2vec, thereby resulting in two different methods which we call onehot-CNN and loc2vec-CNN respectively. We proceed as the following. We construct a subset by randomly choosing different users whose mobility sequences vary widely in length with a minimum of 2500, a medium of 6000, and a maximum of 13500 location sequences. For each user, we split the location sequence into two halves: we use the first half to build and train the model, and the second half as

a live data to test the performance of our predictor. The training of the proposed CNN model is based on Stochastic gradient descent algorithm with the following hyperparameters: (learning rate = 0.001, momentum = 0.9, batch size= 20, number of epochs = 20).

We compare the performance of our models loc2vec-CNN and onehot-CNN with O(k) Markov predictor, which is one of the most popular next location predictors of the literature.

In Figure 3.8, we evaluate the performance of three models: loc2vec-CNN, onehot-CNN, and O(k) Markov, according to different context lengths by varying the value of k from 2 to 32. We show that the loc2vec-CNN provides a much higher accuracy compared with onehot-CNN for all the values of k. We also show that loc2vec embedding improves the accuracy of about 40% on the average compared to onehot representation.

We also show that low-order loc2vec-CNN models worked as well or better than high-order ones, and better than both onehot-CNN and O(k) Markov models. Figure 3.8 shows that O(k) Markov is better than onehot-CNN only when k is small. It also shows that increasing the value of k in both loc2vec-CNN and onehot-CNN provides a smaller decreasing in the accuracy compared to the O(k) Markov model which exhibits a very high decreasing in the accuracy. For example, with O(k) Markov, the average accuracy decreases by more than 30% between the two values of $k = 8$ and $k = 32$ whereas with our models onehot-CNN and loc2vec-CNN, it only decreases by less than 4% and 7% respectively for the same values of k.

This means that CNN models are able to keep somewhat the accuracy even with a high order of context k due to its efficiency in learning patterns contrary to traditional mobility prediction models such as O(k) Markovian which are based on examining the history of sequence to predict the next location.

In Figure 3.9, we plot comparison results of 20 users with a context length $k = 8$, a sequence length s corresponding to one day, and a loc2vec embedding vector size $m = 10$. We show that the accuracy varies from a user to another for models loc2vec-

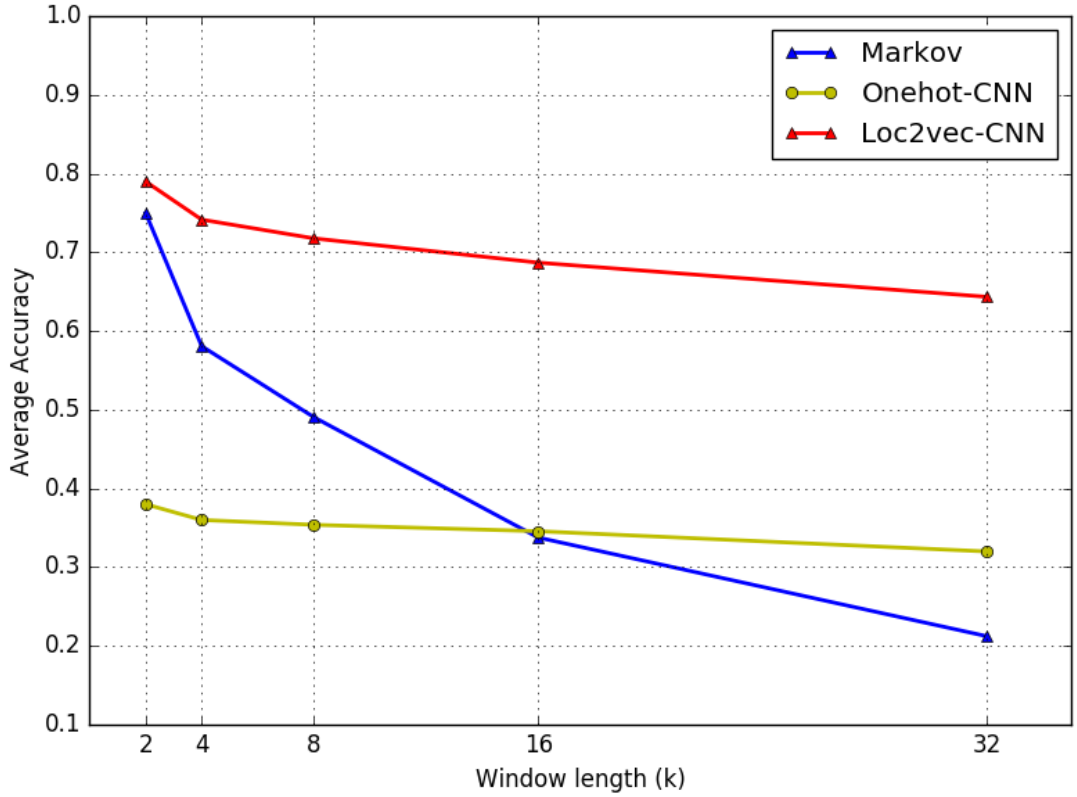

Fig. 3.8 Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov, according to the average accuracy metric.

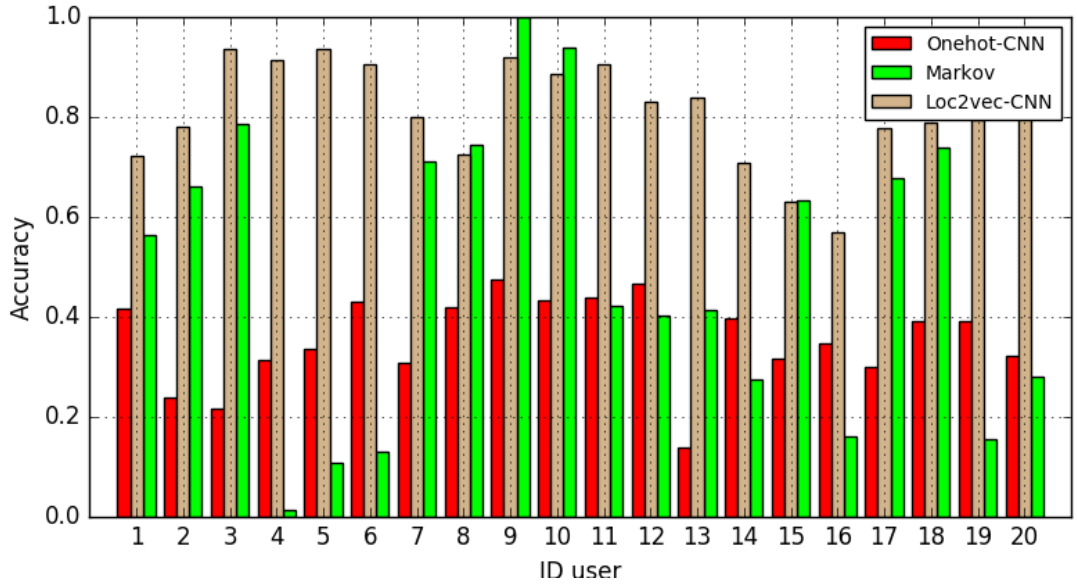

Fig. 3.9 Comparison of loc2vec-CNN, onehot-CNN and O(k) Markov according to the accuracy metric for every user.

CNN, onehot-CNN, and O(k) Markov. The results show that loc2vec-CNN provides the best accuracy compared to the other models for almost all considered users. Only in four cases, O(k) Markov provided slightly better accuracy compared to loc2vec-CNN.

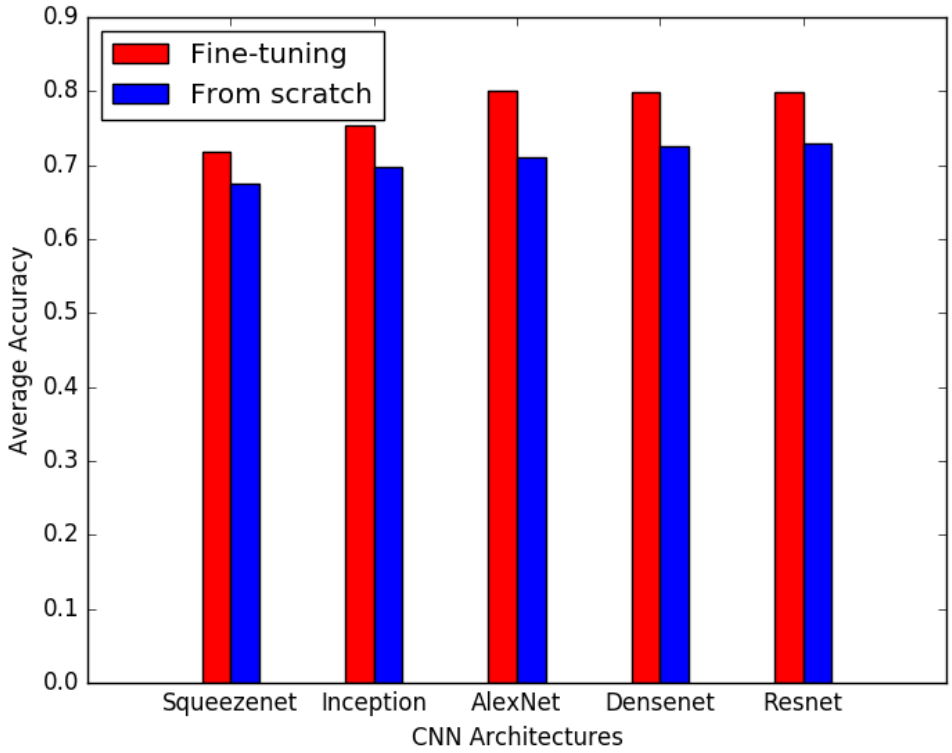

Fig. 3.10 Effect of pre-training on the performance of loc2vec-CNN.

Table 3.3 Average accuracy with and without fine-tuning for five pre-trained models

Models	Fine Tuning	From Scratch
SqueezeNet	71.78%	67.54%
Inception	75.31%	69.76%
AlexNet	79.97%	71.14%
DenseNet	79.87%	72.52%
ResNet	79.92%	72.90%

In Figure 3.10 and Table 3.3, we compare the average accuracy of five existing CNN models by training the whole network from scratch, and by fine-tuning these models using transfer learning from existing popular pre-trained models. We considered the following models: SqueezeNet Iandola et al. (2016) (the model used to obtain the previous results in this work), Inception Szegedy et al. (2016), AlexNet Krizhevsky

et al. (2012), DenseNet Huang et al. (n.d.) and ResNet He et al. (2016). We show that starting from an already trained CNN model provides a much higher accuracy compared to the training from scratch. We show that the average accuracy obtained with all the five considered CNN models increases with fine-tuning. The difference between a model trained from scratch and another one taking advantage of transfer learning after fine-tuning exceeds 7% for most considered models.

The use of pre-trained models can be viewed as transfer learning from real images to our images. This transfer learning may be justified by the presence of some low-level image features (edges, colors, etc.) shared between real images and generated images.

3.5 Conclusions

In this chapter, we have presented and evaluated several models for next location prediction using a subset of real mobility traces. In contrast to most existing proposals, we proposed to make use of modern machine learning techniques based on deep neural networks. We have proposed the use of Convolutional Neural Networks for which we enhanced the representation of input data by the use of embedding techniques. We have explicitly derived a new location embedding technique which we called loc2vec to enhance the quality of input location representations. Our loc2vec embedding technique improves the representation of locations by encoding close locations in mobility sequences in a way that makes their loc2vec representations also close after the embedding. We enhanced the performance of our CNN models that are based on loc2vec embedding with the use of transfer learning which allows us to take advantage of pre-trained CNN networks which we fine-tuned on our location prediction application domain. We evaluated the performance of our proposals on real mobility datasets and showed that the combination of loc2vec, CNN, and the use of transfer learning from existing CNN model provide the best results compared to popular state of the art prediction techniques relying on Markovian models.

CHAPTER 4

Residence Time Prediction

4.1 Introduction

The ability to predict the arrival and residence time of mobile users at a particular place is essential for the development of a wealth of new applications and services, such as smart heating control, transportation planning or urban navigation.

In addition to the prediction of spatial aspects of human mobility such as predicting the next location Wu et al. (2018), Zheng et al. (2018), Sassi et al. (2019), there has been several models and algorithms developed to predict the temporal aspects too. Most of these methods focused on predicting the residence times of users at their relevant places typically defined as the places frequently visited by those users Baumann et al. (2013*a*), Chon et al. (2012), Sassi et al. (2017). Others focused on predicting both spatial and temporal aspects as the same time such as predicting the residence times at the current location and the next location movement Song, Deshpande, Kozat, Kotz and Jain (2006), Chon et al. (2012), Scellato et al. (2011).

Regarding residence time, it has been shown that users tend to spend most of their time in a few places with temporal regularity. In Chon et al. (2012), Montoliu et al. (2013), it has been shown that users spend 60% to 65% of their residence time in the top-1 place and between 80% and 85% of residence-time in the top-2 places. This indicates that, in order to predict the temporal behaviours, focus has to be put on predicting the residence time in places which represents the majority of users' time.

In this chapter, we describe our models and the algorithms we use in this thesis to predict the residence time of a user at their relevant places. Our aim is to find an answer

to the question: how long a user will stay at their relevant places in the future? We consider relevant places as those places that the user visits more frequently compared to other places.

This chapter is organized as follows:

- The chapter starts by presenting an overview of the related works on the residence time prediction (section 4.2).
- The Section 4.3 describes the system model of the first contribution on predicting the residence time of mobile users. It also presents the different algorithms we use to solve the residence time prediction problem (Section 4.3.2). At the end of the first contribution (i.e. Section 4.3.3), a series of experiments are presented in order to evaluate the performance of the presented approach.
- In the Section 4.4, we present the second contribution of this chapter. Like the previous Section 4.3, this section (Section 4.4) presents algorithms, discussion and evaluation of these algorithms.
- Finally, section 4.6 concludes the chapter by a general discussion of temporal prediction models.

4.2 Related Work

Human mobility prediction has attracted extensive research covering both temporal and spatial aspects Pirozmand et al. (2014).

In Song et al. (2004), the authors evaluated and compared the performance of several different location predictors by using two popular families of domain-independent predictors, named Order-k ($O(k)$) Markov Predictors, and LZ-based Predictors. The major advantage of this category of domain-independent predictors is that they can be performed online, i.e. by examining the already available history, extracting the k most recent locations, and predict the next location. The sequence of the k most recent locations in the location history is also called the current context. The $O(k)$ Markov model

consists of a finite set of states, and transitions from one state to another. The states represent the possible contexts, while the transitions represent the possible locations that follow each context with their corresponding probabilities.

In Song et al. (2004), the authors found that a low order Markov predictors provide more accurate results compared to a high order ones. They also showed that adding fallback to those lower order predictors further improves their accuracy.

Markovian models such as the Order-k $O(k)$ Markov model can be considered as a domain independent model which targets to predict the next residence time of a user based only on the history of previous residence times Song, Deshpande, Kozat, Kotz and Jain (2006), Chon et al. (2012), Baumann et al. (2013a).

In Song, Deshpande, Kozat, Kotz and Jain (2006), the authors applied a Markov model to a sequence of previous durations, where each duration is quantized into intervals of equal lengths.

The $O(k)$ Markov model has many advantages as it is easy to implement and requires a relatively small memory space. In fact, after each movement to the next location, the predictor updates only one transition probability which make it relatively fast. The $O(k)$ Markov model has however some limitations caused by the difficulty to find the best value for k a priori as it varies from a situation to another. Note that the $O(k)$ Markov predictor might also be unable to make a prediction when a new pattern that has never been recorded before appears. To cope with the case where a predictors encountered a pattern that has never been seen before in the location history, which would affect the prediction quality by failing to make prediction, the authors of Song et al. (2004), Song, Kotz, Jain and He (2006), Chon et al. (2012) used a fallback mechanism to reduce the length of the pattern sequence gradually until a matching becomes available so it would be able to make a prediction. In brief, the fallback mechanism recursively uses the result of the low-order predictor if the high-order predictor has no prediction result. The fallback mechanism has been shown to significantly improve the overall accuracy of the predictor.

In Vu et al. (2011), the authors proposed a model based on a Naive Bayesian clas-

sifier to predict next location and residence duration at that location. They added extra information such as the type of the day (weekdays or weekends) and quantized the time into one and two hours buckets. They found that adding these high granularity time-related information improved the prediction quality.

In Scellato et al. (2011), the authors used delay embedding technique to extract similar temporal patterns from time series. The proposed algorithm used the previous residence times (resp. arrival times) of a user at a given location to predict the next residence times (resp. arrival times). The prediction of residence time (resp. arrival time) is based on the similarity existing between the current residence time pattern (resp. arrival time pattern) and the previous residence times patterns (resp. arrival times patterns) of the same location in the location history. To predict residence time they consider the similarity between sequences of residence time without taking into account other temporal features about the current residence time pattern.

In Song, Deshpande, Kozat, Kotz and Jain (2006), the authors considered three techniques to predict the next residence time by considering the history of previous residence times at a location which are: Markovian, Moving-Average, Cumulative Distribution Function (CDF), predictors. They combined spatial and temporal information to predict both the next spatial and temporal information about a particular user. They particularly combined a Markovian location predictor with a duration predictor where the states of the Markovian location model are only based on spatial information and do not depend on temporal information. The next location prediction is based on the current context and the transition probabilities between the states of the Markovian model. Regarding the residence time predictor, it is based on the previous residence times at the same location. For the residence time prediction, they used CDF time predictors but without integrating the arrival time information in the predictor. They named the proposed integrated approach that predicts the location and time jointly MarkovCDF. The predictor outputs a list of location and probability pairs; the user may move to each of the locations with the corresponding probability within duration d.

In Sassi et al. (2017), the authors proposed a time predictor model to predict the

residence time at the current location of a particular user by combining both the current arrival time and the previous residence times at that location. Their work differs from Song, Deshpande, Kozat, Kotz and Jain (2006), Scellato et al. (2011) by the way they used the joint temporal and spatial information to predict the residence time. They developed two time-aided algorithms that include the arrival time in the model named k-moving-average-arrival-time and k-CDF-arrival-time and tested them against existing models such as k-moving-average and k-CDF which do not take into consideration the arrival time in their model Song, Deshpande, Kozat, Kotz and Jain (2006). They showed that including the arrival time information in the models used to predict the residence time, i.e k-moving-average-arrival-time and k-CDF-arrival-time provides a significant improvement of the prediction accuracy by about 20% on the average compared to k-moving-average and k-CDF models proposed in Song, Deshpande, Kozat, Kotz and Jain (2006).

In Baumann et al. (2013*a*), the authors analyzed the theoretical predictability of arrival and residence times and evaluated the performance of eight different existing residence time predictors proposed in Song, Deshpande, Kozat, Kotz and Jain (2006), Scellato et al. (2011). They also consider MarkovCDF time-aided (i.e., the residence time depends on the arrival time) which differs from Song, Deshpande, Kozat, Kotz and Jain (2006) by the way they included the arrival time in the model to predict both the next spatial and temporal aspects. They found that the predictability of the arrival times is in general lower than the predictability of the residence times, and that spending a higher period of time at a specific place influences negatively on the predictability of the arrival and residence time.

In Chon et al. (2012), the authors evaluated several time-aided mobility models for the prediction of the residence time of users at their current locations. In their work, they predict the residence time of the current location by looking at the residence times at the same location in the location history of the user in addition to the arrival time. They take only locations that have similar location sequence pattern. They showed that the location-dependent model is better than the location-independent model for the pre-

diction of temporal behavior, and that a longer location-sequence does not necessarily improve the accuracy of the residence time prediction. They also showed that the spatiotemporal regularity is only inherent to a few locations and in other situations previous locations do not help improve the accuracy of the residence time.

4.3 First System Model

In our first model, we particularly focus on two parameters: the arrival time, and the residence time at that location. Consider a user with the following time-ordered movement sequence history H such that:

$$H = \{(t_1, d_1, l_1), (t_2, d_2, l_2), \ldots, (t_n, d_n, l_n)\} \tag{4.1}$$

where t_i are the arrival times, d_i are the residence time and l_i are the locations for $1 \leq i \leq n$. From the movement history H we extract the location history $L = l_1, l_2, \ldots, l_{n-k+1}, \ldots, l_{n-1}, l_n$. We define the k recent location context $c_l(k)$ as subset of L such that $c_l(k) = L(n-k+1, n) = l_{n-k+1}, \ldots, l_{n-1}, l_n$. The visit history at a particular location l_j is $H_j = \{(t_1, d_1, l_j), (t_2, d_2, l_j), \ldots, (t_n, d_n, l_j)\}$. From H_j, we extract the history of arrival times at location l_j, $T_j = t_1, t_2, \ldots, t_{n-k+1}, \ldots, t_{n-1}, t_n$, and recent k arrival time context $c_{l,t}(k) = T(n-k+1, n) = t_{n-k+1}, ..., t_{n-1}, t_n$.

To predict the residence time at a certain location l_j, where $1 \leq j \leq m$ and m is a finite set of relevant places, previous works proceed as the following. For the entire history of locations L, identify all occurrences of the context c_a. For each occurrence of c_a find the set of all possible destinations. For each destination x, calculate the set of residence times corresponding to the last location in each occurrence of the context. The set of all these durations is D_x. Formally, D_x is defined as the following.

$$D_x = \{d_i | d_i = t_{i+1} - t_i \text{ where } L(i-k+1, i+1) = (c_a x)\} \tag{4.2}$$

In this case, the set D_x will contain all the residence times of the current location, i.e. the one that will correspond to l_j, where the next location after l_j will be x. Note that

D_x has been constructed according to contexts of length equal to k.

Note that the previously described approach, proposed in Song, Deshpande, Kozat, Kotz and Jain (2006), does not take into consideration the effect of arrival time in predicting the residence time. As we will show, the arrival time has a significant effect on the residence time, we propose a new model in which we include the arrival time in the prediction of the residence time.

We start by constructing the set of all occurrences of a given context, c_a. We sort all occurrences of this context according to the arrival time which we quantize to obtain a discrete set of possible arrival times. We can take as an example, a one-hour bucket for quantization. Therefore, we will have time-dependent context, $c_(a,t)$ which contain all occurrences of the context where the arrival time at the last location of the context is equal to t. For each time-dependent context $c_{a,t}$, we perform a similar technique as the previously described one to construct the set of all time-dependent resident times $D_{t,x}$ for each possible destination x that follows the time-dependent context $c_{a,t}$. Formally, the set $D_{t,x}$ is defined as the following:

$$D_{t,x} = \{d_i | d_i = t_{i+1} - t_i \text{ where } L(i-k+1, i+1) = (c_{a,t}x)\} \tag{4.3}$$

Note that the set of residence times $D_{t,x}$ which we constructed in our model is a subset of the set of residence times D_x constructed in the previously cited work.

Our subset contains only residence times that have the same quantized arrival time in the history of visits to that location according to the context length we chose which we took equal to k. We will show that this will significantly improve the residence time prediction accuracy. The rationale behind including the arrival time in the prediction of the residence time is that in most cases the residence time depends on the arrival time. For example, assume that we have a user that arrived at their workplace at 9am. The user is most likely to leave work place at the end of the work shift, typically at around 5pm. In this case the residence time is about 8 hours. Note that even if the same user arrived at 10am or 10.30am, they are likely to leave at 5pm too, in which case the residence time becomes 7 hours or 6.5 hours respectively which is different from the

Table 4.1 Example of the visits history at Workplace when the next location is Home

Work Place		Home	
AT	RT	AT	RT
10.45pm	90min	12.15pm	45min
1.10pm	320min	6.30pm	90min
...	...	...	...
...	...	...	...
3.10pm	195min	6.25pm	120min
10.50am	?	?	?

Table 4.2 Example of the visits history at Workplace when the next location is Cafeteria

Work Place		Cafeteria	
AT	RT	AT	RT
12.15am	55min	1.10pm	20min
5pm	98min	7.32pm	30min
...	...	...	...
...	...	...	...
12pm	?	?	?

original residence time of 8 hours.

4.3.1 An illustrative Example

Assume that the current location of a user is Workplace and the possible next locations according to the context are Home and Cafeteria. In our model, we need to estimate the residence time at Workplace for each of the possible next locations: Home and Cafeteria. Tables 4.1 and 4.2 provide a view on the history of mobility data of the user. We use AT (resp. RT) to refer to the Arrival Time (resp. Residence Time)

When relying only on the set of residence times corresponds to the one defined in Eq. (4.2). From the mobility data presented in tables 4.1 and 4.2, the set of residence times corresponding to a current location equal to Workplace is obtained as follows.

D_{Home} = {90min, 320min, ..., 195min, where $L(i, i+1) = (c_a, x)$ = Workplace, Home}

$D_{\text{Cafeteria}}$ = {55min, 98min, ..., where $L(i, i+1) = (c_a, x)$ = Workplace, Cafeteria}

Note that in the previous example, we assumed that length of the context is equal to 1, i.e. $k = 1$. When adding the arrival time to the context such as shown in tables 4.3

Table 4.3 Example of visit history taking into account the arrival time when the next location is Home.

Work Place			Home	
Time Slot	AT	RT	AT	RT
10	10.45pm	90min	12.15pm	45min
1	1.10pm	320min	6.30pm	90min
...	...	...	...	...
...	...	...	...	...
3	3.10pm	195min	6.25pm	120min
10	10.50am	?	?	?

Table 4.4 Example of visit history taking into account the arrival time when the next location is Cafeteria.

Work Place			Cafeteria	
Time Slot	AT	RT	AT	RT
12	12.15am	55min	1.10pm	20min
5	5pm	98min	7.32pm	30min
...	...	...	...	...
...	...	...	...	...
12	12pm	?	?	?

and 4.4, taking a quantized arrival time in one-hour buckets, and applying the definition or residence times expressed in Eq. (4.3), we can obtain various arrival-time dependent residence time sets.

For example, if we take the arrival time as equal to 10am for a next place equals to home and another arrival time equals to 12 for a next place equal to Cafeteria, we obtain the following residence time sets $D_{10am,\ Home}$ and $D_{12pm,\ Cafeteria}$.

$$D_{10am,\ Home} = \{90\text{min where } L(i, i+1) = (c_{a,10am}, x) = \text{Workplace, Home}\} \quad (4.4)$$

$$D_{12pm,\ Cafeteria} = \{55\text{min where } L(i, i+1) = (c_{a,12pm}, x) = \text{Workplace, Cafeteria}\}$$

4.3.2 Residence Time Prediction

After the construction of the residence time sets, we focus on predicting the residence time in the current location for each possible next location. We apply the following algorithms:

- The order-k average duration predictor which predicts the next duration value of the sequence to be the average of the previous k values in the sequence.
- The order-k CDF duration predictor with probability p to predict the residence time in the current location. The probability p expresses the desired confidence in the result.

4.3.2.1 The Order-k Average Predictor

The order-k average predictor takes a sequence of previous residence times of a user at a given location, and predicts that the next residence time of the sequence is the average of the last k durations in that sequence.

Consider a set of residence times $D_{x,t}$ as defined in Eq. (4.3). We assume that $D_{x,t} = \{d_1, d_2, \ldots, d_n\}$. The order-k average predictor estimates the next residence time to be as follows:

$$\hat{d}_{n+1} = \frac{1}{m} \sum_{i=1}^{m} d_{n-i+1} \tag{4.5}$$

where $m = \min\{k, n\}$. Note that we used m that is the smallest value between k and n to reflect the fact that if we have not enough history to reach k elements in the set of residence times $D_{x,t}$, we take all the values contained in $D_{x,t}$, i.e. all values of $D_{x,t}$.

Algorithm 5 The Order-k Average Predictor.

Input: H, k
Output: the average duration Av

1: $n \leftarrow LENGTH(H)$
2: $sum \leftarrow 0$
3: $i \leftarrow 1$
4: **if** $k > n$ **then**
5: $\quad k \leftarrow n$
6: **end if**
7: **while** $i \leq k$ **do**
8: $\quad sum \leftarrow sum + d_{n-i+1}$
9: $\quad i \leftarrow i + 1$
10: **end while**
11: $Av \leftarrow \frac{sum}{k}$

4.3.2.2 The Order-k CDF Predictor

Similarly to the order-k average predictor, we start by constructing the set $D_{x,t}$. The order-k CDF predictor takes the last k values of the set $D_{x,t}$ and computes the probability that the next residence time value is less than (or greater than) a given value. Assume that D is the random variable that outputs the actual values in $D_{x,t}$. The order-k CDF predictor computes the probability that next residence time, i.e. d_{n+1}, is less than a given value d.

$$\begin{aligned}\hat{d}_{n+1} &= \arg\min_{d} \left(\Pr(D < d) \geq p\right) \\ &= \arg\min_{d} \left(\frac{1}{m}\sum_{i=1}^{m} I(d_{n-i+1} < d) \geq p\right)\end{aligned} \tag{4.6}$$

where $m = \min\{k, n\}$ and I is the indicator function. We can also use this algorithm in a way that takes the last k values from a sequence of values and a given probability P (the desired confidence in the result), and outputs the value of d that satisfies the probability P.

Algorithm 6 CDF-Prediction.

Input: H, d, k
Output: the probability P

1: $n \leftarrow LENGTH(H)$
2: $c \leftarrow 0$
3: $i \leftarrow 1$
4: **if** $k > n$ **then**
5: $\quad k \leftarrow n$
6: **end if**
7: **while** $i \leq k$ **do**
8: $\quad$ **if** $d_{n-i+1} > d$ **then**
9: $\quad\quad c \leftarrow c + 1$
10: $\quad$ **end if**
11: **end while**
12: $p \leftarrow \frac{c}{k}$

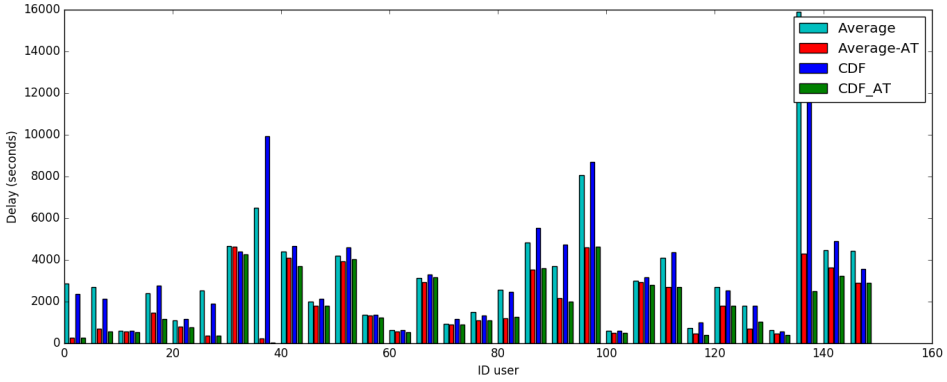

Fig. 4.1 Comparison of the four predictors according to the time difference metric. The k-CDF and k-moving have been first proposed in Song, Deshpande, Kozat, Kotz and Jain (2006).

4.3.3 Results

In this section, we evaluate the performance of our approach presented in Section 4.3 according to time difference metric. We compare the performance of our models k-CDF-AT (k-CDF Arrival Time) and k-moving-average-AT (k-moving-average Arrival Time) with k-CDF and k-moving-average proposed in Song, Deshpande, Kozat, Kotz and Jain (2006) which have been proposed by analysing the same dataset we worked on in this thesis.

Our results have been obtained with a quantization of one hour bucket. They show that some predictors provide better results than others for some situations. Some predictors work better for certain users or for some particular locations.

Figure 4.1 has been plotted by randomly taking a subset of users. It shows that the accuracy of a given predictor depends on the user and that the residence time of some users is difficult to predict. However, even with this difficulty, the results plotted in the figure show that our proposed models k-CDF-AT and k-moving-average-AT always provide a better accuracy even for those users whom the residence time is difficult to predict.

In Figure 4.2, we show the cumulative distribution of the time difference metric obtained with the four tested prediction models. We show that our models k-CDF-

AT and k-moving-average-AT constantly provide a smaller time difference between the predicted and the real residence time compared to the other models. For example, with our models we achieve less than 5000 seconds prediction error for about 80% of the users whereas the other models achieve that accuracy, i.e. 5000 seconds, for a much lesser percentage of users (only about 60% of users residence times are predicted with that accuracy). Overall, the presented results show that our proposed model improve the accuracy of existing ones by a 20% margin on the average.

In Figure 4.3, we explore the performance of two different orders of moving-average predictors: O(5) and O(10) with a recent history of a 5 and 10 residence times respectively. The results show that an O(5) moving-average-AT predictor with one hour interval yields the best performance over all other tested models. The results also show that an O(5) moving-average predictor was more accurate than an O(10) moving-average

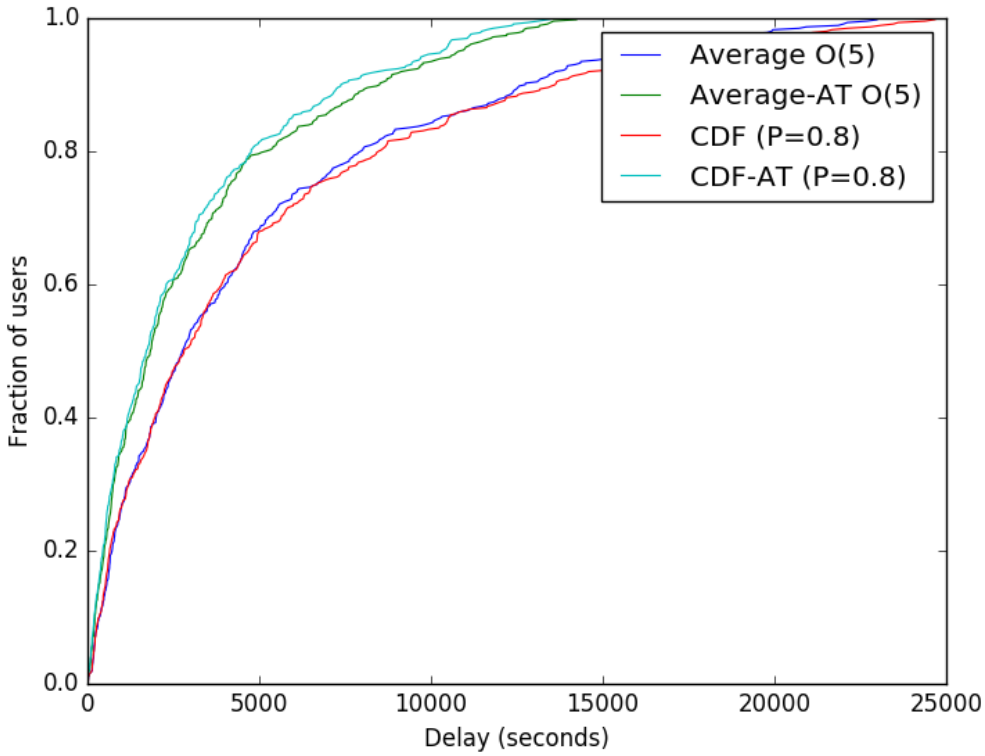

Fig. 4.2 Comparison of the four predictors according to the time difference metric. The values used for the context length k is 5 and the value used for the desired confidence p is 0.8.

one. Similarly, an O(5) moving-average-AT was better than an O(10) moving-average-AT. These results confirm existing findings that lower-order moving-average predictors, with or without arrival times, perform better than higher-order ones.

In Figure 4.4, we evaluate the performance of k-CDF predictors with and without arrival time. We considered two values $p = 0.2$ and $p = 0.8$ for the desired confidence. The results also show that higher values of p yield better results. For example, for a $p = 0.8$, the predicted value is larger than 80% of the existing residence times and thus there is a higher chance (about 80%) that the predicted value match the real value.

These results show that our models relying on the integrating the arrival time in the prediction of the residence time significantly reduce the time difference between the predicted and the real residence times compared to traditional methods which not include the arrival time in the prediction.

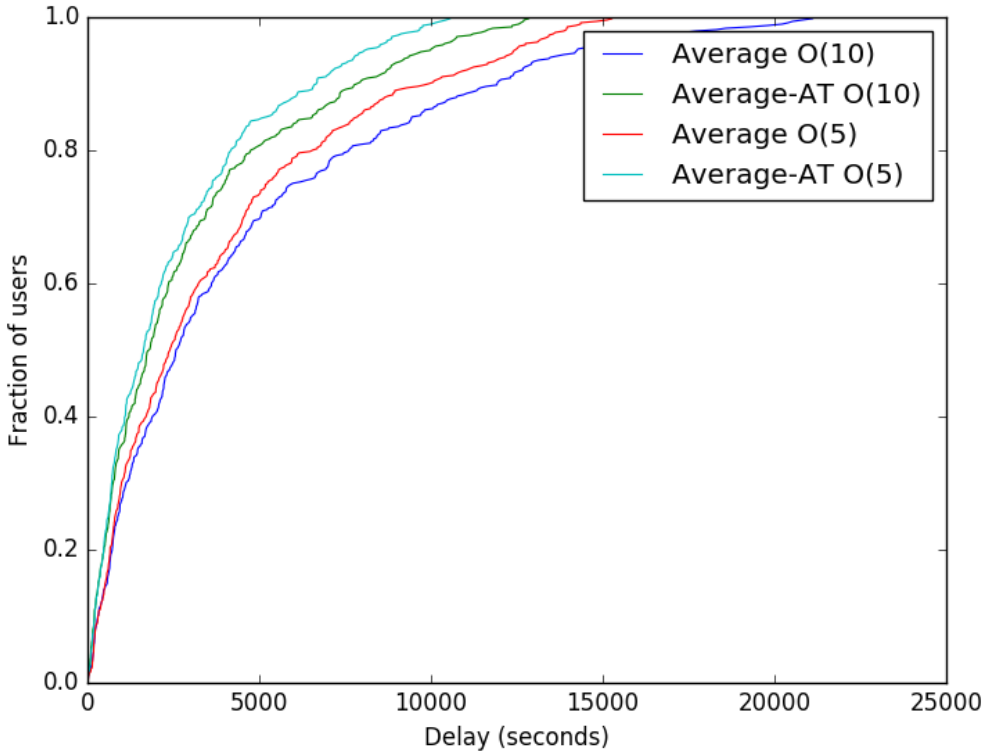

Fig. 4.3 Comparison of the two k-moving average and k-moving-average-AT predictors according to various context lengths ($k = 5$ and $k = 10$

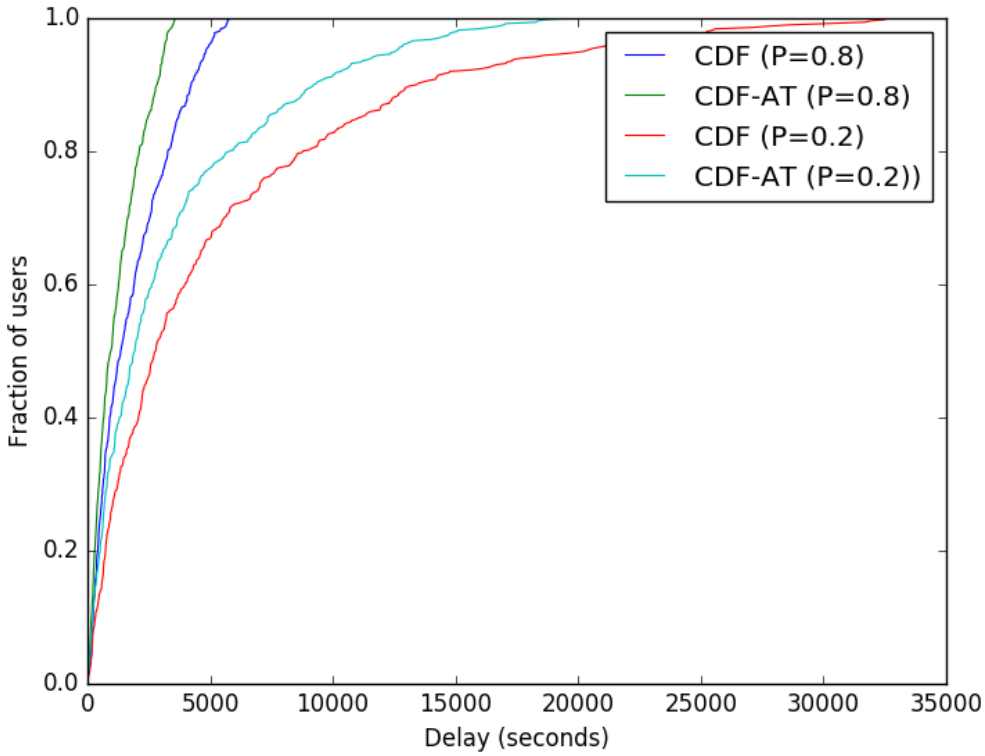

Fig. 4.4 Comparison of the two k-CDF and k-CDF-AT predictors according to various desired confidence values ($p = 0.2$ and $p = 0.8$)

4.4 Second System Model

In this section, we follow a different approach and focus on using regression-based learning algorithms to predict the residence time of a user at its relevant location. We specifically build models using Linear Regression (LR) and Auto Regression (AR) by considering both linear combination of previous residence times and other spatial or temporal features as well. Our first model LR uses a linear combination of several temporal features, and our second mode AR uses a linear combination of the k immediate past values of residence times and is also called Order-k AR and noted as (O(k)AR). Our work differs from previous works Song, Deshpande, Kozat, Kotz and Jain (2006), Sassi et al. (2017), Scellato et al. (2011) mainly by the way we use a linear combination of spatial and temporal features to predict the residence time. We evaluated our proposal by comparing their performance against existing models such Order(k)-Moving-Average (O(k)-MA) and Order(k)-Cumulative-Distribution-Function (O(k)-CDF) proposed in Song, Deshpande, Kozat, Kotz and Jain (2006), Sassi et al. (2017), and non-linear (NL) time series analysis method proposed in Scellato et al. (2011). In our evaluation, we worked on two real mobility traces from the CRAWDAD project: one provided by Dartmouth College Kotz et al. (2009), and another one called the "Ile Sans Fils" Wireless Network Lenczner and Hoen (2015). These two datasets have been constructed by recording WiFi associations of mobile users with access points — with the location of a user is taken to be that of the access point with which it is associated. The obtained results on these two benchmark datasets showed that our proposals LR and AR provide significant improvement of the prediction accuracy compared to the aforementioned state-of-the-art methods.

4.4.1 Regression-Based Residence Time Prediction

4.4.1.1 System Model and Assumptions

We assume in this work that, at any given time, a user resides at a given discrete location. In our data, the location is expressed as the name of an access point with which the

user's device associated (see Section 4.5).

In our work, we propose to make use of two types of models: Linear Regression-based (LR) models, and Autoregression-based (AR) models. Both models can be trained on the history of visits of an individual and used to predict its residence time. To predict the residence time of a user at a relevant location, we consider only the history of visits to that same location. We particularly focus on three parameters: the arrival time which allows to derive other additional temporal features, the residence time at that location, and the current location.

For a given user u at a given location l, we define the visit history $\mathcal{H}_{ul}$ as the set of the residence times spent by user u at location l as the following:

$$\mathcal{H}_{ul} = \{(t_1, d_1), (t_2, d_2), \ldots, (t_n, d_n)\} \tag{4.7}$$

where d_i is the residence time spent by the same user u at location l at time t_i.

4.4.1.2 Linear Regression-Based Prediction

The Linear Regression (LR) model aims at finding the optimal function h_α that aims at approximating the following equation for all values $\mathcal{H}_{ul}$ for a given user at a given location such that:

$$d_i = h_\alpha(t_i) \tag{4.8}$$

Given a temporal information t, many features can be extracted such as time of the day, day of the week, type of the day, etc. In the general case, let us assume that n features could be extracted from the temporal information, and use the following variables $x^{(1)}, \ldots, x^{(n)}$ to represent these features. Therefore, Eq. (4.8) can be rewritten as:

$$d = h_{\boldsymbol{\alpha}}(x^{(1)}, \ldots, x^{(n)}) \tag{4.9}$$

With linear regression for function $h_{\boldsymbol{\alpha}}$, Eq (4.9) can be rewritten as:

$$d_i = \alpha_0 + \alpha_1 x_i^{(1)} + \cdots + \alpha_n x_i^{(n)} \tag{4.10}$$

$$= \boldsymbol{\alpha}^T \mathbf{x_i} \tag{4.11}$$

where $\mathbf{x_i} = \left(1, x_i^{(1)}, ..., x_i^{(n)}\right)^T$ and $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, ..., \alpha_n)$. By finding the optimal value for $\boldsymbol{\alpha}$, Linear Regression presented in Eq. (4.11) can be used as a method to make prediction for next residence time d given a set of input features $\mathbf{x}$. To find the optimal value for $\boldsymbol{\alpha}$, represented by $\hat{\boldsymbol{\alpha}}$, the model can be trained on a set of m instances $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}$ of input values with the corresponding labels $d_1, d_2, \cdots, d_m$, respectively. Therefore, given an input instance $\mathbf{x_{m+1}}$, the corresponding output value $\hat{d}_{m+1}$ can be predicted according to the following equation:

$$\hat{d}_{m+1} = \hat{\boldsymbol{\alpha}}^{\boldsymbol{T}} \mathbf{x_{m+1}} \tag{4.12}$$

Finding the best set of parameters $\hat{\boldsymbol{\alpha}}$ for function $h_{\boldsymbol{\alpha}}$ with m training examples, i.e. $(\mathbf{x_1}, d_1), (\mathbf{x_2}, d_2), \ldots, (\mathbf{x_m}, d_m)$, can be obtained by minimizing the mean squared error function $J(\boldsymbol{\alpha})$ defined as follows:

$$J(\boldsymbol{\alpha}) = \frac{1}{2m} \sum_{i=1}^{m} \left(\boldsymbol{\alpha}^{\boldsymbol{T}} \mathbf{x_i} - d_i\right)^2 \tag{4.13}$$

Finding the optimal parameter $\hat{\boldsymbol{\alpha}}$ that minimizes $J(\boldsymbol{\alpha})$, i.e. $\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha})$ can be done with a Gradient Descent (GD) algorithm or using Normal Equation (NE) technique as shown in Section 4.4.2.

Figure 4.5 shows an example of a Linear Regression model $h_{\boldsymbol{\alpha}}(t)$ with one input variable t after getting an optimal value of the parameter $\hat{\boldsymbol{\alpha}}$ by training the model on a set of m training examples, i.e, $(t_1, d_1), (t_2, d_1), \ldots, (t_m, d_m)$.

Figure 4.5 shows how the fitted line $h_{\boldsymbol{\alpha}}(t)$ can be used as predictor by assigning a value d to each input value of the variable t.

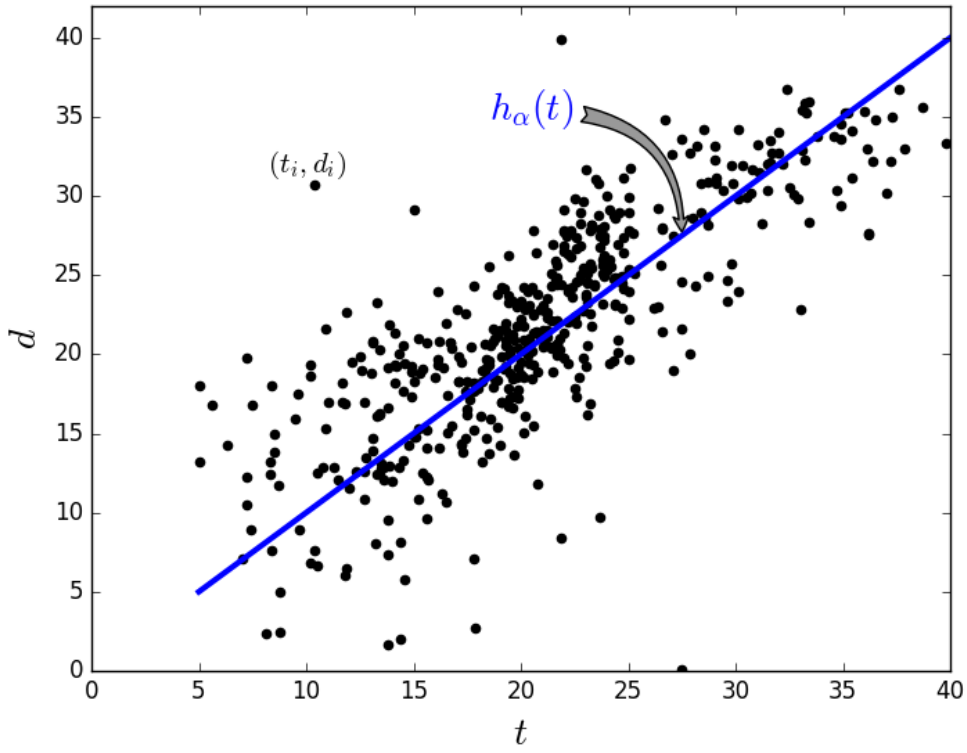

Fig. 4.5 The best fitting straight line after getting the optimal parameter $\hat{\boldsymbol{\alpha}}$.

4.4.1.3 Auto Regression-Based Prediction

By contrast to LR that is based on input values, the Auto Regression (AR) predictor is solely based on building relations between successive output values, residence times in our case. Consequently the prediction of residence time $\hat{d}_{m+1}$ is calculated from previous values $d_m, d_{m-1}, \ldots, d_1$. Order($k$) AR model performs prediction according to the following equation:

$$d_i = h_{\boldsymbol{\beta}}(d_{i-1}, d_{i-2}, \ldots, d_{i-k}) \tag{4.14}$$

where the output variable d at time i is defined by a function $h_{\boldsymbol{\beta}}$ of the k immediate past values. Assume that $h_{\boldsymbol{\beta}}$ is a linear function and consider the history of values $d_1, d_2, \ldots, d_m$. The O(k) AR model $h_{\boldsymbol{\beta}}$ will be then written as:

$$d_i = \beta_0 + \beta_1 d_{i-1} + \beta_2 d_{i-2} + \cdots + \beta_k d_{i-k} \tag{4.15}$$

for $i \in \{k+1, \ldots, m\}$. This means that the next value is a linear weighted sum of the k immediate past values. By putting $\mathbf{d_i} = (d_{i-1}, d_{i-2}, \ldots, d_{i-k})^T$, and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)^T$, Eq. (4.15) can be rewritten as:

$$d_i = \boldsymbol{\beta}^T \mathbf{d_i} \tag{4.16}$$

Also, by defining D and d as follows:

$$D = \begin{pmatrix} d_k & \cdots & d_1 \\ d_{k+1} & \cdots & d_2 \\ \vdots & \vdots & \vdots \\ d_{m-1} & \cdots & d_{m-k} \end{pmatrix}, d = \begin{pmatrix} d_{k+1} \\ d_{k+2} \\ \vdots \\ d_m \end{pmatrix}$$

we have:

$$d = D\boldsymbol{\beta} \tag{4.17}$$

Given a history of p observations (we take $p = m-k$ for the sake of simplicity and without loss of generality), $\boldsymbol{\beta}$ may be estimated by minimizing the squared error function $J(\boldsymbol{\beta})$ defined as follows:

$$\begin{aligned} J(\boldsymbol{\beta}) =& \frac{1}{2p} \sum_{i=k+1}^{m} (d_i - h_{\boldsymbol{\beta}}(d_{i-1}, \ldots, d_{i-k}))^2 \\ =& \frac{1}{2p} \sum_{i=k+1}^{m} (d_i - \beta_0 - \beta_1 d_{i-1} - \cdots - \beta_k d_{i-k})^2 \\ =& \frac{1}{2p} (D\boldsymbol{\beta} - d)^T (D\boldsymbol{\beta} - d) \\ =& \frac{1}{2p} \|D\boldsymbol{\beta} - d\|^2 \end{aligned} \tag{4.18}$$

After estimating the parameters $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta})$ by using the Gradient Descent (GD) or the Normal Equation (NE) methods as shown in Section 4.4.2, the predicted

value can be computed by Eq. (4.16) as follows:

$$\hat{d}_{m+1} = \hat{\boldsymbol{\beta}}^T \mathbf{d_{m+1}} \tag{4.19}$$

4.4.2 Minimizing Squared Error

4.4.2.1 Using Gradient Descent

As has been described before, The Gradient Descent (GD) method can be used to find and update values of $\boldsymbol{\theta}$.

For a given value θ_j (with $j = 1, \ldots, n$) of the vector $\boldsymbol{\theta}$, the algorithm GD operates as the following:

$$\theta_j^{\text{new}} := \theta_j^{\text{old}} - \lambda \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \tag{4.20}$$

When $J(\boldsymbol{\theta})$ is taken as the mean squared error function, we have for LR (where $\boldsymbol{\theta}$ is substituted by $\boldsymbol{\alpha}$):

$$\frac{\partial}{\partial \alpha_j} J(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\alpha}}(\mathbf{x_i}) - d_i \right) x_i^{(j)} \tag{4.21}$$

and we have for AR (where $\boldsymbol{\theta}$ is substituted by $\boldsymbol{\beta}$):

$$\frac{\partial}{\partial \beta_j} J(\boldsymbol{\beta}) = \frac{1}{p} \sum_{i=k+1}^{m} \left(h_{\boldsymbol{\beta}}(\mathbf{d_i}) - d_i \right) d_{i-j} \tag{4.22}$$

4.4.2.2 Using Normal Equation

Rather than needing to run an iterative algorithm that takes multiple iterations of GD to converge to the global minimum, we can instead just solve for the optimal value for $\boldsymbol{\theta}$ analytically using the Normal equation method. So that in basically one step the optimal value of the parameters $\boldsymbol{\theta}$ can be easily computed analytically and is given in

the LR case by:

$$\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T d \tag{4.23}$$

where X and d are defined as the following:

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_1^{(n)} \\ 1 & x_2^{(1)} & \cdots & x_2^{(n)} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_m^{(1)} & \cdots & x_m^{(n)} \end{pmatrix}, d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{pmatrix}$$

and it is given in the AR case by :

$$\hat{\boldsymbol{\theta}} = (D^T D)^{-1} D^T d \tag{4.24}$$

where D and d are defined as follows:

$$D = \begin{pmatrix} d_k & \cdots & d_1 \\ d_{k+1} & \cdots & d_2 \\ \vdots & \vdots & \vdots \\ d_{m-1} & \cdots & d_{m-k} \end{pmatrix}, d = \begin{pmatrix} d_{k+1} \\ d_{k+2} \\ \vdots \\ d_m \end{pmatrix}$$

4.4.3 Applying LR and AR on Selected Features

4.4.3.1 With LR

LR takes into account input variables and tries to build a model that links those input variables with the output variables. The input variables can have several features that resulting of a vector of variables. In our case we consider temporal information t_i

(as defined in Eq.(4.9)) from which we extract two features $x_i^{(1)}$ presenting the time of the day quantized into intervals of 1 hour and $x_i^{(2)}$ day of the week. There is also the possibility to extract another variable to represent type of the day, whether it is a working day, week-end, bank holiday, season holiday, etc. We choose to restraint our study to the previously mentioned features. Other features can be extracted and fed to our model as it is designed to be as general as possible and is able to take many feature variables. Under this choice, input values t_i can be used to construct input variables $\mathbf{x_i} = (1, x_i^{(1)}, x_i^{(2)})^T$ where $x_i^{(1)} \in \{0, \ldots, 23\}$ and $x_i^{(2)} \in \{0, \ldots, 6\}$. The intuitions behind choosing those two features are the following:

1. The time of the day: people tend to spend similar residence times at similar times of day. For example, assume that we have a user that arrived at their workplace at 9am. The user is most likely to leave work place at the end of the work shift, typically at around 5pm. In this case the residence time is about 8 hours. Similar behaviour would be observed in restaurants, department stores, etc.

2. The day of the week: people tend to spend similar residence times at similar places in similar days of the week. For example, if an individual periodically goes to the gym on Mondays and Thursdays, their mobility pattern would be affected for those particular days.

4.4.3.2 With AR

AR does not try to find a relation between input and output variables but it builds a model for successive observations of a variable. Therefore building the AR model consists in using residence times d_i as described in (Eq. (4.15)) and use the development described in Section 4.4.1.3 and find optimal value for $\hat{\boldsymbol{\beta}}$ using GD or NE.

4.4.4 Results

In this section, we evaluate the performance of the approaches presented in the previous sections according to time difference metric which measures the difference between the predicted residence time and the real one. We compare the performance of

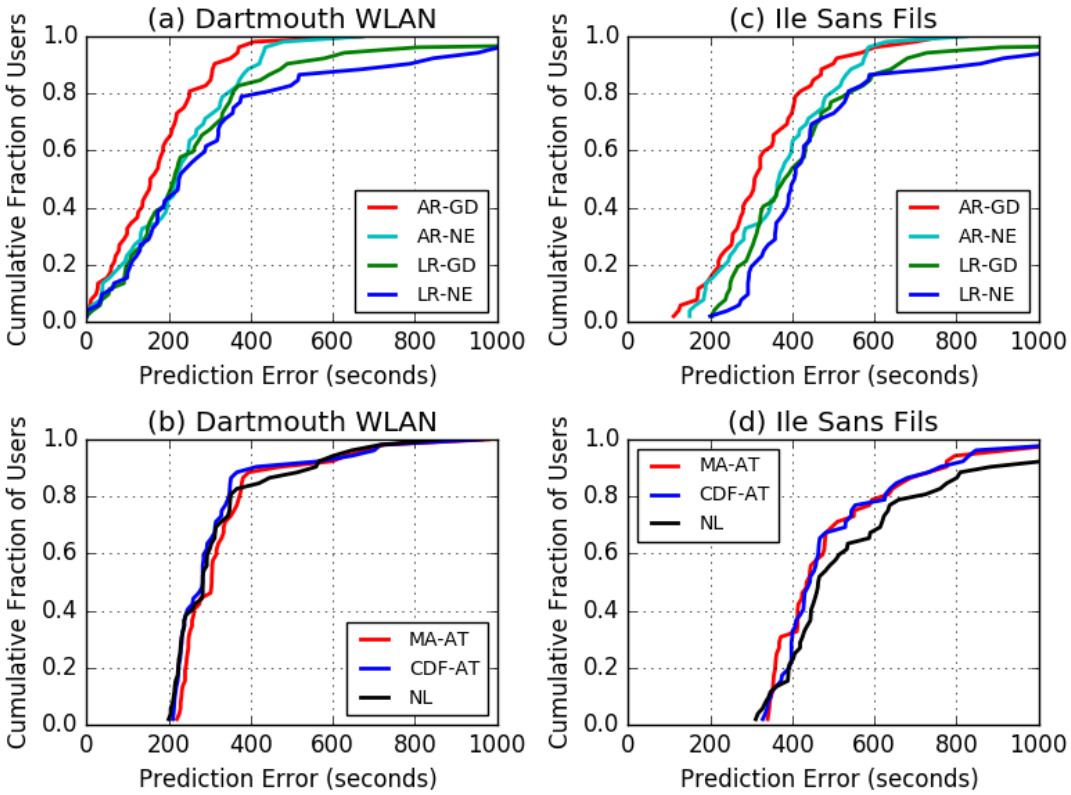

Fig. 4.6 Cumulative Distribution of the fraction of users according to the prediction error achieved on them for the Dartmouth WiFi dataset and the Ile Sans Fils dataset.

our models: O(k)-AutoRegression, Linear Regression with O(k)-moving-average and O(k)-CDF time-aided algorithms proposed in Sassi et al. (2017), and with nonlinear time series analysis method proposed in Scellato et al. (2011). For the sake of conciseness, we use the term "the five algorithms" to refer to AutoRegression (AR), Linear Regression (LR), Moving-Average-Arrival-Time (MA-AT), CDF Arrival-Time (CDF-AT), and NonLinear time series (NL). Wherever it not explicitly stated, the value of default value used for our experiments for k is 8.

In Figure 4.6, we show the cumulative distribution of the prediction errors of 100 users obtained with our prediction algorithms compared to CDF-AT and MA-AT proposed in Sassi et al. (2017), and NL proposed in Scellato et al. (2011). We show that our proposed models AR-GD, AR-NE, LR-GD and LR-GD (with GD standing for Gradient Descent and NE for Normal Equation) provide better accuracy compared to the others. The results presented in the Figure 4.6 show that our models can achieve a prediction

error in the order of seconds and minutes compared to the other algorithms. For example in the Dartmouth WiFi dataset, with our models we achieve less than 60 seconds prediction error for about more than 10% of the users, and less than 200 seconds prediction error for about 40% to 65% of the users, whereas the other models do not achieve that accuracy, i.e. a prediction error of 200 seconds, at all for all the users. All predictions errors with the other models are above 200 seconds (ses Figure 4.6(b)). However, in the case of Ile Sans Fils dataset we achieve between 120 and 250 seconds prediction error for about more than 10% of the users, and less than 400 seconds prediction error for about 40% to 65% of the users which are better results than those provided by NL, CDF-AT, and MA-AT as shown in Figure 4.6(c) and Figure 4.6(d).

These results confirm existing findings that using linear regression and autoregression models perform better than using a simple moving-average models for predicting residence time of a user. Overall, the presented results with the Dartmouth WiFi dataset (resp. Ile Sans Fils dataset) show that our proposed models improve the prediction error of at least 40% (resp. 30%) of users (see Figure 4.6(a) and Figure 4.6(c)) as 40% (resp. 30%) of users have their prediction error below 200 seconds (resp. 300 seconds) which is not obtained for any user with other algorithms. The results plotted in the Figure 4.6 also show that AR-GD model achieves the best performance over all other tested models proposed in this chapter. We also show that GD models achieve lower prediction errors than NE ones with LR-GD achieving better than LR-NE, and AR-GD achieving better than an AR-NE. We see in Figure 4.7 that for the two datasets, AR and LR models are always outperforming the other methods.

Figure 4.7(a) and Figure 4.7(c) show the median prediction error of TOP-n relevant places for the four AR, LR, MA-AT and NL models. they show also that, for all TOP-n places, AR has a lower prediction error than LR which in turn has a lower error than MA-AT and NL.

In Figure 4.7(b) and Figure 4.7(d), we show that all considered models provide relatively low prediction errors for some relevant places compared to others. O(k) AR provides the lowest prediction error for all the relevant places compared to the other

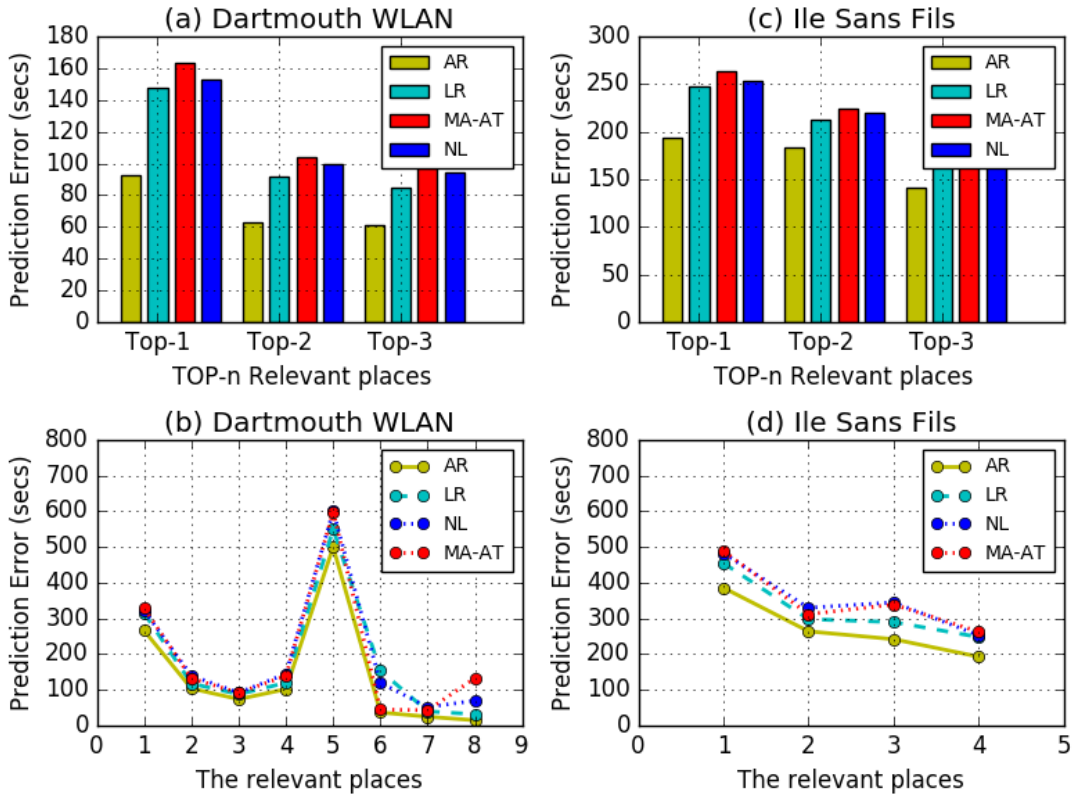

Fig. 4.7 Comparison of Autoregression, Linear Regression, NonLinear and Moving-Average models for the two datasets on different relevant places of a sample user's history.

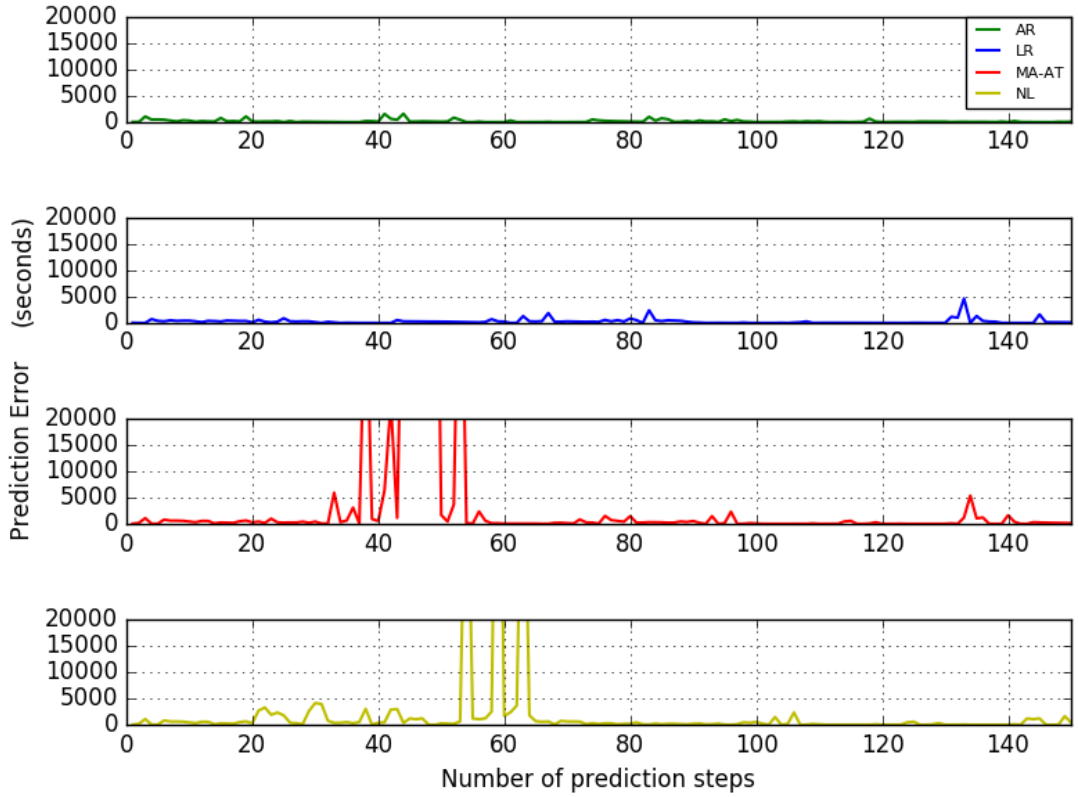

Fig. 4.8 Prediction Error variation for a sample users history.

considered models. We also show that, with the Dartmouth WiFi dataset, LR achieves a lower error than MA-AT and NL for most of the cases (in 7 out of the 8 cases considered). While with Ile Sans Fils dataset, LR achieves a lower error than MA-AT and NL for all the relevant places. We did not plot CDF-AR in Figure 4.8 and Figure 4.7 because its values are very close to that of MA-AT and the curves are almost identical., For the remaining Figures, We present only the results obtained by the Dartmouth WiFi dataset.

In the Figure 4.8, we run the considered algorithms to make predictions and for each prediction step we measure the Prediction Error defined as the difference between the real and the predicted residence times. We show that generally LR and O(k) AR achieve lower Prediction Errors compared to NL and O(k) MA-AT (with $k = 8$). We also show that for some instances AR provides better values and for others LR provide better values, but overall AR provides lower Prediction Error.

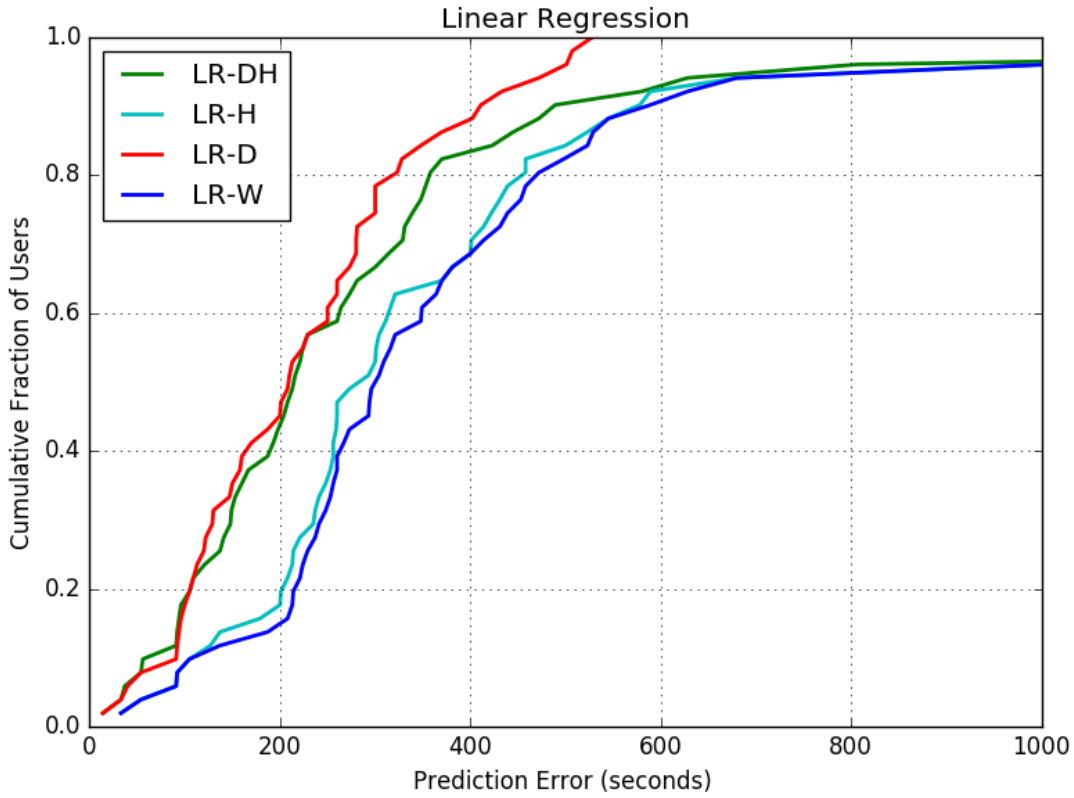

Fig. 4.9 Prediction Error of various LR models with input features including: H (Hour of the Day), D (Day of the Week), and W (Type of the Day).

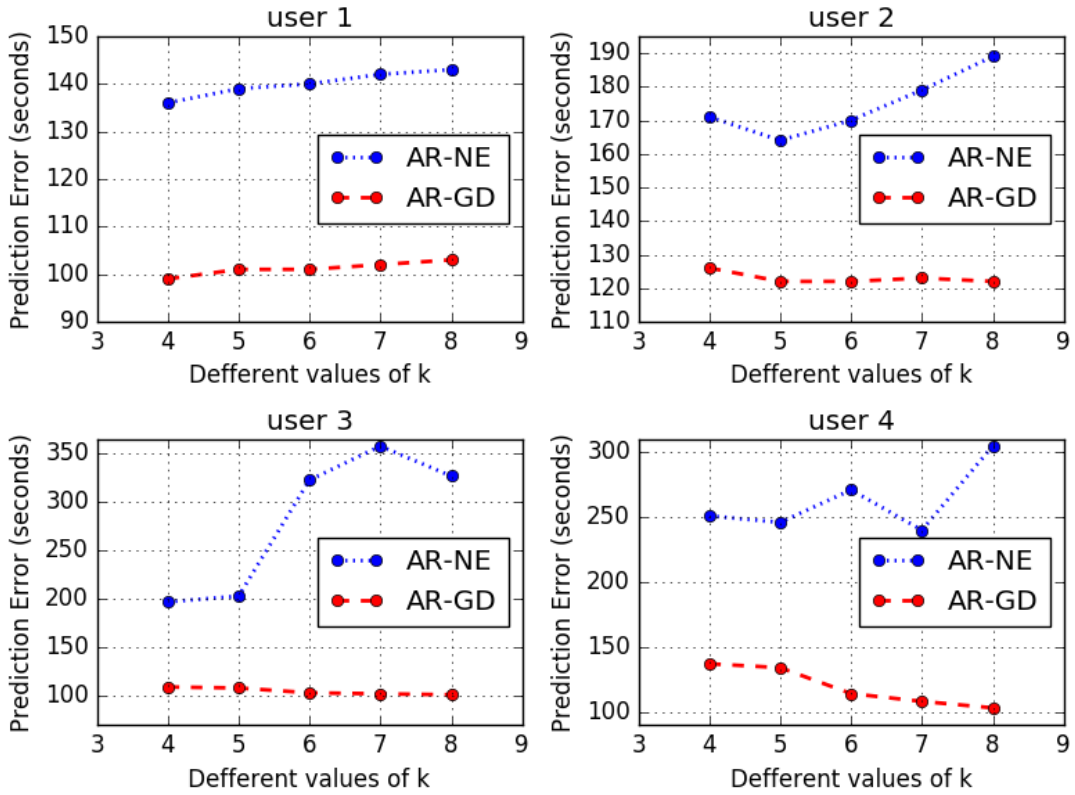

Fig. 4.10 Comparison of the two autoregression models according to the time difference metric with various values of k.

In Figure 4.9, we evaluate various LR models by considering three main input features: H (Hour of the Day), D (Day of the Week), and W (Type of the Day). The figure shows the cumulative distribution of the prediction errors of 100 users obtained with our linear regression models LR-H, LR-D, LR-W, and LR-DH. The results plotted in the Figure 4.9show that LR-D model provide a better accuracy compared to LR-H and LR-W models, and that LR-H was more accurate than LR-W. The results also show that adding a feature such as D to LR-H model, i.e LR-DH model, increases the performance of the model.

Figure 4.10 has been plotted by randomly taking four users. It explores the performance of two AR algorithms AR-GD and AR-NE models for different values of k: $k = 4, k = 5, k = 6, k = 7$ and $k = 8$ with a recent history of a 4, 5, 6, 7 and 8 residence times respectively.

It shows that AR-GD always provides a lower prediction error compared to AR-NE

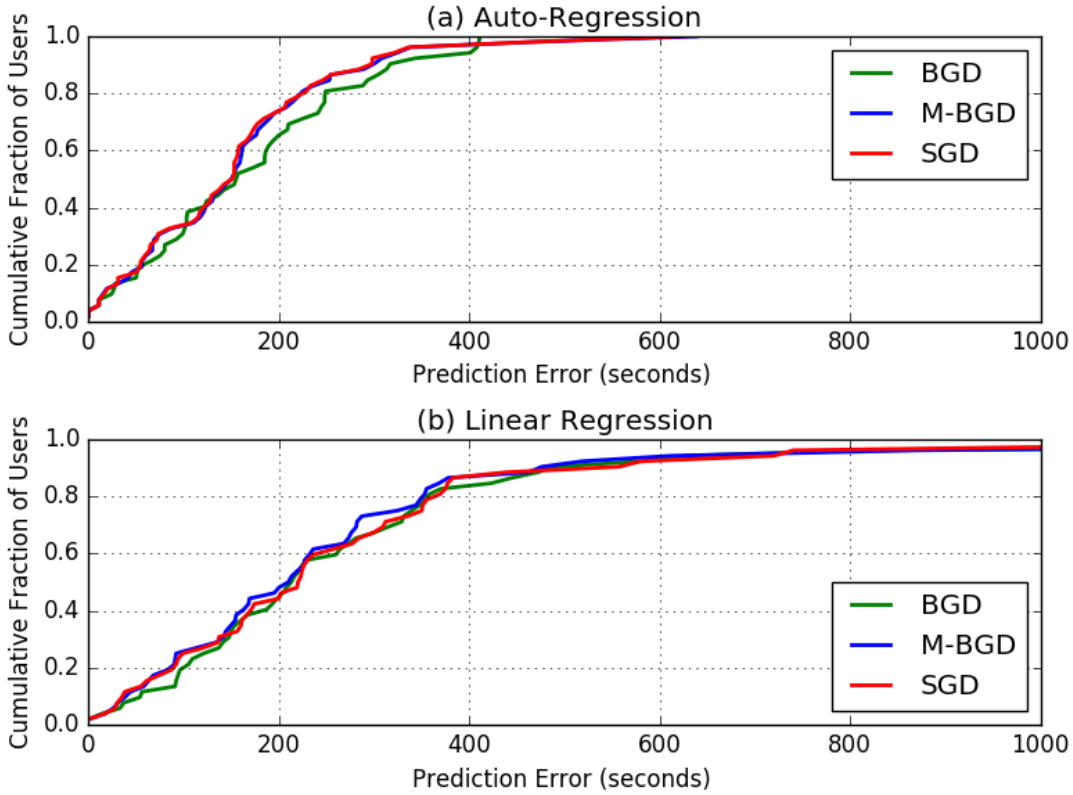

Fig. 4.11 Evaluating three variants of GD techniques: Batch GD, Mini-Batch GD, and Stochastic GD with both autoregression and linear regression models according to the prediction error metric.

model for all considered value of k. This may be explained by the way GD and NE converge to the global minimum. In fact, GD does not necessarily converge to the global minimum exactly but instead in some regions close to the global minimum. And it could be a good model when the parameters end up in some regions close to the global minimum. Figure 4.10 also shows that increasing the value of k in AR-GD provides a lower prediction error compared to AR-NE which shows deteriorating performance with higher prediction errors with higher values of k. For example, with AR-NE the prediction error for user 3 reaches 130 seconds between the two values of k ($k = 4$ and $k = 8$) whereas with AR-GD, the prediction error do not exceed 8 seconds for the same values of k. Figure 4.11 has been plotted by randomly taking a subset of 100 users. The Figure explores different variants of GD techniques, such as Batch GD, MiniBatch GD, and Stochastic GD, with both linear regression and autoregression models. In Figure 4.11(a), we compare the above three GD techniques by using autoregression model.

We show that with Stochastic GD a lower prediction errors are obtained with a higher fraction of users, compared to Batch and Mini-Batch variants. Although the curves are close to each other, it is clear that Batch GD is less accurate than both Stochastic and Mini-Batch GD and that the median user experienced an average prediction accuracy of 152, 155, and 177 seconds, for Stochastic GD, and Mini-Batch GD, and Batch GD, respectively. However, in Figure 4.11(b) we show that with linear regression models, Batch GD is slightly less accurate than Mini-Batch and Stochastic GD.

4.5 Performance Evaluation

In this section, we present the results of our experiments for predicting the residence times of users. Then we evaluate the performance of our predictors using the Prediction Error metric defined as the difference between the predicted residence time $\hat{d}$ and the real one d. First, we describe the datasets we used in our evaluation.

4.5.1 Evaluation Dataset

For the evaluation of our models, we use two different datasets. The first dataset is a subset of WiFi traces extracted from the dataset provided by Dartmouth College Kotz and Essien (2005), Henderson et al. (2008) and made available through the CRAWDAD project Kotz et al. (2009). In this dataset, each user's mobility trace is expressed as a series of (time, location) pairs where a location is taken to be that of the access point to which the user (their hand-held devices) is associated. This dataset contains more than 543 different access points resulting in more than 543 different locations. The second dataset, called "Ile Sans Fils", is composed of access points used by the City of Montreal to provide free internet access through WiFi hotspots Lenczner and Hoen (2015). In this dataset, there are more than 140 hotspots located in publicly accessible spaces such as restaurants, cafes, parks, streets, etc. so that outdoor spaces are also covered.

In our experiments, we consider relevant places. We define a place as relevant to a user, if it has been visited by that user for more than 20 times in the past. We divide

each dataset in two halves: We use the first half to serve as an initial offline training data and the second one as a live data set to test the performance of our predictor. At each prediction, a sliding window is used.

4.5.2 Eliminating the Ping-Pong Effect

Our preliminary studies of the two datasets have shown frequent re-associations of two or more neighboring access points in a short period of time repeatedly. Such behavior, called ping-pong effect, appears even if the user is not moving, which may result in a sequence of locations of the form: $\ldots, l_i, l_j, l_i, l_j, l_i, l_j, \ldots$, where a device keeps switching from location l_i to l_j and then back to l_i then again to l_j for a number of times in a short time interval. Various techniques have been used in the literature to deal with this ping-pong effect with the aim of obtaining accurate residence times Rodríguez Carrión (2015), Burbey (2011). In our case we adopt the following procedure. Consider a user with the following sequence of visits $(l_i, d_i), (l_{i+1}, d_{i+1}), (l_{i+2}, d_{i+2}), \ldots$, where d_i is the residence time spent at location l_i. We define a residence time threshold σ considered as the minimum residence time. Therefore, if $l_i = l_{i+2}$ and $d_{i+1} \leq \sigma$, we merge l_i, l_{i+1}, and l_{i+2} so the entire sequence becomes $(l_i, d_i + d_{i+1} + d_{i+2}), \ldots$, which eliminates the ping-pong effect. In our experiments we set the value of σ to 30 seconds.

In this way the sequences of durations obtained are more likely to imitate the real patterns of residence times of a users.

In order to obtain a more accurate residence time, we should eliminate this effect.

For that, we have performed pretreatments by applying a merging procedure to the two datasets. We choose a two-month subset of mobility data corresponding to the period from January to February 2003 and we did not compute the prediction error for transitions to or from the location OFF. We use the first month to serve as an initial offline training data and the second one as a live data set to test the performance of our predictor. At each prediction, a sliding window of one month is used.

4.6 Conclusions

In this chapter, we presented and evaluated several models for temporal prediction using a subset of real mobility data made available by the CRAWDAD project. We have shown that the prediction of the residence time at the current location of a given user can be significantly improved by using popular linear regression-based learning models namely: linear regression and the autoregression. We have shown that these models perform significantly better than traditional models such as simple moving-average, moving-CDF, and NL models. We have also shown that simple low-order autoregression worked at least as well or better than the linear regression for residence time at relevant place prediction, and that higher order, taking a longer history context, do not necessarily improve the prediction accuracy and rather deteriorate it. In particular with the data set we worked on, we showed that Order(5) (and above) autoregression did not improve over Order(4) autoregression models even with a large number of users. In fact, it is difficult to find the best value for k a priori as it varies from a situation to another.

We have also evaluated the performance of different variants of our proposed linear and auto regression models with two different optimization techniques Gradient Descent and Normal Equation. The resulting variants: AR-GD, AR-NE, LR-GD and LR-NE, have been compared with state-of-the-art approaches according to the prediction error metric. Our results showed that using Gradient Descent as the optimization technique to minimize the mean squared prediction error provided better results than Normal Equation, and resulted in errors in the order of seconds and minutes for a large set of users which has not been achieved by state-of-the-art methods.

There are several directions for further research. First, we have evaluated the performance of our models by using only patterns of a devices association with access points which is not human movements. Our future work will address large-scale data analysis and more sophisticated GPS-based datasets to validate our finding.

Also, it will be interesting to exploit additional data sources, such as other users or

on line social networks (e.g., Facebook, Twitter), to explore more features of human mobility. Secondly, we will address location prediction problem by investigating some deep learning techniques such as Recurrent Neural Network (RNN) as RNN is able to predict a sequence or a value at particular time point. Finally, we will use other metrics to evaluate our models.

CHAPTER 5

Conclusion and future research directions

This chapter gives a general conclusion of this thesis and some future research directions.

In this thesis, we presented and evaluated several models for next location and time prediction using a subset of real mobility data made available by the CRAWDAD project.

For next location prediction, we proposed to make use of modern machine learning techniques based on deep neural networks. We have proposed the use of Convolutional Neural Networks for which we enhanced the representation of input data by the use of embedding techniques. We have explicitly derived a new location embedding technique which we called loc2vec to enhance the quality of input location representations. Our loc2vec embedding technique improves the representation of locations by encoding close locations in mobility sequences in a way that makes their loc2vec representations also close after the embedding. We enhanced the performance of our CNN models that are based on loc2vec embedding with the use of transfer learning which allows us to take advantage of pre-trained CNN networks which we fine-tuned on our location prediction application domain. We evaluated the performance of our proposals on real mobility datasets and showed that the combination of loc2vec, CNN, and the use of transfer learning from existing CNN model provide the best results compared to popular state of the art prediction techniques relying on Markovian models.

For time prediction, We have shown that the prediction of the residence time at the current location of a given user can be significantly improved by using popular linear regression-based learning models namely: linear regression and the autoregression. We

have shown that these models perform significantly better than traditional models such as simple moving-average, moving-CDF, and NL models. We have also shown that simple low-order autoregression worked at least as well or better than the linear regression for residence time at relevant place prediction, and that higher order, taking a longer history context, do not necessarily improve the prediction accuracy and rather deteriorate it. In particular with the data set we worked on, we showed that Order(5) (and above) autoregression did not improve over Order(4) autoregression models even with a large number of users. In fact, it is difficult to find the best value for k a priori as it varies from a situation to another. We have also evaluated the performance of different variants of our proposed linear and auto regression models with two different optimization techniques Gradient Descent and Normal Equation. The resulting variants: AR-GD, AR-NE, LR-GD and LR-NE, have been compared with state-of-the-art approaches according to the prediction error metric. Our results showed that using Gradient Descent as the optimization technique to minimize the mean squared prediction error provided better results than Normal Equation, and resulted in errors in the order of seconds and minutes for a large set of users which has not been achieved by state-of-the-art methods.

There are several directions for further research. First, we have evaluated the performance of our models by using only patterns of a devices association with access points which is not human movements. Our future work will address large-scale data analysis and more sophisticated GPS-based datasets to validate our finding.

Also, it will be interesting to exploit additional data sources, such as other users or on line social networks (e.g., Facebook, Twitter), to explore more features of human mobility. Secondly, we will address location prediction problem by investigating other deep learning techniques such Transformers as Transformers can capture naturally longer term dependencies between sequence components than LSTMs and process them simultaneously. Finally, we will use other metrics to evaluate our models.

REFERENCES

Aalto, L., Göthlin, N., Korhonen, J. and Ojala, T. (2004), Bluetooth and wap push based location-aware mobile advertising system, *in* 'Proceedings of the 2nd international conference on Mobile systems, applications, and services', ACM, pp. 49–58.

Asahara, A., Maruyama, K., Sato, A. and Seto, K. (2011), Pedestrian-movement prediction based on mixed markov-chain model, *in* 'Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems', ACM, pp. 25–33.

Bai, S., Kolter, J. Z. and Koltun, V. (2018), 'An empirical evaluation of generic convolutional and recurrent networks for sequence modeling', *arXiv preprint arXiv:1803.01271* .

Baumann, P., Kleiminger, W. and Santini, S. (2013*a*), How long are you staying?: predicting residence time from human mobility traces, *in* 'Proceedings of the 19th annual international conference on Mobile computing and networking', ACM, pp. 231–234.

Baumann, P., Kleiminger, W. and Santini, S. (2013*b*), The influence of temporal and spatial features on the performance of next-place prediction algorithms, *in* 'Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing', ACM, pp. 449–458.

Bengio, Y., Courville, A. and Vincent, P. (2013), 'Representation learning: A review and new perspectives', *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1798–1828.

Boddapati, V., Petef, A., Rasmusson, J. and Lundberg, L. (2017), 'Classifying environmental sounds using image recognition networks', *Procedia Computer Science* **112**, 2048–2056.

Burbey, I. (2011), Predicting future locations and arrival times of individuals, PhD thesis, Virginia Tech.

Camacho-Collados, J. and Pilehvar, M. T. (2018), 'From word to sense embeddings: A survey on vector representations of meaning', *Journal of Artificial Intelligence Research* **63**, 743–788.

Cho, E., Myers, S. A. and Leskovec, J. (2011), Friendship and mobility: user movement in location-based social networks, *in* 'Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1082–1090.

Chon, Y., Shin, H., Talipov, E. and Cha, H. (2012), Evaluating mobility models for temporal prediction with high-granularity mobility data, *in* 'Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on', IEEE, pp. 206–212.

Chon, Y., Talipov, E., Shin, H. and Cha, H. (2011), Mobility prediction-based smartphone energy optimization for everyday location monitoring, *in* 'Proceedings of the 9th ACM conference on embedded networked sensor systems', pp. 82–95.

Gambs, S., Killijian, M.-O. and del Prado Cortez, M. N. (2010), Show me how you move and i will tell you who you are, *in* 'Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS', ACM, pp. 34–41.

Gambs, S., Killijian, M.-O. and del Prado Cortez, M. N. (2012), Next place prediction using mobility markov chains, *in* 'Proceedings of the First Workshop on Measurement, Privacy, and Mobility', ACM, p. 3.

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

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press.

He, K., Zhang, X., Ren, S. and Sun, J. (2015), 'Deep residual learning for image recognition', *CoRR* **abs/1512.03385**.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Henderson, T., Kotz, D. and Abyzov, I. (2008), 'The changing usage of a mature campus-wide wireless network', *Computer Networks* **52**(14), 2690–2712.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. (n.d.), Densely connected convolutional networks.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. and Keutzer, K. (2016), 'Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size', *arXiv preprint arXiv:1602.07360* .

Jiang, R., Song, X., Fan, Z., Xia, T., Chen, Q., Miyazawa, S. and Shibasaki, R. (2018), Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 32.

Kotz, D. and Essien, K. (2005), 'Analysis of a campus-wide wireless network', *Wireless Networks* **11**(1-2), 115–133.

Kotz, D., Henderson, T., Abyzov, I. and Yeo, J. (2009), 'CRAWDAD dataset dartmouth/campus (v. 2009-09-09)', Downloaded from `http://crawdad.org/dartmouth/campus/20090909`.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2017), 'ImageNet Classification with Deep Convolutional Neural Networks', *Commun. ACM* **60**(6), 84–90.

Laraba, S., Brahimi, M., Tilmanne, J. and Dutoit, T. (2017), '3d skeleton-based action recognition by representing motion capture sequences as 2d-rgb images', *Computer Animation and Virtual Worlds* **28**(3-4), e1782.

LeCun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

Lenczner, M. and Hoen, A. G. (2015), 'CRAWDAD dataset ilesansfil/wifidog (v. 2015-11-06)', Downloaded from `https://crawdad.org/ilesansfil/wifidog/20151106`.

Liu, Q., Wu, S., Wang, L. and Tan, T. (2016), Predicting the next location: A recurrent model with spatial and temporal contexts.

Liu, W. and Shoji, Y. (2019), 'Deepvm: Rnn-based vehicle mobility prediction to support intelligent vehicle applications', *IEEE Transactions on Industrial Informatics* **16**(6), 3997–4006.

Lv, J., Li, Q., Sun, Q. and Wang, X. (2018), T-conv: A convolutional neural network for multi-scale taxi trajectory prediction, *in* '2018 IEEE international conference on big data and smart computing (bigcomp)', IEEE, pp. 82–89.

Marmasse, N. and Schmandt, C. (2000), Location-aware information delivery with-commotion, *in* 'International Symposium on Handheld and Ubiquitous Computing', Springer, pp. 157–171.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. and Khudanpur, S. (2010), Recurrent neural network based language model, *in* 'Eleventh Annual Conference of the International Speech Communication Association'.

Minh, T. L., Inoue, N. and Shinoda, K. (2018), 'A fine-to-coarse convolutional neural network for 3d human action recognition', *arXiv preprint arXiv:1805.11790* .

Montoliu, R., Blom, J. and Gatica-Perez, D. (2013), 'Discovering places of interest in everyday life from smartphone data', *Multimedia tools and applications* **62**(1), 179–207.

Ng, A. Y. (2013), 'CS229: Machine Learning Lecture Notes', `http://cs229.stanford.edu/notes/cs229-notes1.pdf`. [Online; accessed 15-July-2020].

Noulas, A., Scellato, S., Lathia, N. and Mascolo, C. (2012*a*), Mining user mobility features for next place prediction in location-based services, *in* 'Data mining (ICDM), 2012 IEEE 12th international conference on', IEEE, pp. 1038–1043.

Noulas, A., Scellato, S., Lathia, N. and Mascolo, C. (2012*b*), A random walk around the city: New venue recommendation in location-based social networks, *in* 'Privacy, security, risk and trust (PASSAT), 2012 international conference on and 2012 international confernece on social computing (socialcom)', Ieee, pp. 144–153.

Pirozmand, P., Wu, G., Jedari, B. and Xia, F. (2014), 'Human mobility in opportunistic networks: Characteristics, models and prediction methods', *Journal of Network and Computer Applications* **42**, 45–58.

Ravi, L. and Vairavasundaram, S. (2016), 'A collaborative location based travel recommendation system through enhanced rating prediction for the group of users', *Computational intelligence and neuroscience* **2016**, 7.

Rehurek, R. and Sojka, P. (2010), Software framework for topic modelling with large corpora, *in* 'In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks', Citeseer.

Rodríguez Carrión, A. (2015), 'Contributions to the understanding of human mobility and its impact on the improvement of lightweight mobility prediction algorithms'.

Rodriguez-Carrion, A., Garcia-Rubio, C., Campo, C., Cortés-Martín, A., Garcia-Lozano, E. and Noriega-Vivas, P. (2012), 'Study of lz-based location prediction and its application to transportation recommender systems', *Sensors* **12**(6), 7496–7517.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2015), 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)* **115**(3), 211–252.

Sassi, A., Brahimi, M., Bechkit, W. and Bachir, A. (2019), Location embedding and deep convolutional neural networks for next location prediction, *in* '2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium)', IEEE, pp. 149–157.

Sassi, A., Henouda, S. E. and Bachir, A. (2017), On predicting the residence time of mobile users at relevant places, *in* 'Networks, Computers and Communications (ISNCC), 2017 International Symposium on', IEEE, pp. 1–6.

Scellato, S., Musolesi, M., Mascolo, C., Latora, V. and Campbell, A. T. (2011), Nextplace: a spatio-temporal prediction framework for pervasive systems, *in* 'International Conference on Pervasive Computing', Springer, pp. 152–169.

Scott, J., Bernheim Brush, A., Krumm, J., Meyers, B., Hazas, M., Hodges, S. and Villar, N. (2011), Preheat: controlling home heating using occupancy prediction, *in* 'Proceedings of the 13th international conference on Ubiquitous computing', ACM, pp. 281–290.

Singh, M. S., Pondenkandath, V., Zhou, B., Lukowicz, P. and Liwickit, M. (2017), Transforming sensor data to the image domain for deep learningan application to footstep detection, *in* 'Neural Networks (IJCNN), 2017 International Joint Conference on', IEEE, pp. 2665–2672.

Song, L., Deshpande, U., Kozat, U. C., Kotz, D. and Jain, R. (2006), Predictability of wlan mobility and its effects on bandwidth provisioning, *in* 'Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications', pp. 1–13.

Song, L., Kotz, D., Jain, R. and He, X. (2004), Evaluating location predictors with extensive wi-fi mobility data, *in* 'INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies', Vol. 2, IEEE, pp. 1414–1424.

Song, L., Kotz, D., Jain, R. and He, X. (2006), 'Evaluating next-cell predictors with extensive wi-fi mobility data', *IEEE Transactions on Mobile Computing* **5**(12), 1633–1649.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. et al. (n.d.), Going deeper with convolutions.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2015), 'Rethinking the Inception Architecture for Computer Vision', *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* **abs/1512.0**, 2818–2826.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2818–2826.

Taigman, Y., Yang, M., Ranzato, M. and Wolf, L. (2014), Deepface: Closing the gap to human-level performance in face verification, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1701–1708.

Vu, L., Do, Q. and Nahrstedt, K. (2011), Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace, *in* 'Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on', IEEE, pp. 54–62.

Wu, F., Fu, K., Wang, Y., Xiao, Z. and Fu, X. (2017), 'A spatial-temporal-semantic neural network algorithm for location prediction on moving objects', *Algorithms* **10**(2), 37.

Wu, R., Luo, G., Shao, J., Tian, L. and Peng, C. (2018), 'Location prediction on trajectory data: A review', *Big Data Mining and Analytics* **1**(2), 108–127.

Yue, S. (2017), 'Imbalanced malware images classification: a cnn based approach', *arXiv preprint arXiv:1708.08042* .

Zheng, X., Han, J. and Sun, A. (2018), 'A survey of location prediction on twitter', *IEEE Transactions on Knowledge and Data Engineering* **30**(9), 1652–1671.