Attaques d’inférence sur des bases de données géolocalisées
Miguel Nuñez del Prado Cortez

To cite this version:

HAL Id: tel-00926957
https://tel.archives-ouvertes.fr/tel-00926957
Submitted on 10 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
THÈSE

En vue de l’obtention du

DOCTORAT DE L’UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue par :

Miguel NUÑEZ DEL PRADO CORTEZ

le 12 Décembre 2013

Titre :

Attaque par Inférence sur des Bases des Données Géolocalisées

Inference Attacks on Geolocated Data

École doctorale et discipline ou spécialité :

ED MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de recherche :

Laboratoire d'Analyse et d'Architecture des Systèmes

Directeur(s) de Thèse :

Marc-Olivier KILLIJIAN
Sébastien GAMBS

Jury :

Président :
Abdelmalek BENZEKRI Université de Toulouse 3 - Paul Sabatier

Rapporteurs :
David GROSS-AMBLARD Université de Rennes 1 - IRISA
Yücel SAYGIN Université de Sabanci

Examinateurs :
Refik MOLVA EURECOM
Guillaume RASCHIA Polytech Nantes
# List of Figures

1. Classification des attaques par inférence. ........................................ F-3
2. Exemple de la CMM d’un utilisateur à partir de l’ensemble de données Arum. . F-6
3. Précision et prédicatibilité mesuré pour un utilisateur de Geolife. ............ F-8
4. Précision et prédicatibilité sur les trois ensembles de données. ................. F-8
5. Début-fin attaque par inférence. .................................................... F-9
6. Simple CMM avec sémantique d’un ensemble de données utilisateur dans ARUM. F-10
7. Taux de réussite des désanonymizers sur l’ensemble des données Geolife. ... F-15
8. Courbe ROC pour le jeu des données de Geolife. ................................ F-15
9. Taux de réussite des de-anonymizers sur l’ensemble de données de Nokia. ... F-16
10. Taux de réussite de stat-prox de-anonymizer lorsque les ensembles d’entraînement et de test sont les mêmes. ............................................. F-16

1.1 LBS as the crossing of technologies [Bri02]. ..................................... 12
1.2 Big data cost simulation [Tal13]. .................................................... 13
1.3 Classification and hierarchy of inference attacks. ............................... 18

2.1 Precision-recall of the four algorithms with different sampling rates. ....... 32
2.2 Precision-recall of the four algorithms with different perturbation rates. .... 33
2.3 Radial graph comparing the different clustering algorithms. ................... 40
2.4 Example of quality indices, precision, recall and F-measure. .................. 41
2.5 Correlation of quality indices with the computed F-measure. .................. 42
2.6 Flux diagram representing the method for choosing the optimal parameters of the clustering algorithm. ................................................. 43
2.7 Relation of F-measure and parameters selection based on schema in Figure 2.6. Where DBI=Davis-Bouldin index, IL=Information loss, IL_DBI= combination of IL and DBI and MAX is the maximal computed F-measure (taken as reference to compare with IL_DBI). Resume is the average of all the results using different clustering algorithms. .................................................. 44

3.1 Simple MMC of a user of the ARUM dataset. ..................................... 50
3.2 A 2nd order MMC of a user of the ARUM dataset. ............................. 51
3.3 MMC of 3 time slices divided in weekdays and weekends for a user of the ARUM dataset. ......................................................... 52
3.4 MMC of a user on ARUM dataset at different granularities. ................. 55
List of Figures

3.5 Multi-layer architecture of GEPETO. ........................................... 56
3.6 Architecture of GEPETO. ........................................................... 58
3.7 Workflow for the MapReduced version of k-means. ......................... 61
3.8 Workflow for the MapReduced version of DJ cluster. ....................... 65
3.9 Workflow for the MapReduced version of d. .................................... 68
3.10 Sampling by taking the mobility trace closest to the upper limit of the time window as the representative one. .......................... 70
3.11 Sampling by taking the mobility trace closest to the middle of the time window as the representative one. .............................. 70
4.1 A graphical representation of the 2-MMC from Bob ......................... 79
4.2 Accuracy and predictability measured for a single user of Geolife. .... 81
4.3 Accuracy and predictability on the three datasets. .......................... 81
4.4 Begin end inference attack. ......................................................... 84
4.5 Simple MMC with semantic of a user in ARUM dataset. ................ 85
5.1 Example of the optimal matching of two MMCs. The number in each node corresponds to the numbers of traces that falls inside this state. .. 97
5.2 Overview the de-anonymization attack process. ............................ 100
5.3 Distribution of the users according to their number of traces for the Geolife and Nokia datasets (the biggest datasets). ...................... 102
5.4 Distribution of the users according to their number of traces for the smallest datasets. ................................................................. 103
5.5 Number of traces per hour in training, test and complete(all) dataset. 104
5.6 Stationary distance between the “reduced” training set and the “full” training set for the Geolife and the Nokia datasets, when the size of the training set varies between 10% to 50% of the total number of traces. .......................... 105
5.7 Success rate with the simple vote function. .................................. 107
5.8 Success rate of the linear/exponential weighted votes. .................... 107
5.9 Success rate of the combined de-anonymizer. ................................ 108
5.10 ROC curve for the Geolife dataset. ............................................ 109
5.11 Success rate of the de-anonymizers on the Nokia dataset. ............... 110
5.12 Success rate of the stat-prox de-anonymizer when the training and testing sets are the same. ................................................. 111
List of Tables

1.1 Example of a mobility trace. .............................................. 10
1.2 Main characteristics of the datasets used. .......................... 14
2.1 Resume table of comparative study in [BW11]. ....................... 26
2.2 Summary of notations ..................................................... 34
2.3 Summary of input parameters for clustering algorithms. ......... 39
2.4 The characteristics of the clustering algorithms. .................. 39
3.1 $12 \times 12$ transition matrix of the graph represented in Figure 3.1. . 54
3.2 Extracted data fields from different datasets. ......................... 58
3.3 Arguments for $k$-means. ................................................. 62
3.4 Result of the MapReduce $k$-means experimentations. ............ 64
3.5 Number of traces in the sampled datasets after the preprocessing 65
3.6 Number of traces in the GeoLife dataset under different sampling conditions: no sampling, sampling rates of 1, 5 and 10 minutes. 71
4.1 Comparison of the precision, recall and F-measure between SemanPredict, Geographic-Only and Full-Matching. ............... 75
4.2 Accurate comparison of different methods ............................. 77
4.3 Transition matrix of Bob. ............................................... 79
5.1 Summary of related work. ............................................... 93
5.2 Summary of MMC distances. In this table, the complexity is measured with respect to $n$ the number of individuals in the datasets that has to be de-anonymized (for simplicity the training and testing datasets are assumed to contained the same number of individuals). 99
5.3 Illustration of the voting method. ...................................... 99
5.4 Spatial distance between training and test set using 3 different datasets. 104
5.5 Kullback-Leibler divergence between training and test sets of location datasets. 105
5.6 Clustering parameters validated. ....................................... 106
5.7 The linear and exponential weighting schemes. .................... 108
5.8 Some users are resilient to sampling. For this experiment we take into account downsampling of 5s, 10s, 30s and 60s. ...................... 109
A.1 Responsible party: Developer ........................................... 118
List of Tables

A.2 Responsible party: Agencies (Public & Private) .......................... 119
A.3 Responsible party: Users .................................................. 120
B.1 Identification of points of interests (POI). ................................. 122
B.2 Next place prediction. ..................................................... 123
B.3 Linking records and semantic extraction. ................................. 124
B.4 Discover social links and demographic attributes. ...................... 125
Contents

List of Tables iii

Résumé F-1

1 Introduction ................................................. F-1
2 Classification des attaques par inférence ......................... F-2
3 Chaîne de Markov de mobilité ................................ F-5
4 Prédire les mouvements ...................................... F-6
4.1 Evaluation expérimentale .................................... F-7
5 Extraction de la sémantique des POI ............................. F-8
5.1 Les résultats expérimentaux ................................ F-9
6 Attaque de désanonymisation .................................. F-10
6.1 Les distances entre les chaînes de Markov de mobilité .... F-11
6.2 De-anonymizers .............................................. F-13
6.3 Expériences .................................................. F-14
7 Conclusion .................................................... F-17

Foreword 1

Introduction 3

Chapter 1 Geo-privacy 7
1.1 Privacy ..................................................... 7
1.1.1 Privacy regulation ........................................ 8
1.1.2 Inference attack ........................................... 9
1.2 Location .................................................... 10
1.2.1 Location Data ........................................... 10
1.2.2 Location-Based Services ................................. 11
1.2.3 Economical value of mobility traces .................... 12
1.2.4 Dataset description ...................................... 13
1.3 Location privacy ............................................ 14
### Contents

1.4 Privacy leak .................................................. 15
   1.4.1 Inference Attacks on Geolocated Data .................. 15
   1.4.2 Inference Techniques ................................. 16

1.5 Classification of inference attacks ......................... 18
   1.5.1 Identification of Points of Interests .................. 18
   1.5.2 Next place prediction ............................... 19
   1.5.3 De-Anonymization attacks ............................. 19
   1.5.4 Extracting the semantic of POIs ....................... 19
   1.5.5 Discovering the social graph .......................... 20
   1.5.6 Predicting demographic attributes ..................... 20

1.6 Geosanitization mechanisms .............................. 20

1.7 Data utility ................................................. 22

1.8 Summary .................................................... 22

### Chapter 2 Extraction of points of interest 23

2.1 Related work on POI extraction techniques ................. 24
   2.1.1 Extraction via heuristics ............................ 24
   2.1.2 Extraction through clustering algorithms .............. 25

2.2 Clustering algorithms for extraction of points of interest .... 27
   2.2.1 Density Joinable cluster ............................. 27
   2.2.2 Density Time cluster .................................. 27
   2.2.3 Time Density cluster .................................. 27
   2.2.4 Begin-end heuristic ................................. 27

2.3 Evaluation of performance and resilience to sanitization .... 31

2.4 Cluster quality index ...................................... 34
   2.4.1 Intra-inter cluster ratio .............................. 34
   2.4.2 Additive margin ........................................ 35
   2.4.3 Information loss ....................................... 35
   2.4.4 Dunn index ............................................. 36
   2.4.5 Davis-Bouldin index ................................... 36

2.5 Selecting the optimal parameters for clustering ............ 37
   2.5.1 Precision, recall and F-measure ....................... 37
   2.5.2 Experimental results .................................. 39

2.6 Summary .................................................... 44

### Chapter 3 Mobility Markov chain and GEPETO 47

3.1 Mobility Models .............................................. 47
Chapter 3 Mobility Markov chain

3.2 Mobility Markov chain ................................................. 49

3.3 GEPETO .......................................................... 54

3.3.1 GEPETO Design .................................................. 56

3.3.2 GEPETO Implementation ....................................... 56

3.4 Mapreducing GEPETO .............................................. 59

3.4.1 Clustering algorithms ............................................ 59

3.4.2 Sanitization techniques ......................................... 70

3.5 Summary .......................................................... 71

Chapter 4 Next place prediction and semantic inference

4.1 Prediction .......................................................... 73

4.1.1 Related Work .................................................... 74

4.1.2 Next Place Prediction ........................................... 78

4.1.3 Experimental Evaluation ....................................... 79

4.2 Extraction of the semantics of POIs .............................. 82

4.2.1 Related Work .................................................... 82

4.2.2 Experimental results ........................................... 84

4.3 Summary .......................................................... 85

Chapter 5 De-anonymization and linking attack

5.1 Related work ....................................................... 88

5.2 Distances between mobility Markov chains .................. 93

5.2.1 Stationary distance ............................................. 93

5.2.2 Proximity distance .............................................. 94

5.2.3 Matching distance .............................................. 95

5.2.4 Density-based distance ......................................... 98

5.3 De-anonymizers ................................................... 98

5.4 Experiments ........................................................ 100

5.4.1 Analysis of the characteristics of the datasets ............ 101

5.4.2 Fine-tuning clustering algorithms and de-anonymizers .. 106

5.4.3 Measuring the efficiency of de-anonymizers ............... 108

5.4.4 Fair comparison with prior work ............................. 110

5.5 Summary .......................................................... 111

Chapter 6 Conclusion and Perspectives

6.1 Conclusion ........................................................ 113

6.2 Perspectives ....................................................... 115
## Contents

Appendix A Actors of privacy  
117

Appendix B Resume of the state of the art and datasets used to perform inference attacks  
121

Bibliography  
127
Résumé

Au cours des dernières années, nous avons observé le développement de dispositifs connectés et nomades tels que les téléphones mobiles, tablettes ou même les ordinateurs portables permettant aux gens d’utiliser dans leur quotidien des services géolocalisés qui sont personnalisés d’après leur position. Néanmoins, les services géolocalisés présentent des risques en terme de vie privée qui ne sont pas forcément perçus par les utilisateurs. Dans cette thèse, nous nous intéressons à comprendre les risques en terme de vie privée liés à la dissémination et collection de données de localisation. Dans ce but, les attaques par inférence que nous avons développé sont l’extraction des points d’intérêts, la prédiction de la prochaine localisation ainsi que la désanonymisation de traces de mobilité, grâce à un modèle de mobilité que nous avons appelé les chaînes de Markov de mobilité. Ensuite, nous avons établi un classement des attaques d’inférence dans le contexte de la géolocalisation se basant sur les objectifs de l’adversaire. De plus, nous avons évalué l’impact de certaines mesures d’assainissement à prévenir l’efficacité de certaines attaques par inférence. En fin nous avons élaboré une plateforme appelé GEoPrivacy Enhanced Toolkit (GEPETO) qui permet de tester les attaques par inférences développées.

Mots-clés: Attaque par inférence, respect de la vie privée, géolocalisation, désanonymisation, extraction des points d’intérêts, prédiction de la mobilité, désanonymisation.

Abstract

In recent years, we have observed the development of connected and nomad devices such as smartphones, tablets or even laptops allowing individuals to use location-based services (LBSs), which personalize the service they offer according to the positions of users, on a daily basis. Nonetheless, LBSs raise serious privacy issues, which are often not perceived by the end users. In this thesis, we are interested in the understanding of the privacy risks related to the dissemination and collection of location data. To address this issue, we developed inference attacks such as the extraction of points of interest (POI) and their semantics, the prediction of the next location as well as the de-anonymization of mobility traces, based on a mobility model that we have coined as mobility Markov chain. Afterwards, we proposed a classification of inference attacks in the context of location data based on the objectives of the adversary. In addition, we evaluated the effectiveness of some sanitization measures in limiting the efficiency of inference attacks. Finally, we have developed a generic platform called GEPETO (for GEoPrivacy Enhancing Toolkit) that can be used to test the developed inference attacks.

Keywords: Inference attack, privacy, location-based service, extraction of points of interests, mobility prediction, de-anonymization.
Résumé

1 Introduction

Aujourd’hui, en raison de l’émergence des téléphones mobiles, de la connectivité et des services basés sur la localisation, nous vivons dans un monde de connectivité ubiquitaire. Dans cet environnement, de plus en plus de traces de mobilité issues du Global Positioning System (GPS), de registre de données d’appels (CDR) ou des réseaux sociaux sont générées sur une base quotidienne à haute fréquence et automatiquement collectées sous la forme d’ensembles de données géolocalisées (Big Data). Ces données de mobilité sont collectées et générées par des applications sur des téléphones mobiles, lesquelles sont fournis par des sociétés de marketing ou des opérateurs de télécommunications et que permettent aux utilisateurs l’accès à certains services comme l’assurance par kilomètre, le suivi de congestion, trouver des POIs, connaître où sont ses amis, la navigation, etc ...

Le problème est soulevé lorsqu’une entité non autorisée peut accéder à des données de localisation. Elle peut alors les utiliser pour déduire des informations personnelles sur les individus dont les mouvements sont contenues dans ces ensembles de données, tels que déduire leur domicile ou lieu de travail, prédire des possibles localisations futures, identifier les points d’intérêt (POI) les plus fréquentés, extraire la sémantique des POIs fréquentés ou même de construire leur réseau social, provoquant ainsi une violation de la vie privée. L’adversaire pourrait même combiner des renseignements des bases de données qui apparaîtront dans l’avenir (données ouvertes), ce qui est quelque chose d’encore plus difficile à prévoir. Néanmoins, afin de protéger la vie privée des individus, certains efforts de régulation ont été réalisés dans le monde, comme en Europe, aux États-Unis, au Japon et en Australie n plus de l’effort de la Coopération économique Asie-Pacifique (APEC). Ces règles visent à renforcer la protection de la vie privée par le biais des autorités administratives nationales, telles que la “Commission nationale de l’informatique et des libertés” (CNIL) en France. Par conséquence, des normes sont proposées tel que la norme sur la vie privée comme la norme ISO/IEC 29100:2011 sur la technologie de l’information, des techniques de sécurité et d’un cadre de confidentialité [Int11], la norme AICPA/CICA en évaluation des risques en matière de protection des renseignements personnels [JAF’11], la proposition de règlement du Parlement Européen et du Conseil relative à la protection des personnes [oSS12] et la méthodologie de gestion de risques de la vie privée, “Comment mettre en œuvre la Loi sur la protection des données” [CNIon]. Toutefois, la réglementation de la vie privée ne suffit pas, un processus d’assainissement sur les données doit être effectué.

En outre, l’un de plus grands défis de ce domaine est de quantifier la conception et les
Résultats et discussion
2. Classification des attaques par inférence

![Diagram](image)

Figure 1: Classification des attaques par inférence.

objectifs allant de l’identification de la maison de la victime à la reconstruction de son réseau social. Plus précisément, les objectifs possibles, des attaques par inférence sont détaillés dans les paragraphes suivants.

**Identifier des endroits importants.** Nous considérons que l’identification des lieux importants, appelé *points d’intérêt* (POI), caractérisent la mobilité d’un individu [KSBW04]. Un POI peut être par exemple le domicile ou le lieu de travail d’un individu ou des endroits comme un centre de sport, le théâtre ou le siège d’un parti politique. Révéler les POIs d’un individu particulier est susceptible d’entraîner une violation de sa vie privée, car ces données peuvent être utilisées pour déduire des informations sensibles telles que les loisirs, les croyances religieuses, les préférences politiques ou même des maladies potentielles [LFK07]. Par exemple, si une personne a visité un centre médical spécialisé dans un type spécifique de la maladie, alors on peut déduire qu’il a une probabilité non négligeable d’avoir cette maladie.

**Prédire les mouvements.** Cette attaque prédit les *habitudes de déplacement des individus* tels que leurs emplacements passés, présents et futurs. Selon certains travaux récents [KSBW04, MD01], nos mouvements sont facilement prévisibles par nature. Par exemple, les auteurs de ces documents ont estimé à 93 % la chance de prédire correctement les futur emplacements d’un individu après une certaine période d’entraînement sur son schémas de mobilité [SQBB10].

**Désanonymiser.** L’objectif de l’attaque de désanonymisation est de relier les registres de la même personne qui peuvent être contenus dans des différents ensembles de données géolocalisées, que ce soit anonyme ou sous différents pseudonymes. Ce type d’attaque peut être considéré comme l’équivalent du *risque de divulgation statistique* où la vie privée est mesurée selon le risque de lier le record de la même personne dans deux bases de données différentes (par exemple, en établissant qu’un individu particulier dans le registre de vote est aussi un patient spécifique d’un hôpital [Swe02]). Dans un contexte géolocalisé, l’objectif de cette attaque est d’associer les mouvements d’Alice (contenus par exemple dans le jeu de données A) avec le suivi de ses emplacements de son téléphone portable (enregistrés dans un autre ensemble de données B). Étant donné que
Résumé

les points d’intérêt d’un individu et ses habitudes de déplacement constituent une forme d’empreintes digitales, en anonymisant tout simplement les données géolocalisées n’est pas suffisant comme forme de protection contre les attaques de désanonymisation. Par exemple, Colle et Kartridge [GP09] ont montré que même le pair domicile-travail devient presque unique par individu, et ils agissent donc comme une quasi-identifier, si la granularité n’est pas assez grossière (par exemple, si la rue est révélée à la place du quartier).

Extraction de la sémantique des POIs. L’objectif de l’extraction de la sémantique des POI est d’apprendre la sémantique du comportement de mobilité d’un individu à partir de la connaissance de ses POIs et ses habitudes de déplacement. Par exemple, certains modèles de mobilité tels que les trajectoires sémantiques [SPD’08, ABK’07a] ne représentent pas seulement l’évolution des déplacements d’une personne au fil du temps, mais ils attachent aussi une étiquette sémantique sur les lieux visités. A partir de cette information sémantique, l’adversaire peut réussir une meilleure compréhension des intérêts et du comportement de la mobilité d’un individu, au lieu de se baser simplement sur les habitudes de déplacement. Par exemple, l’adversaire pourrait être en mesure de conclure que sur un jour de semaine ordinaire l’individu considéré quitte généralement son domicile (POI 1) pour amener son enfant à l’école (POI 2) avant d’aller travailler (POI 3), qui est une connaissance plus profonde que le simple modèle de mouvement “ POI 1 → POI 2 → POI 3”. La sémantique peut être extraite à l’aide des heuristiques, de bases des données externes ou de la structure du modèle.

Découvrir des liens sociaux L’idée de base de cette attaque est d’exploiter les réseaux sociaux des individus et leurs habitudes de mobilité sont corrélés. Ainsi, un adversaire pourrait construire le graphe social d’une personne en utilisant ses traces de mobilité. D’une part, le réseau social peut être déduit à partir des données de localisation comme le montre le travail de Wang et co-auteurs [WGX+11]. Il est également possible d’apprendre un modèle de mobilité à partir des interactions sociales. Par exemple, Boumerdassi et co-auteurs [HBR10, CB12] ont étudié les relations entre les traces de mobilité réels et les traces de mobilité issues des interactions sociales. Finalement, ils ont proposé un modèle de mobilité qui prend en compte l’influence du réseau social.

Prédire des attributs démographiques. Les attributs démographiques peuvent être déduits grâce à la prédictibilité et l’entropie de la mobilité représentés par un modèle de Markov caché (HMM) comme le montrent le travail de Kelly, Smyth et Caulfield [KSC13]. L’objectif est de caractériser l’âge, le sexe, les tendances de voyage et les habitudes sociales des personnes. Le travail de Montjoye et co-auteurs [dMQRP13] propose une méthode pour prédire les caractéristiques de la personnalité (e.g., le neurotism, l’extraversion, la conscience, l’agrément ou l’ouverture) des utilisateurs de téléphones portables. La caractérisation se base sur le journal d’activités du téléphone.

Dans la section suivant nous nous utiliserons les POIs identifiés pour construire un modèle de mobilité.
3 Chaîne de Markov de mobilité

Une chaîne de Markov de mobilité (CMM) [GKNndPC11a] modèle le comportement de la mobilité d’un individu comme un processus stochastique discret dans lequel la probabilité de passer d’un état (i.e., POI) à un autre ne dépend que de l’état précédemment visité et de la distribution de probabilité sur les transitions entre états. Plus précisément, une CMM est composée de:

- Un ensemble d’états $P = \{P_1, \ldots, P_n\}$, dans lequel chaque état est un POI fréquent (classés par ordre décroissant d’importance), il y a une exception par rapport au dernier état $P_n$ qui peut correspondre à l’ensemble composé de tous les POI peu fréquentés. Les POIs sont extraits en exécutant un algorithme de clustering sur les traces de mobilité d’un individu. Ces états sont associés à un lieu, et généralement ils ont aussi une signification sémantique intrinsèque. Par conséquent, les étiquettes sémantiques telles que “maison” ou “travail” peuvent souvent être déduites et attachées à ces états.

- Transitions, comme $t_{i,j}$, représentant la probabilité de passer de l’état $P_i$ à l’état $P_j$. Une transition d’un état à lui-même n’est possible que si l’individu a une probabilité non nulle de passer d’un état à un emplacement occasionnel avant de revenir à cet état. Par exemple, un individu peut quitter la maison pour aller à la pharmacie et ensuite revenir à son domicile. Dans cet exemple, il est probable que la pharmacie ne soit pas extraite comme un POI par l’algorithme de clustering, sauf si la personne se rend à ce lieu sur une base régulière et y reste pendant un temps considérable.

Il faut noter que plusieurs modèles de mobilité basés sur les chaînes de Markov ont été proposés dans le passé [GKNndPC11a, AS03], y compris l’utilisation de modèles de Markov cachés pour extraire la sémantique des POIs [YCP’11a]. En un mot, la construction d’une CMM est un processus en deux étapes. Pendant la première phase, un algorithme de clustering est exécuté pour extraire les POI des traces de mobilité. Par exemple, dans le travail de Gambs et al. [GKNndPC11a], un algorithme de clustering appelé Density-Joinble Cluster (DJ-Cluster) a été utilisé (nous nous appuyons sur le même algorithme dans ce travail), mais bien sûr d’autres algorithmes de clustering sont possibles. Dans la deuxième phase, les transitions entre ces POIs sont calculées et intégrées dans le CMM.

DJ-Cluster prend en entrée un ensemble des traces de mobilité en plus de trois paramètres: le nombre minimal de points $MinPts$ nécessaires pour créer un cluster, le rayon maximum $r$ du cercle dans lequel les points d’un cluster doivent être délimités et le nombre minimal de jours pour considérer un point d’intérêt à être fréquents. DJ-Cluster fonctionne en trois phases. Pendant la première phase, qui correspond à une étape de prétraitement, toutes les traces de mobilité dans lesquelles l’individu est déplacé à une vitesse supérieure à une vitesse pré-définie ainsi que des traces redondantes sont supprimées. En conséquence, seules les traces statiques sont conservées. La deuxième phase consiste dans le clustering elle-même: toutes les traces restantes sont traitées afin d’extraire des clusters qui ont au moins $MinPts$ des points à l’intérieur d’un rayon $r$ du centre du cluster. Enfin, la dernière phase fusionne tous les groupes qui ont au moins une trace en commun. Une fois les POIs (i.e., les états de la chaîne de Markov) découverts, les probabilités des transitions entre les états peuvent être calculées. Pour réaliser
Résumé

cela, la piste des traces de mobilité est examinée dans l’ordre chronologique et chaque trace de mobilité est étiquetée avec une étiquette, soit le numéro d’identification d’un état particulier de la CMM, soit la valeur “inconnu”. Enfin, quand toutes les traces de mobilité ont été étiquetées, les transitions entre états sont comptées et normalisées par le nombre total de transitions afin d’obtenir les probabilités de chaque transition. Une CMM peut être représentée par une matrice de transition ou un graphe (cf., Figure 2) dans lequel les nœuds correspondent à des états et les flèches représentent les transitions entre les états et leur probabilité associée. Lorsque la CMM est représentée comme une matrice de transition de taille $n \times n$, les lignes et les colonnes correspondent à des états de la CMM tandis que la valeur de chaque cellule est la probabilité de transition associée entre les états correspondants.

Figure 2: Exemple de la CMM d’un utilisateur à partir de l’ensemble de données Arum.

Dans les sections suivantes, nous utiliserons les CMM pour implémenter les attaques par inférence telles que la prédiction des futurs mouvements (cf., Section 4), l’extraction de la sémantique des POIs (cf., Section 5) et la désanonymisation (cf., Section 6).

4 Prédire les mouvements

Afin de prédire le prochain emplacement sur la base des $n$ dernières positions dans le modèle CMM, nous calculons une forme modifiée de la matrice de transition dont les lignes représentent les $n$ dernières positions visitées. Ainsi, l’algorithme de prédiction a besoin en entrée des $n$ derniers précédents endroits visités en plus du modèle de mobilité CMM. L’algorithme trouve la ligne correspondant à ces $n$ emplacements précédents et recherche la transition la plus probable (les jeux décisifs sont rompus arbitrairement). S’il y a plus d’une colonne avec la même probabilité maximale, nous choisissons au hasard l’une d’entre elles.
4. Prédire les mouvements

4.1 Evaluation expérimentale

Dans cette sous-section, nous présentons les expériences menées pour évaluer la précision de l’algorithme de prédiction et la prédicatabilité théorique d’utilisateurs. Dans ces expériences, nous avons utilisé trois ensembles de données différents: Arum (phonétique), Geolife et synthétique, dont les caractéristiques sont résumées dans le Tableau 1.2 de la Section 1.2.4. Afin d’évaluer l’efficacité de nos prédicteurs de localisation, nous calculons deux mesures: la précision et la prédicatabilité. La précision $\text{Acc}$ est le rapport entre le nombre de prédictions correctes $p_{\text{correct}}$ et le nombre total de prédictions $p_{\text{total}}$:

$$\text{Précision} = \frac{p_{\text{correct}}}{p_{\text{total}}}.$$  

(1)

La prédicatabilité $\text{Pred}$ est une mesure théorique qui représente la mesure dans laquelle la mobilité d’un individu est prévisible en fonction de son CMM (dans le même esprit que le travail de Barabasi et co-auteurs [SQBB10]). Par exemple, si la prédiction de l’emplacement sait que Bob était au travail ($W$) et qu’il est actuellement à la maison ($H$), la probabilité de faire une proposition réussie est théoriquement égale à la transition de probabilité sortant maximale, qui est 84 % pour cet exemple particulier (voir le Tableau 4.3). Plus formellement, la prédicabilité $\text{Pred}$ d’une CMM particulier (et donc une personne en particulier) est calculé comme la somme des produits entre chaque élément du vecteur stationnaire $\pi$ du modèle CMM, ce qui correspond à la probabilité d’être dans un état particulier (pour $l$, le nombre total d’états de cette CMM) et la probabilité sortant maximal ($P_{\text{max, out}}$) de l’état $k$:

$$\text{Pred} = \sum_{k=1}^{l} (\pi(k) \times P_{\text{max, out}}(k, *)).$$  

(2)

Dans nos expériences, nous avons divisé chaque ensemble des traces de mobilité en deux groupes de même taille: l’ensemble d’apprentissage, utilisé pour construire la CMM, et l’ensemble d’entraînement, utilisé pour évaluer la précision du prédicteur. Enfin, nous calculons aussi le score moyen de la prédicabilité pour chaque utilisateur en fonction de la CMM La Figure 3 montre les résultats obtenus pour un utilisateur à partir de l’ensemble de données Geolife avec $n$ allant de 1 à 4. Comme prévu, la précision s’améliore lorsque $n$ augmente mais elle semble se stabiliser ou même diminuer légèrement lorsque $n > 2$. En outre, alors que sans surprise la précision de la prévision est généralement meilleure sur l’ensemble d’entraînement que sur l’ensemble de test , cette différence n’est pas significative. Cela semble indiquer que le comportement de mobilité d’un individu est similaire entre la deuxième partie de son ensemble de traces (le jeu de test) et la première partie de la trace (l’ensemble d’entraînement). Cela n’est pas nécessairement le cas si le comportement d’un utilisateur change naturellement en raison d’un changement important dans sa vie. Enfin, la Figure 4 affiche les résultats obtenus pour tous les utilisateurs des trois ensembles de données différents. Pour résumer, les résultats montrent de façon constante que la précision et la prédicabilité sont optimales (ou presque optimales) lorsque $n = 2$, avec une précision et prédicabilité allant de 70% à 95%. Comme nous pouvons le voir dans les deux métriques dans les ensembles de données Arum (Phonetic) et Geolife où ont tendance à suivre la même forme de la courbe décrite à la Figure 3. Le meilleur score est obtenu lorsque nous utilisons une CMM de deuxième ordre et en diminuant lorsque $n$ tend
Résultat à augmenter. Néanmoins, pour l’ensemble de données synthétique le score des deux indicateurs augmente lorsque \( n \) devient plus grand parce que nous observons les mêmes événements particuliers.

![Figure 3: Précision et prédicatabilité mesuré pour un utilisateur de Geolife.](image)

Figure 3: Précision et prédicatabilité mesuré pour un utilisateur de Geolife.

Jusqu’à ce point, nous avons décrit la prédiction des futurs emplacements. Une autre inférence intéressante et qui donne plus de sens à la prédiction est l’extraction de la sémantique des POI. Cette attaque est présentée dans la section suivante.

5 Extraction de la sémantique des POI

Ce type d’inférence est une attaque complémentaire à l’extraction des POIs. Plus détaillée, l’attaque d’extraction sémantique permet à l’adversaire d’avoir une connaissance significative sur les points d’intérêts dans la CMM de l’utilisateur. Néanmoins, l’extraction sémantique dépend de la connaissance de l’adversaire sur la région géographique. Toutefois, si l’adversaire n’a pas une connaissance complète de la sémantique des POIs trouvés, il peut compter sur des heuristiques afin de trouver la sémantique d’un POI à l’aide de la structure du modèle ou l’adversaire peut utiliser une base de données externe comme Open Maps Street, Google maps, Yahoo Maps, etc… comme source d’information sémantique.
5. Extraction de la sémantique des POI

5.1 Les résultats expérimentaux

La mise en œuvre de l’inférence d’extraction sémantique est double. D’une part, nous nous appuyons sur une heuristique appelée début-fin, afin de trouver la maison [GKNndPC10a]. L’attaque a été réalisée sur les données des taxis de San Francisco et elle consiste à analyser les premières et dernières traces GPS en plusieurs jours, en se basant sur le fait que les conducteurs de taxi mettent en marche et atteignent leur GPS, quand ils partent et arrivent à la maison. Comme nous pouvons le voir dans la Figure 5, en utilisant cette attaque, nous avons localisé la maison de certains conducteurs de taxi. L’inconvénient de l’ensemble de données est l’absence de réalité de terrain. Par conséquent, nous devons vérifier manuellement la maison des conducteurs de taxi, afin de trouver des preuves de la pertinence de l’attaque. Dans le bas de la Figure 5 nous avons trouvé, en utilisant Google Street View, un taxi garé dans une impasse où nous avons trouvé la maison d’un conducteur de taxi. Cet élément de preuve ainsi que la logique derrière l’attaque début-fin démontre la capacité de l’attaque.

D’autre part, la chaîne de Markov de mobilité (CMM), peut être utilisée pour déduire le domicile et le lieu de travail des utilisateurs. Par exemple, nous prenons la CMM dans la Figure 6, qui est le modèle de mobilité de Bob. D’une part, l’état “2” (i.e., Home[2_A] dans le graphe de la Figure 6) est un état central d’où Bob peut atteindre presque tous les états. En outre, à partir de presque tous les autres états, il est possible de passer à l’état “2”. En conséquence. Nous pouvons donc en déduire que cet état correspond à la maison. D’autre part, nous pouvons prendre les deux plus grandes valeurs du vecteur stationnaire, π = [0,02, 0,01, 0,48, 0,02, 0,02, 0,02, 0,01, 0,37, 0,05], de POI et puis ces deux endroits correspondent à la maison et au lieu de travailler, dans ce cas POI “2” est la maison et POI “7” (i.e., Home[2_A] dans le graphe de la Figure 6) est le travail.

Le dernier recours pour découvrir le sens des autres POIs pourrait être d’interroger une source de données externes, comme l’ontologie d’Open Street Map (OSMonto). Plus précisément, pour obtenir la sémantique d’un POI représenté par son medoid, qui est composé

1 wiki.openstreetmap.org/wiki/OSMonto
une coordonnée latitude et longitude, nous utilisons l’API Cloudmade pour interroger OS-Monto afin d’obtenir la sémantique, comme illustré dans la CMM de la figure 6. Dans la section suivante nous allons introduire une autre attaque par inférence pour désanonymiser des utilisateurs en utilisant la mobilité comme signature.

6 Attaque de désanonymisation

Dans cette section, nous concentrons sur une forme particulière d’attaque par inférence appelée attaque de désanonymisation, par laquelle un adversaire tente de déduire l’identité d’un individu particulier derrière un ensemble de traces de mobilité. Plus précisément, nous supposons que l’adversaire a pu observer les mouvements de certains individus au cours d’une quantité non négligeable de temps (e.g., plusieurs jours ou semaines) dans le passé lors de la phase d’entraînement. Plus tard, l’adversaire accède à un ensemble de données géolocalisées différentes contenant les traces de mobilité de certains individus observés au cours de la phase.

cloudmade.com
6. Attaque de désanonymisation

d’entraînement, en plus de quelques personnes inconnues éventuellement. Ensuite, l’objectif
de l’adversaire est de désanonymiser cette base de données (appelée ensemble de test) en la
reliant aux personnes correspondantes contenues dans l’ensemble de données d’entraînement.
Noter que en remplaçant simplement les noms réels des individus par des pseudonymes avant
de libérer un jeu de données n’est généralement pas suffisant pour préserver l’anonymat de
leurs identités, car les traces de mobilité elle-mêmes contiennent des informations qui peuvent
être liée uniquement à un individu. En outre, tout l’ensemble de données peut être assaini en
ajoutant du bruit spatial et temporel avant d’être libéré. Le risque de ré-identification par le
biais l’attaque de désanonymisation existe néanmoins. Ainsi, afin d’être en mesure d’évaluer
l’importance de ce risque, il est important de développer une méthode pour le quantifier.

6.1 Les distances entre les chaînes de Markov de mobilité

Dans cette sous-section, nous proposons deux distances différentes pour quantifier la similari-
eté entre deux chaînes de Markov de mobilité. Ces distances sont basées sur différentes carac-
téristiques des CMMs et donnent donc des résultats différents mais complémentaires. Nous
nous appuierons sur ces distances dans les sous-sections suivantes pour effectuer l’attaque de
désanonymisation.

La distance stationnaire

L’intuition derrière la distance stationnaire est que la distance entre les deux CMMs peut être
évaluée comme la somme des distances entre les états les plus proches des deux CMMs. Pour
calculer la distance fixe, les états des CMMs sont jumelés afin de réduire cette distance. En
conséquence, il est possible que l’état de la première CMM soit jumelé avec plusieurs états
de la deuxième CMM (ceci est particulièrement vrai si les CMMs sont de taille différente).
Par ailleurs, le calcul de la distance repose en grande partie sur les vecteurs stationnaires des
CMMs. Plus précisément, le vecteur stationnaire d’une CMM est un vecteur colonne \( V \), obtenu
en multipliant plusieurs fois un vecteur initialisé uniformément \( V_{\text{initialisation}} \) par la matrice de
transition \( M \) jusqu’à convergence (i.e., jusqu’à ce que la distribution des valeurs dans ce vecteur
attend la distribution stationnaire de la CMM).

La distance stationnaire est directement calculée à partir des vecteurs stationnaires de deux
CMMs (d’où son nom). Plus précisément, étant donné deux CMMs, les matrices de transition \( M_1 \) et \( M_2 \), les vecteurs stationnaires \( V_1 \) et \( V_2 \) de chaque modèle sont calculés, respecte-
ivement. Ensuite, l’Algorithme 1 est exécuté sur ces deux vecteurs fixes. Pour chaque état en \( V_1 \),
l’algorithme recherche l’état le plus proche en \( V_2 \), puis il multiplie la distance entre ces deux
états par la probabilité correspondante du vecteur stationnaire l’état de \( V_1 \) considéré.

Une fois que l’algorithme a pris en compte tous les états de \( V_1 \), la valeur calculée représente
la distance à partir de \( M_1 \) vers \( M_2 \) (\( distance_{AB} \)). Cette distance n’est pas symétrique en tant que
tel. Par conséquent, afin de la symétriser, Algorithme 1 est appelé une fois de plus, mais sur \( V_2 \)
et \( V_1 \), pour obtenir la distance à partir de \( M_2 \) vers \( M_1 \) (\( distance_{BA} \)). Le résultat est symétrique
en calculant la moyenne de ces deux distances.
**Résumé**

**Algorithm 1** Distance_stationnaire($V_1, V_2$)

1. $distance = 0$
2. **for** $i = 1$ to $n_1$ (le nombre de nœuds en $V_1$) **do**
3.     $MinDistance = 100000$ kilomètres
4. **end for**
5. **for** $j = 1$ to $n_2$ (le nombre des nœuds en $V_2$) **do**
6.     $courantDistance = \text{Euclidien}_\text{-Distance}(p_i, p_j)$
7.     **if** ($courantDistance < MinDistance$) **then**
8.         $MinDistance = courantDistance$
9.     **end if**
10. **end for**
11. $distance = distance + Prob_{V_1}(p_i) \times MinDistance$
12. **end for**
13. **return** distance

**Algorithm 2** distance_stationnaire_Symétrique($V_1, V_2$)

1. $distance_{AB} = \text{Distance}_\text{stationnaire}(V_1, V_2)$
2. $distance_{BA} = \text{Distance}_\text{stationnaire}(V_2, V_1)$
3. $distance = (distance_{AB} + distance_{BA})/2$
4. **return** distance

**Distance de proximité**

L'idée derrière la *distance de proximité* est que deux CMMs peuvent être considérées comme proches si elles partagent des états “importants”. Par exemple, si deux personnes partagent à la fois leur domicile et le lieu de travail, elles doivent être considérées comme étant très similaires.

En outre, l'importance d’un état est directement proportionnelle à la fréquence à laquelle il est visité. Par conséquent, les premiers états ordonnés, par ordre décroissant d’importance, sont comparés, puis les seconds, puis les troisièmes, et ainsi de suite. Le *score* de proximité obtenu pour partager le premier état est considéré comme deux fois plus important que le second état, qui est lui-même deux fois plus important que le troisième état, et ainsi de suite.

Étant donné deux modèles MMCs $M_1$ et $M_2$ (ordonnés de manière décroissante en fonction de leurs probabilités stationnaires), cette distance est paramétrée par un seuil $\Delta$ et un *rang*. L'objectif de ce classement est de quantifier l'importance de l'appariement de deux états à un niveau spécifique. En particulier, le plus élevé est la valeur du *rang*, le plus grand est le poids qui sera donné à ces POIs. Par exemple, pour le premier pair de points d'intérêt, nous avons mis un *rang* = 10.

L'algorithme 3 commence par vérifier pour chaque paire de nœuds entre $M_1$ et $M_2$ si la distance euclidienne entre eux est inférieure au seuil $\Delta$. Si cette condition est remplie, la valeur de *rang* est ajoutée à la valeur du *score*. Ensuite, *rang* est divisé par deux. Une fois que les deux nœuds ont été traités, la distance globale est configurée pour être l'inverse du score global si ce *score* est non-nul. Autrement, la distance émise est fixée à une valeur élevée (par exemple, 100}$-\text{F-12}$
6. Attaque de désanonymisation

La distance stationnaire est composée de la somme des distances euclidiennes d’un jumeelage des états alors que la distance de proximité est complètement différente, car elle est basée sur la sémantique derrière les CMMs. En effet, le premier état dans le modèle est inféré comme étant très représentatif de la mobilité d’un individu (par exemple, à la maison), le second comme assez représentatif (par exemple, le lieu de travail). Les individus sont considérés comme très semblables s’ils partagent ces deux endroits. Par la suite, nous allons voir comment ces distances peuvent être utilisées pour construire des de-anonymizers et comment leur diversité peut être exploitée et combinée afin d’accroître le succès de l’attaque.

6.2 De-anonymizers

Dans cette section, nous nous appuyons sur les deux distances proposées dans la sous-section précédente pour construire des prédicteurs statistiques afin d’effectuer une attaque de désanonymisation. Nous appelons un tel prédicteur, un de-anonymizer en référence à son objectif principal. Un de-anonymizer prend en entrée la CMM représentant la mobilité d’un individu et tente d’identifier dans un ensemble de CMMs anonymes, celui qui est le plus proche (i.e., le plus proche CMM en terme de distance). Par exemple, un de-anonymizer peut apprendre, à partir des données d’entraînement, la CMM représentant la mobilité d’Alice et vérifier plus tard la présence d’Alice dans le jeu de données de test. Un de-anonymizer peut être basé sur une ou une combinaison des deux distances.
• Le de-anonymizer basé sur la distance minimale estime que dans chaque ligne (i.e., une ligne correspond à un individu à désanonymiser), la CMM avec la distance minimale (i.e., la colonne) est l’individu correspondant à l’identité de la ligne. Nous avons considéré deux instances de ce déanonymizer, un pour chaque distance précédemment décrite (i.e., la distance stationnaire et la distance de proximité).

• Le de-anonymizer the stat-prox se comporte exactement comme la distance stationnaire minimale de-anonymizer, sauf lorsque la distance stationnaire est au-dessus d’un seuil donné et la distance de proximité est inférieure à sa valeur maximale (i.e., soit 100 000 km). L’idée est que si la distance minimale stationnaire est très faible, nous devrions l’utiliser. Sinon, nous nous appuyons sur la distance de proximité minime, à moins qu’il ne donne pas de résultat probant, auquel cas nous revenons à la distance minimale stationnaires.

6.3 Expériences

Description des jeux de données
Les jeux des données utilisés dans les expériences sont les suivantes:

1. Le jeu de données Geolife [ZLC’08] a été recueilli par des chercheurs de Microsoft Asie et se compose de traces GPS recueillies à partir de Avril 2007 à Octobre 2011 principalement dans la région de la ville de Shanghai. Cette base de données contient les traces de mobilité de 178 utilisateurs collectées à un taux très élevé de 1 à 5 secondes.

2. Le jeu de données Nokia [Kiu09] est le résultat d’une campagne de collecte de données effectuée dans la ville de Lausanne pour 200 utilisateurs qui a commencé en Septembre 2009 et qui a duré plus de deux ans. La vitesse à laquelle la localisation a été échantillonnée varie en fonction du niveau actuel de la batterie.


6. Attaque de désanonymisation

Figure 7: Taux de réussite des désanonymizers sur l’ensemble des données Geolife.

Figure 8: Courbe ROC pour le jeu des données de Geolife.


Dans la suite, nous nous concentrerons d’abord sur le jeu de données Geolife afin d’analyser et de comprendre le comportement, les distances et les dé-anonymizers. Plus précisément, pour chaque individu de cet ensemble de données, nous avons divisé l’ensemble des traces de mobilité en deux parties disjointes de taille similaire. La première moitié des données constitue l’ensemble d’entraînement, et sera utilisée comme la connaissance de base de l’adversaire, tandis que la seconde moitié constitue l’ensemble de test sur lequel l’attaque de désanonymisation sera menée. Par exemple, si l’ensemble des données originales d’un individu est composé de \( n \) traces de mobilité \( \{mt_1, mt_2, \ldots, mt_n\} \), il sera divisé en un ensemble d’entraînement \( \{mt_1, mt_2, \ldots, mt_{\frac{n}{2}}\} \) et un test set \( \{mt_{\frac{n}{2}+1}, mt_{\frac{n}{2}+2}, \ldots, mt_n\} \) (à des fins d’illustration, nous supposons que \( n \) est un nombre pair). Par conséquent, l’objectif de l’adversaire est de désanonymiser les individus de l’ensemble de test en les reliant à leurs homologues dans l’ensemble d’entraînement.

Mesurer l’efficacité des de-anonymizers

Pour mesurer le taux de réussite des de-anonymizers, nous avons échantillonné l’ensemble de données de Geolife à des taux différents et nous avons observé l’influence de l’échantillonnage sur le taux de réussite des dé-anonymizers. La Figure 7 montre que le taux de réussite de l’attaque avec la distance stationnaire minimale et la distance de proximité minimale varie de 20% à 40%, mais que le meilleur prédicteur est stat-prox avec des résultats allant de 35% à 40%.

À ce stade des expériences, il semble important de pouvoir comparer précisément les de-anonymizers. En effet, le taux de réussite d’une attaque de désanonymisation n’est pas le seul aspect qui doit être pris en considération. Par exemple, une stratégie possible pour chaque adversaire est de se concentrer sur les personnes faibles qui offrent une forte probabilité de réussite de l’attaque plutôt que d’être capable de désanonymiser l’ensemble des données. La mesure de la probabilité de réussite de l’attaque par inférence pour un individu donné est
Résumé

Figure 9: Taux de réussite des de-anonymizers sur l’ensemble de données de Nokia.

Figure 10: Taux de réussite de stat-prox de-anonymizer lorsque les ensembles d’entraînement et de test sont les mêmes.

semblable à avoir une sorte de mesure de confiance pour un candidat désanonymisé. L’issue de cette mesure de confiance est assez intuitive pour nos de-anonymizers. En effet, pour la distance minimale, plus faible est la distance, plus haute est la confiance.

Afin de comparer la performance des de-anonymizers, nous nous appuyons sur la notion de Receiver Operating Characteristic (i.e., courbe ROC) [Faw06]. En un mot, une courbe ROC est une esquisse graphique représentant la sensibilité (i.e., le rapport entre le taux de vrais positifs et le taux de faux positifs) pour un classificateur. L’intuition derrière cette métrique est que quand deux de-anonymizers atteignent le même taux de réussite, il faut privilégier celui qui possède la plus grande confiance. Désormais, la soi-disant courbe ROC qui (Figure 8) montre le taux de vrai positifs (TPR) par rapport au taux de faux positifs (FPR) pour la meilleure exécution des de-anonymizers, avec les candidats classés par ordre croissant. Cette courbe ROC confirme en outre que le de-anonymizer stat-prox est la meilleure alternative parmi les de-anonymizers que nous proposons.

Notre approche donne de bons résultats pour l’ensemble de données de Geolife où le taux de succès obtenu est entre 35% et 45% pour le de-anonymizer stat-prox. Afin de valider davantage l’approche, nous l’avons appliquée sur l’ensemble de données Nokia. Cette base de données dispose de 195 utilisateurs, parmi lesquels nous pouvons générer un CMM “valide” composé de plus d’un POI pour 157 utilisateurs. La Figure 9 montre que le taux de réussite varie entre 35% et 42%, avec le meilleur score obtenu à nouveau par le de-anonymizer stat-prox.

Comparaison équitable avec les travaux antérieurs

Dans cette section, nous avons présenté différentes expériences sur les attaques de désanonymisation qui conduisent à la définition d’un de-anonymizer appelé stat-prox, qui obtient un taux de réussite entre 42% et 45% sur différents jeux de données. Bien qu’à première vue, cette
7. Conclusion

La performance peut sembler être moins bonne que celle obtenue par le prédicteur de Ma et al. [MYYR10], qui monte de 60% à 90%, nous croyons que ces résultats ne sont pas directement comparables, car nous faisons clairement la différence entre les jeux d’entraînement et de test, alors que ces auteurs effectuent l’apprentissage et le test sur le même ensemble de données, induisant ainsi un biais expérimental.

En effet, nos modèles de mobilité sont construits sur l’ensemble d’entraînement, qui est disjoint de l’ensemble de test, alors que l’un des modèles de l’adversaire de Ma et al. extrait directement les traces de mobilité de l’ensemble d’entraînement. En outre, dans notre cas, les données d’apprentissage sont temporairement séparées des données de test (i.e., l’entraînement et le test ont été enregistrés à différents périodes de temps non chevauchantes) parce que l’ensemble de données a été divisée en deux parties temporellement disjointes, alors que le second modèle d’adversaire de Ma et al. choisit les informations qu’il utilise pour désanonymiser dans le même délai que les données de test sont enregistrées. Par conséquent, notre approche est très différente d’eux car notre attaque consiste d’abord à récolter des données sur la mobilité d’un individu avant, pour tenter d’identifier cette personne dans un soi-disant jeu de données anonymisé plus tard, alors que leur attaque vise à rassembler des données de localisation dans le même temps au cours duquel l’attaque de désanonymisation se produit. En plus, un paramètre important de leur attaque est le nombre d’emplacements horodatés recueillies, qui peut être comparé au nombre d’états que nous avons dans notre modèle de mobilité. En moyenne et en fonction de l’ensemble de données considérés, nous avons entre 4 et 8 états par CMM, qui correspondent à une représentation compacte du comportement de la mobilité d’un individu. Quand elles sont restreintes à une telle limite d’information en terme de nombre de données de localisation horodatées, les attaques proposées par Ma et al. ne fonctionnent pas bien avec un taux de désanonymisation d’entre 10% à 40%.

Aux fins de comparaison, nous avons mené l’attaque de désanonymisation, sans séparer les ensembles de test et d’entraînement. Les résultats obtenus dans les travaux connexes et pour le de-anonymizer stat-prox en utilisant différents ensembles de données sont résumés dans la Figure 10. Ces expériences sont, comme prévu, de sorte biaisées qu’elles conduisent à un taux proche de réussite 100% pour tous les ensembles de données. Encore une fois, nous ne prétendons pas que notre attaque de désanonymisation permettrait d’atteindre un taux de réussite de près de 100% et battre toutes les méthodes précédentes si l’expérience est dans les mêmes conditions (par exemple, l’ensemble de test a été seulement un sous-ensemble de l’ensemble d’entraînement, comme il a été échantillonné à partir de celui-ci). Nous sommes simplement en train de souligner que pour des questions d’équité, il est important de comparer les de-anonymizers avec le même réglage afin de réduire le biais expérimental, les ensembles d’entraînement et de tests doivent être clairement séparés (ce qui n’est pas le cas dans presque tous les les travaux antérieurs).

7 Conclusion

Cette section est la synthèse des conclusions pour répondre aux questions de recherche telles que:

I. Quel genre d’information pourrait être déduit à partir des ensembles de données géolo-
Résumé

calissées?

a. La détection des **points d’interêt** est effectuée en utilisant des algorithmes de clustering. Nous avons fait une analyse comparative, en termes de précision, rappel, F-mesure, temps d’exécution, complexité et nombre de paramètres nécessaires, notamment entre les algorithmes tels que DBSCAN, k-means, DT cluster, TD cluster ou DJ cluster qui effectuent cette tâche avec une précision de 74% et un rappel de 52%.

b. La **prédiction des mouvements** a été faite en utilisant le modèle de chaîne de Markov de mobilité (CMM), l’exactitude de cette prédiction est d’entre 70% à 95%. Nous avons pu également quantifier la prédictabilité d’un individu, qui est une mesure théorique basée sur la CMM, pour mesurer à quel point quelqu’un est prévisible.

c. L’identité peut être déduite par une attaque de désanonymisation, qui dépend de la distance entre les modèles de mobilité en raison de ce que nos CMM actent comme une signature. Puis, nous mesurons la similitude d’un modèle à l’autre afin de trouver la personne correspondante. Cette conclusion est précise, avec un taux de réussite de jusqu’à 45% sur des données réelles à grande échelle et même en présence d’assainissement par sous-échantillonnage des traces de mobilité.

d. La **sémantique des endroits personnels** est extraite de la structure du modèle de chaîne de Markov de mobilité ou en utilisant des heuristiques. Nous n’étions pas en mesure de quantifier la précision ni l’exactitude de cette attaque par inférence car, au mieux de notre connaissance, il n’existe pas un ensemble de données publique et disponible avec les étiquettes sémantiques des POIs des utilisateurs.

II. Comment le processus d’inférence pourrait être mise en œuvre?

a. Le **Modèle de mobilité**: les attaques par inférence présentées dans l’étude actuelle ne reposent pas sur plusieurs modèles “Ad-hoc”. En revanche, les attaques par inférence ont été mis en œuvre en utilisant un modèle de mobilité unique basé sur les chaînes de Markov de mobilité, qui est un modèle compact et représentatif capable d’effectuer toutes les attaques que nous avons dans notre classement.

b. Le **Entraînement et le test**: la manière dont l’ensemble de données est divisé en entraînement et test influe directement sur la précision et l’exactitude de l’inférence.

c. Le **calibrage des paramètres**: une méthodologie pour maximiser l’extraction de POIs basé sur des indices de qualité du cluster a été proposée.

III. Quel impact pour la vie privée?

a. Le **respect de la vie privée** ne peut être mesuré de manière globale. Il doit être effectué indépendamment pour chaque service (LBS). Plus précisément, il n’est pas possible d’évaluer de la même manière un service qui fournit des rapports météorologiques basés sur la localisation de l’utilisateur, où une trace de mobilité à grain gros est nécessaire, et un service pour localiser le restaurant le plus proche en utilisant en entrée une trace de mobilité à grain fin.
b. **L’anonymat:** remplacer simplement les noms réels des individus par des pseudonymes avant de publier un jeu de données n’est généralement pas suffisant pour préserver l’anonymat de leurs identités, car les traces de mobilité elles-mêmes contiennent des informations qui peuvent être uniquement liées à un individu.

c. **Les métriques du respect de la vie privée:** Les paramètres théoriques (e.g. prédicatibilité) et empiriques (e.g. exactitude, la précision, le rappel et l’entropie) peuvent être réutilisés pour une compréhension du respect de la vie privée dans le contexte de la géolocalisation (geoprivacité) et des risques de fuite du respect de la vie privée.

IV. **Y a-t-il une classification ou taxonomie des attaques par inférences sur des données géolocalisées?**

a. **Classification:** Nous avons établie une classification du point de vue des objectifs de l’adversaire. Cette classification prend en compte des attaques telles que l’identification des POIs, la prédiction des mouvements, relier des enregistrements, apprendre la sémantique des POIs, Découvrir des liens sociaux et prédire des attributs démographiques.

L’étude offre une perspective sur les attaques par inférences sur des données géolocalisées ainsi qu’une classification de ces inférences fondées sur l’objectif de l’adversaire. L’étude a rencontré un certain nombre de limitations, qui doivent être pris en compte, tels que:

- Le manque des grands ensembles de données accessibles au public. D’une part, en dehors des ensembles de données publiques proposées par “Crawdad” et Geolife, au mieux de notre connaissance, il n’existe pas de grands jeux des données avec une granularité GPS. D’autre part, il y a des jeux de données plus importants comme ceux publiés par Nokia au “Défi de données de mobilité” (MDC) [LGPA+12] et par Orange dans le cadre du “Challenge des données pour le développement” (D4D) [Ora13]. Néanmoins, il n’est pas possible de tester des attaques par inférences sur ces ensembles de données en raison d’un accord d’utilisation qui interdit les attaques, même s’ils sont à des fins scientifiques.

- L’absence de la vérité de terrain de la sémantique des points d’intérêt. Le seul ensemble de données que nous avons trouvé à granularité GPS et qui a des annotations de POIs est le jeu de données de “LifeMap”. Néanmoins, l’annotation est uniquement binaire, où 1 représente une trace de mobilité appartenant à un endroit personnel.

- L’utilité et la confidentialité des données de localisation dépendent de l’application et ne peuvent être quantifiées de façon globale. Par exemple, si nous utilisons une application de prévision météorologique, il est seulement nécessaire de révéler l’emplacement à une granularité de la ville (e.g., Toulouse), mais si nous avons besoin de connaître le distributeur de billets le plus proche de notre emplacement actuel, une granularité GPS est nécessaire. En conséquence, nous avons besoin d’un cas d’étude pour évaluer le respect de la vie privée de façon plus concrète.
Foreword

Privacy has become a catchword in the last few years. Due to the huge amount of personal data generated and collected automatically (Big data) many disciplines, ranging from sociology, law, databases, distributed systems, security, cryptography and data mining, have begun to study privacy in their respective contexts. In the present dissertation we are interested about knowledge extraction using inference attacks over geolocated data to assess privacy impact of Location Based Services (LBS) users. More precisely, we try to be aware and quantify the danger of inferences attacks over people’s mobility habits, which are collected in daily bases through geolocated social networks like Twitter or Facebook or LBSs, such as assisted navigation, pay as you go insurance, traffic congestion reports, to name a few.

The current dissertation is based on the set of contributions published in different workshops, conferences and journals. The set of these publications, alphabetically ordered, is listed below:

1. Show me how you move and i will tell you who you are [GKNndPC11a, GKNndPC10b] (chapters 2 and 3).
2. GEPETO: a GEo-Privacy Enhancing TOolkit [GKNndPC10a] (Chapter 3).
3. MapReducing GEPETO or Towards Conducting a Privacy Analysis on Millions of Mobility Traces [GKMndPC13] (Chapter 3).
5. Next place prediction using mobility Markov chains [GKdPC12] (Chapter 4).
6. De-anonymization attack on geolocated data [GKNndPC13, GKNndPC14] (Chapter 5).
Foreword
Introduction

Nowadays, due to the widespread development of portable electronic devices along with the development of connectivity and location-based services, we live in an ubiquitous world in which more and more mobility traces are gathered. These traces are issued from the smartphones themselves through the Global Positioning System (GPS), from a Call Detail Register (CDR), from WiFi positioning system or from geosocial networks. These mobility traces are gathered on a daily basis, at high frequency and automatically in the form of geolocated datasets (Big Data) by applications on mobile phones, marketing companies, service providers, telecommunication companies and network operators to provide users access to some services like pay as you go insurance, congestion monitoring, finding interesting places, knowing the locations of your friends, navigation, virtual loyalty cards, just to name a few. Thus, there is an economical interest to provide services based on location data. By 2015, this market is expected to generate around 21.14 billions us dollars [GIA]. At the same time, privacy issues are becoming more and more frequent. For instance, the German deputy Malte Spitz sued Deutsche Telekom to obtain the last six months of location data generated from his phone, before publishing this data in the form of an interactive map showing that the combination of location data with contextual information could lead to a serious privacy breach [onl09]. Another example of privacy buzz was the article published in The Wall Street Journal [JA11] about telephone constructors revealing that Apple and Google collect on a large scale location data using unique identifiers in order to develop novel location-based services.

Privacy issues arise when an authorized entity can access this rich and sensitive data. Then, the adversary, which is the term that we will use throughout this thesis to designate this unauthorized entity, is able to use the location data to infer personal information about the individuals whose movements are contained within these datasets, such as inferring their personal points of interest like home and place of work, predict their next location, extract the semantic of a POI or even build their social network, thus causing a privacy breach. The adversary may even combine this location information with databases that will appear in the future like the open data initiative [gou13, Gra13], which is something even more difficult to anticipate.

Nonetheless, to protect the privacy of individuals, some regulation efforts have been done, for instance, in Europe, United States, Japan, Australia and the Asia-Pacific Economic Cooperation (APEC). These privacy laws are enforced in practice by data protection authorities, such as the Commission nationale de l’informatique et des libertés (CNIL) in France. As a result, some privacy standards and methods are emerging such as the ISO/IEC 29100:2011 about Information technology, Security techniques and Privacy framework [Int11], the AICPA/CICA Privacy Risk Assessment Tool [JAF+11], the NIST Privacy Assessment [oSS12], Proposal for a Regula-
tution of the European Parliament and of the Council on the Protection of Individuals [oSS12] and the methodology for Privacy Risk Management, How to implement the Data Protection Act [CNlon]. However, privacy laws are a first step but not sufficient on their own to protect the privacy of users. In addition, privacy-enhancing technologies should be used to enforce in practice these privacy laws.

Furthermore, one of the biggest challenges is to quantify the concept of privacy, which is cultural dependent. Solove, in his Taxonomy of Privacy [Sol06], classifies people into three different groups. Those who are very concerned about their privacy and do not want to make public their personal information, those who find fair to reveal some piece of personal information in order to receive a service in exchange (e.g., a LBS that take as input a location to report the closest restaurant to the current position). The last group represents the users that do not care about their privacy and are ok with publishing all the details of their personal life. Besides, other interesting questions arise like: do users have all the information to quantify the privacy risk or impact of using a given service, what kind of information could be inferred from location their data, could inferences be automatized, what is the privacy impact and is there a classification of inference attacks?

In accordance with the 5th principle put forward by Duckham [Duc10], our objective is to perform inference attacks over location data in order to evaluate the resulting privacy leakage. Thus, we have developed inference attacks to extract the point of interests (POIs) and their semantics, to predict the next location and to de-anonymize users. Afterwards, we establish a classification of inference attacks over geolocated data. Second, we build a mobility model named Mobility Markov Chain (MMC) to implement previously classified attacks. Finally, we detail the platform (GEPETO), a flexible software that can be used to visualize, sanitize, perform inference attacks and measure the utility of a particular geolocated dataset. More precisely, this dissertation is organized as follows:

Chapter 1 describes the background concepts on location, privacy as well as geoprivacy. Afterwards, a taxonomy of inference attacks is also presented in which the attacks are classified according to the objective of the adversary. Finally, the notions of geosanitization methods and utility of data are also introduced.

Chapter 2 deals with the extraction of the points of interests characterizing the mobility of a user. The clustering algorithms, such as $k$-means, Density Time cluster, Time Density cluster and Density Joinable cluster, used to discover these points of interests are explained as well as a methodology to identify the optimal number of clusters.

Chapter 3 details the mobility model based on Markov chains used to perform inference attacks. In addition, this chapter describes the developed platform to automate the model construction, sanitization techniques and inference attacks.

Chapter 4 presents the prediction of the next location and the extraction of semantics of points of interest. Both, the next location prediction and semantic extraction inference attacks, rely on the Mobility markov chain introduced in the previous chapter.

Chapter 5 describes a de-anonymization attack over sanitized (anonymous) datasets using the
mobility Markov chain model introduced in Chapter 3. This attack exploits novel distances measuring the similarity between two mobility models.

Chapter 6 concludes the manuscript by summarizing the main points of this thesis and identifying future research tracks.
Chapter 1

Geo-privacy

“All human beings have three lives: public, private, and secret.”

— Gabriel García Márquez, a Life

The increasing use of location-based services application and social networks makes people release voluntarily (or not) a huge amount of personal data without taking into account the privacy threats. Before the arrival of ubiquitous devices, the biggest concern was about privacy disclosure on the Internet. For instance, the privacy leakage of AOL\(^3\) in 2006, in which a New York Times journalist was able to reidentify people from an anonymized dataset, containing search keywords, released by AOL for research purposes. With the widespread use of geolocated applications, new threats have arrived due to the release of location data. Accordingly, in the present effort we elucidate the basic notions from privacy to geoprivacy. In the first part of the chapter we present the concept of privacy (cf. Section 1.1). Then, the notion of location is introduced (cf. Section 1.2) to extend the definition of privacy to geoprivacy (cf. Section 1.3). Next, we present the new issues triggered by location data (cf. Section 1.4) as well as our own classification of inference attacks (cf. Section 1.5). Afterward, Section 1.6 discuss geosanitization mechanisms that can be used to protect location data. Finally, Section 1.7 discusses about the quantification of the utility of location data after a geosanitization process and Section 1.8 summarizes the present chapter.

1.1 Privacy

Privacy is an ancient concept. In ancient Greece, Socrates and other philosophers already made the differences between public (society) and private (solitude) [Hol09]. In 1361, the Justice of the Peace Act in England laid down sentences for peeping toms and eavesdroppers [SL09]. A more recent definition of privacy was proposed by Alan Westin in 1960: “Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others” [Wes70]. Each person has a

different understanding (and conception) of what privacy is. In a nutshell, privacy is the right to keep secret (protect) sensitive information. *Sensitive information* corresponds to personal data like birth date, personal phone number, location but also other data, such as industrial secrets, videos and pictures.

Privacy varies across domains and therefore can be considered as being a subjective notion. In *classical databases*, the notion of privacy is very close to anonymity. Thus, in this context, inference attacks aim to re-identify people from a database (de-anonymization attack) or link the same person present in different databases by means of quasi-identifiers, which are a set of attributes that can identify uniquely an individual in a database. As a consequence, various privacy models such as *k*-anonymity [Swe02], *l*-diversity [MKGV07], *t*-closeness [LLV07] and differential privacy [Dwo06, Dwo08] were proposed. In *computer security*, privacy is related to the confidentiality of a message or a file, protected in general through encryption or access control. Thus, the attack will attempt to decrypt the cipher text in order to extract the information it contains. In *secure multiparty computation*, privacy is associated with the information one is able to infer besides the result of the computation of some protocol. Therefore, the objective of the attack is to learn as much as possible from the protocol, by reading the messages passing among the nodes, by having a malicious behavior, by implementing a man-in-the-middle attack or by colluding with several nodes. In *social networks* privacy is defined by the amount of information – messages, photos, applications, location etc. – one shares with his friends. Therefore, the objective of the inference attack is to profile members of the network based on their features by extracting implicit information from the social graph.

Protecting privacy, when personal data is shared, whether voluntarily or not, is a difficult and complex task.

### 1.1.1 Privacy regulation

As *sensitive information* are generated on a daily basis as a result of data processing tasks, specific regulation is required to guide or to limit the use of this data. Information by itself is not “good” or “bad”, but rather the problem is how this data is processed, collected and disseminated. There are also four major models for privacy protection [GP08]:

**Comprehensive laws** are general laws governing the collection, use and dissemination of personal information by both the public and private sectors.

**Sectoral laws.** Some countries, such as the United States, have avoided the enactment of broad data protection the privacy laws in favor of specific sectoral laws.

**Self-regulation.** Data protection can also be achieved, at least in theory, through various forms of self-regulation, in which company and industry bodies establish codes of practice or codes of conduct and engage in self-policing.

**Privacy technologies.** With the recent development of commercially available technology-based systems, various aspects of privacy protection have moved into the hands of individual users.
1.1. Privacy

In 1995, the European Union enacted the Data Protection Directive to harmonize the laws of member states [GP08]. Each European country must have a data protection commissioner or agency enforcing the rules. This directive established several basic principles for European citizens. These principles include the following rights:

- The right to know where the data originated.
- The right to have inaccurate data rectified.
- The right of recourse in the event of unlawful processing.
- The right to withhold permission to use data under some circumstances.

In addition, the APEC Privacy Initiative, which is the most important international action after EU Data Protection Directive. This initiative is based on the guidelines of the Organization for Economic Cooperation and Development (OECD) on the Protection of Privacy and Transborder Flows of Personal Data (1980) as starting models and it has the potential to encourage the development of stronger privacy laws in APEC countries. As a consequence of the regulation, it is necessary for service providers to:

- Specify the knowledge that can be collected and searched from digital traces.
- Define the purposes for which digital traces can be stored, analyzed, and shared.
- Indicate who has the right to inspect stored traces of personal information.

Furthermore, some standards and privacy risk analysis related to privacy are currently being put forward. For example, the ISO/IEC 29100:2011 about Information technology, Security techniques and Privacy framework [Int11], the AICPA/CICA Privacy Risk Assessment Tool [JAF‘11], NIST Privacy Assessment [oSS12], and the Methodology for Privacy Risk Management, How to implement the Data Protection Act [CNIon]. In the following subsection, we define the concept of inference attack.

1.1.2 Inference attack

An inference attack is a data mining process by which an adversary - that could be a curious person, a marketing company [AS10], a service provider [Ric10], such as Facebook [SF10], a telecommunication operator or a malicious entity [Cox10] extract new personal information or knowledge about individuals that was implicitly present in the data. Afterward, this new knowledge can be used to perform another, more refined, attack in order to increase the knowledge gathered about the analyzed data. For instance, a famous inference attack was published on the website of "Le Tigre", in which the adversary was able to reconstruct the life of an individual named Marc L.¹ Nevertheless, when location data is added to "traditional" information, the issue becomes even more complex. Thus, in the following subsections we will introduce the definition of location, location-based services, location data, economical value of location data before describing some geolocated datasets.

¹ http://www.le-tigre.net/Marc-L.html
Chapter 1. Geo-privacy

1.2 Location

A location is a point on the earth’s surface, which could be absolute or relative. The former is designated using a geographical coordinates system (e.g., latitude, longitude and altitude). While the latter uses a place of reference to the main location (e.g., Toulouse is south-west of France). In this dissertation, we simplify the location to be a latitude and longitude coordinate.

1.2.1 Location Data

The rapid growth and development of location-based services has multiplied the potential sources of location data. The data generated by these diverse applications varies in form and content but it also shares some common characteristics. Regarding the type of data, we differentiate mainly between mobility traces and contact traces. A mobility trace is characterized by:

- An identifier, which can be the real identifier of the device (e.g., “Alice’s phone”), a pseudonym or even the value “unknown” (when full anonymity is desired). A pseudonym is generally used when we want to protect the true identity of the system while still being able to link different actions performed by the same user.

- A spatial coordinate, which can be a GPS position (e.g., latitude and longitude coordinates), a spatial area (e.g. the name of a neighborhood in a particular city) or even a semantic label (e.g., “home” or “work”).

- A timestamp, which can be the exact date and time or just an interval (e.g., between 9AM and 12AM).

- Additional information such as the speed and direction of a vehicle, the presence of other geolocated systems or individuals in the direct vicinity or even the accuracy of the estimated reported position. For instance, some geolocated systems are able to estimate the precision of their estimated location as a function of the number of GPS satellites they are able to detect.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubjectId</td>
<td>Bob</td>
</tr>
<tr>
<td>Latitude</td>
<td>32,1423</td>
</tr>
<tr>
<td>Longitude</td>
<td>-122,1719</td>
</tr>
<tr>
<td>Altitude</td>
<td>150</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>19/10/08 20:32:15</td>
</tr>
<tr>
<td>AnotherInformation</td>
<td>&quot;Restaurant&quot;</td>
</tr>
</tbody>
</table>

Table 1.1: Example of a mobility trace.

Before presenting the contact traces, in the next paragraph, an example of mobility trace is shown in Table 1.1.
1.2. Location

Contact traces are a specific form of mobility traces which consist in the recording of encounters between different devices. This kind of trace is composed of the identifiers of the devices and a time stamp. It may be recorded for instance by a device which has no integrated capacity for geopositioning but is capable of probing its neighbourhood to detect the presence of other devices (e.g., using Bluetooth neighbor discovery).

A geolocated dataset \( D \) is a dataset containing mobility traces of individuals. Technically, this data may have been collected either by recording locally the movements of each geolocated system for a certain period of time, or centrally by a server which can track the location of these systems in real-time. For instance, some trails of mobility traces could be accessed by deploying a collector application on cell phones [KRT10], using geosocial networks APIs. (i.e., Facebook [Fac12a] or Foursquare [Fou12a]), or by using publicly available datasets. or establishing agreements with telecoms operators. A trail of traces is a collection of mobility traces representing movements of an individual over some period of time. A geolocated dataset \( D \) is generally constituted by a set of trails of traces from different individuals. The Crawdad project [PSDG09] is an example of a public repository giving access to geolocated datasets, which can be used for research purpose.

1.2.2 Location-Based Services

A Location-Based Service (LBS) is an application that tailors the service provided to the current geolocated context. A LBS can be further classified based on the functionality it provides. We distinguish between 4 categories of LBS:

Geo-targeting are designed to respond to the following question: “Where am I?” Given the position of a user, equipped with a device with geolocated capabilities, this LBS shows on a map or in a social network application, the location of the user or some of the content that he has posted, such as “Bob was at Blagnac Airport”. For instance, Picasa [Goo12a], Foursquares [Fou12b], Facebook places [Fac12b] are examples of geo-targeting services.

Recommendation services provide users with suggestions about products or places near him. This type of LBS can be proactive or reactive. Examples of proactive recommendation services include geo-advertising, which pushes information to users about products or services when they reach a certain area, such as a coupon for a pizza when the user is at Capitol square (e.g., PayPal Media Network [Net12a] or Twitter [Twi12]). In contrast, a reactive recommendation service queries about the nearest place to find some service or product such as “where is the closest Peruvian restaurant?”.

Navigation system is a category of LBS computing a trajectory to go from one point to another, (e.g., the best itinerary to go from the Charles-de-Gaulle airport to the Eiffel tower). Google maps and Yahoo maps are examples of such services. The navigation system can also notify if friends of the user are in the neighborhood or passes by in order to carpool with, like in the AMORES project [ADG+12] and CooVIA [Coo12].

Monitoring systems are capable of reporting the location of crowded zones in a city. For instance, the traffic monitoring system, records the movements of users in order to output
Chapter 1. Geo-privacy

the locations like in MapCity [Map12], Tom Tom [Tom12] or TeleNav [Tel12] or the density analyzer computes dense area, like Skyhook [Sky12] or Sense Network [Net12b].

Figure 1.1: LBS as the crossing of technologies [Bri02].

LBSs are at the intersection of three technologies: Geographical Information System (GIS), Internet and Networks and Telecommunication as shown in Figure 1.1. Other LBS classifications exist such as the one presented by Scipioni and Langheinrich [SL10], friend finder in Geo-targeting category, or by Andersen [And11], who divides monitoring category in crowdsensing and city watch.

1.2.3 Economical value of mobility traces

Besides, around 67% of people tag their location in the pictures they take [Jiw12], 74% get information based on their current location, 18% share their location with friends through a geosocial network [Zic12] and 4% have proactively received mobile coupons from neighboring stores but 40% of users are interested in receiving those types of geolocated ads. In addition, data mining over location data is becoming an attractive market. Services such as mobile advertisements and mobile coupons will represent, in 2014, a discount of $7 billion for about 300 million people worldwide [BB11a] while LBS will represent US$21.14 billion by 2015 [GIA].

For marketing applications, Riederer and co-authors proposed a framework to sell personal data taking into account the user privacy [REC+11]. The main idea of this framework is to implement privacy by design by building an architecture, in which a proxy information gathers personal information. The personal data that can be collected is specified beforehand by the user. For instance, Alice defines the areas from where data collectors could take her personal information (without knowing her real identity) when she is moving in the city.

Concerning, the value and cost of geolocated (big) data Tallon [TS07], [Tal13] suggests two methods to estimate it value of data. The first one quantifies the potential economical loss when this data is not available. The second one is applied when risk cannot be measured but instead the opportunity. Thus, when companies cannot judge the value of data, they estimate instead that this data can be important some day in the future. The drawback when we think data could potentially be useful some day is that this approach encourages data retention. Another aspect
of the value is the cost of storing data, Tallon [Tal13] shows that the cost of maintaining data is not only related to hardware cost as shown in Figure 1.2, but also it is important to take into account the cost of keeping data like backups, license, maintenance. Merill [Mer12] proposes a more comprehensive list of 34 factors to take into account for quantifying cost of data storage ranging from hardware/software depreciation, purchase, maintenance to data loss, litigation (discovery risk), security, encryption and procurement.

In order to provide privacy-aware services that are still profitable for business, it is necessary to model mobility data to be able to understand the risks of disclosing mobility traces to third parties and the possible privacy leakage issues.

### 1.2.4 Dataset description

Thereafter, we describe the main characteristic of the geolocated datasets that we are going to use for the experiments in the remaining chapters of the thesis:

- The *Arum dataset* [KRT10] is composed of the GPS mobility traces from 6 researchers sampled at a rate of 1 to 5 minutes in the city of Toulouse from October 2009 to January 2011.

- The *Geolife dataset* [ZLC+08] has been gathered by researchers from Microsoft Asia and consists of GPS mobility traces collected from April 2007 to October 2011, mostly in the area of the city of Shanghai. It contains the mobility traces of 178 users captured at a very high rate of 1-5 seconds.

- The *Nokia dataset* [Kiu09] is the outcome of a data collection campaign performed in the
city of Lausanne for 185 users started in September 2009 until March 2011. The sampling rate is not specified.

- The *San Francisco Yellow Cabs* (SFD) [PSDG09] contains the GPS coordinates of 500 taxis gathered over 30 days in San Francisco Bay. The sampling rate was approximately of 5 minutes. We work with a subset of 165 randomly sampled taxis.

- The *Borlange dataset* [FSH12] has been collected as a part of traffic congestion experiment over two years from 1999 to 2001. The public version of this dataset contains the GPS traces of 24 vehicles. The sampling rate is not specified.

- The *LifeMap dataset* [CTSC12] composed of 12 was collected in Seoul, Korea. This dataset comprises location (latitude and longitude) collected with a frequency between 2 to 5 minutes with user defined point of interest.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Arum</th>
<th>Geolife</th>
<th>Nokia</th>
<th>SFC</th>
<th>Borlange</th>
<th>LifeMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total nb of users</td>
<td>6</td>
<td>175</td>
<td>185</td>
<td>165</td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td>Collection period (nb of days)</td>
<td>255</td>
<td>146</td>
<td>1443</td>
<td>23</td>
<td>4103</td>
<td>366</td>
</tr>
<tr>
<td>Average nb of traces/user</td>
<td>16 374</td>
<td>121 615</td>
<td>63 145</td>
<td>20 812</td>
<td>12 555</td>
<td>4 224</td>
</tr>
<tr>
<td>Total nb of traces</td>
<td>98249</td>
<td>21 203 955</td>
<td>11 681 896</td>
<td>34 341 00</td>
<td>301 332</td>
<td>50 685</td>
</tr>
<tr>
<td>Min #Traces for a user</td>
<td>2376</td>
<td>17</td>
<td>369</td>
<td>59</td>
<td>2806</td>
<td>307</td>
</tr>
<tr>
<td>Max #Traces for a user</td>
<td>39 787</td>
<td>1 276 221</td>
<td>515 702</td>
<td>29 742</td>
<td>34 411</td>
<td>9 473</td>
</tr>
<tr>
<td>Median nb of traces/user</td>
<td>16 762</td>
<td>24 531</td>
<td>41 537</td>
<td>23 230</td>
<td>11 422</td>
<td>3 240</td>
</tr>
</tbody>
</table>

Table 1.2: Main characteristics of the datasets used.

Table 1.2 summarizes the main characteristics of the datasets described above.

To exploit mobility data, it is necessary to use adapted representation such as trajectory or mobility models. In the next section, we extend the concept of privacy to the geolocated context.

### 1.3 Location privacy

A *geolocated system* is an object or device that has an associated location. For instance, it can be a smartphone or a GPS-equipped vehicle. Usually, a geolocated system belongs to an individual or to a group of individuals, and as such its location corresponds to the location of its owner(s).

Extending the concept of privacy of an individual to spatiotemporal data means that the sensitive data to protect is the position for a given time. In particular, Beresford and Stajano define location privacy as “the ability to prevent undesired entities from knowing one’s past, present, and future locations” [BS03].

However, geolocated data is already publicly available and sometimes easy to obtain. For instance, some persons disseminate publicly, almost in real-time, their current location via (geo)social application such as Facebook places [Fac12b], Foursquare [Fou12b] or Twitter [Dor06].
1.4 Privacy leak

In turn, this data can be collected to predict whether or not they are currently at home like with the website Please rob me [BvAG10]. Other applications, such as Google Latitude [Goo09], can be used to track the movements of friends’ cellphones and display their position on a map. Apart from these (geo)social applications, there are also other public sources of information that can be exploited by a potential adversary for causing a privacy breach, such as free and easy access to geographic knowledge with Google maps [Goo12d], Yahoo!Maps [Yah12] and Google Earth [Goo12b]. Thereafter, we introduce the entities involved in the collection of location data.

The actors involved in obtaining and processing mobility traces are the clients, who use application designed by developers. This application communicates with the service providers through the telecom operators network. Thus, privacy disclosure is possible at each step of the chain. Cottrill [Cot11] proposes a classification based on who is responsible for providing protection to such a data, which is detailed in tables A.1, A.2 and A.3 of Appendix A.

More precisely, the users or clients should establish their own privacy policy through the available privacy settings. They should refuse to use an application without a declared privacy policy and, have the minimum amount of location data if possible. Users should be conscious about the risks of location disclosure. Developers are in charge of designing privacy-aware software by applying the privacy by design approach. Service providers are the responsible for setting up access controls to guarantee information confidentiality and secure storage to prevent information leak. They are also in the best position to embed privacy in the architecture. Telecom operators should follow a self-regulatory approach in order to preserve the privacy of their clients. Finally, public and private agencies should provide regulation and guidelines about how to manipulate location data in order to ensure location privacy protection. In all the cases, there is a risk of privacy leakage, which is discussed in the next section.

1.4 Privacy leak

In this section we aim to introduce a definition as well as some techniques to perform inference attacks. These kinds of attacks, which use location data as input, allow an adversary to cause a privacy leak.

1.4.1 Inference Attacks on Geolocated Data

An inference attack is an algorithm that takes as input some geolocated data $D$, possibly with some auxiliary information $aux$, and produces as output some additional knowledge. For example, an inference attack may consist in identifying the house or the place of work of an individual. The auxiliary information reflects any a priori knowledge that the adversary might have gathered (for instance through previous attacks and by accessing some public data source) and which may help him in conducting an inference attack. In the next subsection we detail certain inference techniques.
Chapter 1. Geo-privacy

1.4.2 Inference Techniques

We describe thereafter some learning algorithms and methods that can be used as inference to gain knowledge:

- **Clustering** is a form of unsupervised learning that tries to group objects that are similar in the same cluster while putting objects that are dissimilar in different clusters. A clustering algorithm needs a *distance measure* (or a similarity metric) to quantify how far/similar are two objects relative to each other and to drive the clustering process. A natural distance between two locations is simply the Euclidean distance but of course more complex metrics can be used, such as the length of the shortest path according to the existing roadmap. For instance, *k*-means is an iterative clustering algorithm that outputs *k* clusters as well as their respective centres (which are effectively the average of the locations within each cluster). This algorithm can be used straightforwardly to discover the POIs of one particular individual if it is fed only with his data [AS03], or the generic hotspots if it is given the geolocated data of a whole population. Hoh, Gruteser, Xiong and Alrabady have performed a study [BHA06] on the geolocated data of vehicles within the Detroit area (Michigan, USA). The goal of their study was to automatically discover the home of the vehicles’ drivers. The authors have used a clustering algorithm to automatically identify the houses and their findings is that among the 2 neighbourhoods and the 65 persons on which the authors have focused, the estimated houses correspond at 85% to the houses that a human would have recognized\(^5\). More complex techniques such as density-based clustering [ZFL+04] can be used instead of *k*-means to overcome some of its shortcomings, such as *k* the predefined number of clusters and the constraint that the shape of the clusters has to be spherical.

- **Classification** is a form of supervised learning, whose main objective is to learn a function that can predict the class of an unknown data point. The classifier is learned from the training dataset in which the datapoints are already labeled with the corresponding class. For instance, the *k*-nearest neighbor classifier [ZZ05] categories by analogy. Given a set of data with different features the algorithm takes a new point and some distance metric to classify the new point based on majority vote of its *k* closest neighbors. The classifier can be used to deduce the semantic of an important place based on characteristics such as the duration of the stay, the hour of the day, the density of mobility traces. More precisely, if an adversary is able to build a profile about Bob locations and the amount of time spend in each location, like at home for at least 4 hours from 8PM to 8AM. The adversary could repeat this process for other locations like work, leisure, sport, restaurant. Since the adversary knows these particularities of Bob movements, he will be able to label future places that will be extracted from the mobility traces of Bob.

- **Mobility models** can be learned from the geolocated data of individuals, and then used either to identify them among a geolocated dataset (even when they are anonymous) or to

\(^5\) As the exact identity of the drivers have been kept secret it was not possible for the authors to compare directly the houses returned by the algorithm against the ground truth (i.e. the exact address of the drivers) which explained why this particular evaluation method was chosen.
predict their next movements. For instance, Lio, Fox and Kautz [LFK05] have shown that it is possible to train a relational Markov network, so that it can predict with a relatively good accuracy the next location of an individual or his current activities. Another possibility is to use an algorithm for tracking the movement of targets [Zho79] to reconstruct the paths followed by several individuals in a geolocated dataset even if they are anonymous. Some mobility models [ABK*07a, SPD*08] also attach a semantic to the places visited, thus enriching the knowledge extracted.

- **Heuristics** also give good results in practice [Kru07, GKNdPC10a] for identifying POIs at a relatively low cost. An heuristic can be as simple as choosing the last stop before midnight or the average (or median) of several stop locations for identifying the home or the most stable location during the day for finding the place of work.

Often, the inference process does not consist only in the application of one inference attack, but rather is an incremental process in which the adversary gradually gathers more and more knowledge through the combination of several inference attacks. Thus, we can consider two auxiliaries sources of knowledge:

- **Data coming from social applications** is a possible source of information that the adversary might draw on to attack the privacy of individuals. The website “Please Rob Me” is a striking example of how it is possible from publicly available information in the form of Twitter’s posts (i.e., tweets) to build a classifier that can predict whether or not somebody is currently at home. Another example of social application is Google Latitude that offers the possibility of following in real-time on a map the movements of siblings and friends who have previously agreed to this service by confirming this on a SMS received on their phone. However, some social applications such as Locaccino [Bak] tries to integrate explicitly the privacy issues in their design, by giving the possibility to a user to choose how he wants to disclose and share its location with its friends, and helping him understand what are the potential privacy risks that he might incur.

- **Data coming from public sources** is also a potential source of knowledge that can be exploited by the adversary. For instance, by using Google Maps and Yahoo!Maps the adversary can easily reconstruct the path followed by an individual between two consecutive mobility traces. Moreover, reverse geocoding tools exist that can transform a spatial coordinate into a physical address, which in turn can be cross-referenced with the corresponding entries in the Yellow Pages.

Now that we have presented some techniques to accomplish inference attacks. We establish a classification of inference attacks based on the adversary objective, which is introduced in the next section.

---

6 An infamous use of Google Latitude is known as the *shower attack* where a suspicious husband waits for his wife to take her shower, before sending the Google Latitude SMS to her cellphone, accepting this service on her behalf on the cellphone and then erasing it thus leaving not clue for her that she is now tracked.
1.5 Classification of inference attacks

In this section, we introduce a detailed description of the objectives of an adversary when performing an inference attack on location data. Wernke and co-authors [WSDR12] classify the inference attacks based on the auxiliary knowledge that the adversary may have. Namely, they categorize attacks according to: single position that infers the position or identity of a user based on a single mobility trace; context linking, which combines external data sources with the mobility trace to gain knowledge about users; multiple positions that uses the correlations available on a trail of mobility traces to discover new information about the user and combination of multiple positions and context linking, which is combination of aforementioned attacks. The authors also present the compromised trusted third party attack, which refers more to a way of obtaining location data to an inference attack. In our classification, we assume that the adversary has access to one or several geolocated datasets, which form the input of the attacks, as prior knowledge (see Chapter 3). The tables in the Appendix B show the inferences attacks existing in the literature as well as the type of dataset that is used as input to these attacks.

Figure 1.3 summarizes our classification of inference attacks. The main fundamental attack is the identification of POIs, which allows to next location prediction, semantic extraction, de-anonymization and/or reconstruction of the social graph. We mainly consider inference attacks that are performed in an offline manner. An adversary attacking a geolocated data may have various objectives ranging from identifying the home of the target to reconstructing his social network, through obtaining knowledge of his favorite jogging tracks. More precisely, the possible objectives of inference attacks are detailed in the following subsections.

1.5.1 Identification of Points of Interests

The identification of important places, called Points Of Interests (POIs), characterize the mobility of an individual [KSBW04]. A POI may be for instance the home or place of work of an individual or locations such as a sport center, theater or the headquarters of a political party. Revealing the POIs of a particular individual is likely to cause a privacy breach as this data may be used to infer sensible information such as hobbies, religious beliefs, political preferences or even potential diseases [LFK07]. For instance, if an individual has been visiting a medical cen-
1.5. Classification of inference attacks

1.5.2 Next place prediction

This attack predicts the movement patterns of an individual such as his past, present and future locations. According to some recent works [KSBW04, MD01], our movements are easily predictable by nature. For instance, the authors of these papers have estimated to 93% the chance of correctly guessing the future location of a given individual after some training period on his mobility patterns [SQBB10].

1.5.3 De-Anonymization attacks

The aim of the de-anonymization attack is to link the records of the same individual, which can be contained in different geolocated datasets or in the same dataset, either anonymized or under different pseudonyms. This type of attack can be considered to be the geoprivacy equivalent of the statistical disclosure risk in which privacy is measured according to the risk of linking the record of the same individual in two different databases (e.g., establishing that a particular individual in the voting register is also a specific patient of an hospital [Swe02]). In a geolocated context, the purpose of a linking attack might be to associate the movements of Alice’s car (contained for instance in dataset A) with the tracking of her cell phone locations (recorded in another dataset B). As the POIs of an individual and his movement patterns constitute a form of fingerprinting, simply anonymizing or pseudonymizing the geolocated data is clearly not a sufficient form of protection against linking attacks. For example, Colle and Kartridge [GP09] have shown that even the pair home-work becomes almost unique per individual, and thus acts as a quasi-identifier, if the granularity is not coarse enough (e.g., if the street is revealed instead of the neighborhood).

1.5.4 Extracting the semantic of POIs

The objective of extracting the semantic of POIs is to learn the semantics of the mobility behavior of an individual from the knowledge of his POIs and movement patterns. For example, some mobility models such as semantic trajectories [SPD’08, ABK’07a] do not only represent the evolution of the movements of an individual over time but also attach a semantic label to the places visited. From this semantic information, the adversary can derive a clearer understanding about the interests of an individual as well as his mobility behavior than simply from his movement patterns. For instance, the adversary might be able to infer that on a typical weekday the individual considered generally leaves his home (POI 1) to bring his kid to school (POI 2) before going to work (POI 3), which is a deeper knowledge than simply knowing the movement pattern “POI 1 ⇒ POI 2 ⇒ POI 3”. Semantic can also sometimes be extracted by using external knowledge, through heuristics or by analyzing the structure of the model.
1.5.5 Discovering the social graph

The basic idea of this attack is to exploit the fact that the social networks of individuals and their mobility patterns are correlated together. Thus, an adversary could build the social graph of someone using his mobility traces. On one hand, social network can be inferred from location data as shown in the work of Wang and co-authors [WGX+11]. It is also possible to learn a mobility model from social interactions. For example, Boumerdassi and co-authors [HBR10, CB12] have studied the relations between real mobility traces and the mobility traces issued from social interactions. Then, they have proposed a mobility model that takes into account the influence of the social network.

1.5.6 Predicting demographic attributes

The predicting demographic attributes and discovering the social graph, both attacks, are related one to each other. Relaying on demographic attributes, social links can be discovered [LK08, JM09] and through social links, demographic attributes could be used to deduce people’s hidden attributes [HPV07, GA05]. The demographic attributes could be inferred using predictability and hidden Markov mobility (HMM) entropy as show in the work of Kelly, Smyth and Caulfield [KSC13]. Their objective is to characterize the age, gender, travel tendencies and social habits of people. The work of Montjoye and co-authors [dMQRP13] propose a method to predict the personality characteristics, such as neurotism, extroversion, conscientiousness, agreeableness or openness, based on telephones activities log of cell phone users.

1.6 Geosanitization mechanisms

A geosanitization algorithm $S$ takes as input a geolocated dataset $D$, introduces some uncertainty and removes some information from this dataset to increase the privacy of individuals whose movements are contained in the dataset. $S$ produces as output $D'$, a sanitized version of the original dataset $D$. The main idea behind sanitization is that, for a potential adversary, breaching the privacy of a particular user is harder when working on $D'$ than with $D$. A sanitization procedure usually comes with some privacy guarantees. For instance, it can guarantee that at each time step, there is a minimum number of individuals in each spatial area. Possible sanitization techniques include:

- **Pseudonymization** replaces the common identifier of several mobility traces by either a randomly generated pseudonym (thus providing anonymity but not unlinkability) or by the unknown value (thus theoretically granting full anonymity and unlinkability). Pseudonymization is generally performed as the first step of a sanitization process but as such is often not sufficient for protecting the privacy of individuals.

---

7 *Anonymity* can be defined as being able to perform a particular action without having to reveal his identity whereas *unlinkability* is a stronger notion that involves not being able to link two different actions that have been performed by the same user. Typically, performing different actions under a pseudonym (instead of using his real name) provides anonymity but not unlinkability. See [PH08] for more details.
1.6. Geosanitization mechanisms

- **Perturbation** methods [ARZ99] modify the spatial coordinate of a mobility trace by adding some random perturbation. For example, this noise can be generated uniformly or using Gaussian noise within a sphere of radius $r$ centered on the original coordinate. If the geography of the surrounding area is not taken into account, it may happen that the perturbed coordinate corresponds to a location which has no physical sense (e.g., in the middle of a river or on a cliff).

- **Aggregation** merges several mobility traces into a single spatial coordinate. For instance, this spatial coordinate can be a surrounding spatial area such as a neighbourhood or the average of the mobility traces. During data preprocessing, a clustering algorithm (such as $k$-means) can be used to group traces that are close together into the same cluster while putting traces that are significantly distant into distinct clusters. This can be used to detect which traces should be merged together during an aggregation step. Another possibility is to detect traces occupying the same spatial area (for instance the same neighbourhood) at a certain moment in time and to replace each one of these individual traces by the coordinate of this spatial area.

- **Sampling** can be seen as a form of temporal aggregation. A sampling mechanism summarizes several mobility traces into fewer traces, generally by representing an ensemble of traces, which have occurred within some time window, into one median or average trace. By decreasing the total number of traces, sampling has the additional benefit that it compresses the data and, therefore, reduces the computational resources needed to further sanitize the data.

- **Spatial cloaking** [GG03] is an extension of the concept of $k$-anonymity [Swe02] to the spatio-temporal domain and a form of aggregation. The main idea is to ensure that at each time step, each individual is located within a spatial area that is shared by at least $k-1$ other individuals. This spatial area is disclosed instead of the exact location of these individuals, thus guaranteeing that even if an adversary can target the group where an individual is located, his behavior will be indistinguishable from at least $k-1$ other individuals ($k$ is a privacy parameter of the algorithm). A possible approach to achieve the property of spatial cloaking is to split recursively the space into areas of different sizes, until each area contains at least $k$ individuals.

- **Mix-zones** [BS03] are inspired from the concept of mix-nets due to Chaum. Mix-zones are spatial areas where (1) no measurements about the locations of individuals are performed and (2) such that each individual entering a mix-zone will have a different pseudonym when he exits the mix-zone. The main purpose of a mix-zone is to make it more difficult to link the different actions of an individual. Areas or buildings with a high traffic are usually good candidates for mix-zones.

- **Swapping** consists in exchanging the mobility traces of two different individuals for a certain period of time. For example, by swapping Alice’s and Bob’s traces during one day, their behaviors become more atypical and less predictable.

- **Removing** the mobility traces that are deemed too sensible can also be considered as a
sanitization procedure. In the same spirit, it is also possible to add fake records (called dummies) [YPL07] inside the sanitized dataset to blend the true movements of individuals inside artificial data.

As sanitization leads to a loss of information, it is important to have a utility metric in order to compare the utility of the original dataset \( D \) and the sanitized one \( D' \). A compromise needs to be found to ensure privacy, while keeping a reasonable data utility. In the next subsection, we briefly discuss the issue of how to measure utility.

1.7 Data utility

To ensure privacy, a sanitization process has to be performed, which adds uncertainty to the data and removes sensitive data. This loss of information, incurred by the sanitization process, comes with a dilemma: it certainly brings some privacy guarantees but at the cost of a decrease of the utility due to the data quality degradation. Therefore, there is often a trade-off that has to be set between the utility of the global task and the individuals’ privacy protection. The utility measure can either be generic or application-dependent. The former is linked to some global statistical properties like the distance between probability distributions of the original and sanitized datasets [GKKM07, KKO’06, WROK09], while the latter evaluates how well a particular application can be performed by using the sanitized dataset \( D' \) instead of original data \( D \) [LW10, WYC04].

One of the main challenges for geoprivacy is to balance the benefits for an individual’s participation in a geolocated application with the corresponding privacy risks. For example, Alice’s car is equipped with a GPS and she accepts to participate in real-time computation of the traffic map, which corresponds to a task that is mutually beneficial to all the drivers but at the same time Alice wants to have some privacy guarantees that her individual locations will be protected and not disclosed without sanitization.

1.8 Summary

In this chapter, we have introduced some concepts like location, location-based service and discuss the economical value of location data in order to establish a terminology for this dissertation. Then, we have presented privacy and privacy laws to extend these concepts to a definition of location privacy. We have described the main data we will use in the present dissertation like mobility traces and location datasets. Finally, we have described inference attacks, counter measures and the utility of the geolocated datasets.

In order to evaluate the privacy risks in the geolocated context. We describe in the subsequent chapters the inference attacks that we have developed. In particular in the next chapter, we will review how to extract the points of interests of a user out of his mobility traces.
“The ways in which the information we give off about our selves, in photos and e-mails and MySpace pages and all the rest of it, has dramatically increased our social visibility and made it easier for us to find each other but also to be scrutinized in public.”


The extraction of the point of interest (POI), identification or extraction, from a trail of mobility traces, is the first step in inference attacks. Indeed, this phase impacts directly the global accuracy of an inference attack that relies on POI extraction. For instance, if an adversary wants to discover Alice’s home and place of work the result of the extraction must be as accurate as possible, otherwise they can confuse or just not find important places. In addition, for a more sophisticated attack such as next place prediction, a mistake when extracting POIs can decrease significantly the global precision of the inference. Most of the previous works use heuristics and clustering algorithms to extract POIs from location data. For instance, heuristics rely on the *dwell time*, which is the lost of signal when user gets into a building. Another used heuristic is the *residence time*, which represents the time that a user spends at a particular place. On the other hand, clustering algorithms group nearby mobility traces into clusters. Most of the previous works estimate the parameters of the clustering algorithms for the point of interest extraction by using empirical approaches or highly computationally expensive methods. For instance, we use for illustration purpose two classical clustering approaches, *K*-means [M+67] and DBSCAN [EpKSX96]. In the *k*-means clustering algorithm, the main issue is how to determine *k*, the number of clusters. Therefore, several approaches have been proposed to address this issue [HE03, PDN05]. The DBSCAN algorithm relies on OPTICS [ABKS99] algorithm, which searches the space of parameters of DBSCAN in order to find the optimal number of clusters. The more parameters the clustering algorithm has, the bigger the combinatorial space of parameters is. Nevertheless, these methods to calibrate cluster algorithm inputs do not guarantee a good accuracy for extracting meaningful places. In particular in the context of POI extraction, it is important to find a suitable set of parameters, for a specific cluster algorithm, in order to obtain a good accuracy as result of the attack. In this chapter, we will present
Chapter 2. Extraction of points of interest

a methodology to find “the optimal” configuration of input parameters based on quality indices and the relationship between them. This chapter is organized as follows. First, we present some POI extraction techniques in the literature as well as the four clustering algorithms that we have used for POI extraction in Section 2.1 and 2.2, respectively. Afterward, an evaluation and comparison of the clustering algorithms is performed in Section 2.3. Then, we introduce some metrics for measuring the quality of a cluster (Section 2.4) and a method to optimize the choice of the parameters (Sections 2.5). Finally, Section 2.6 summarizes the results of this chapter.

2.1 Related work on POI extraction techniques

The first step to infer implicit information from a given a set of mobility traces is to deduce the important visited places. The identification of POIs is an inference attack that relies on a heuristic or a clustering algorithm to identify the POIs characterizing the interest of an individual. POI may be for instance, the home or place of work of an individual or locations such as a sport center, theater or the headquarters of a political party. Revealing the POIs of a particular individual is likely to cause a privacy breach as this data may be used to infer sensible information.

2.1.1 Extraction via heuristics

One of the most common heuristic is the dwell time [War96], which is the amount time of loss of GPS signal of a Global Positioning System (GPS). Indeed, when a user enters to building or in a city valley, this cause interferences with the GPS positioning. Nevertheless, the impact of signal loss can be decreased with the use of Assisted GPS (AGPS). Dwell time makes reference to an amount of time spent within some defined region. Using this heuristic, Alvares et al. [ABK’07a, ABK’07b] identified important places from a dataset of people arriving and going to the conference center, visiting touristic places and moving in different directions. This work does not present an evaluation of the accuracy of the POI extraction algorithm. The work of Krumm [Kru07] relies on the probability of a user being at home given the time of a day. He assumes that the highest probability to find someone at home is between 6PM to 8AM. Based on this heuristic, the median error was 61 meters away from the real home in a dataset of 172 unknown drivers. POI extraction using temporal series of time arrivals and residence time (duration) related to movements captured by GPS and WiFi logs was performed by Scellato et al. [SMM’11]. The POI discovery of GPS mobility traces consists of computing the Gaussian distribution of residence time analysis. The time a user spends in a certain place is proportional to the probability. In the case of the WiFi mobility traces, each access point is considered to be a POI. Regarding the results, this method finds on average about 12 POIs for each user in the CenceMe GPS dataset [MLF’08], 18 in the Dartmouth WiFi and 24 the Cabspotting datasets [KHAY09] and 4 in the Ile Sans Fils dataset [LGoP07]. Additionally, they estimate the residence time, which is the percentage of time spend in a given POI, in each dataset: CenceMe GPS (14.74%), Dartmouth WiFi datasets (11.24%), Cabspotting (7.27%) and Ile Sans Fils datasets (0.18%). Khetrapaul et al. [KCG’11] derive common POIs from 62 random users of Geolife.
2.1. Related work on POI extraction techniques

[ZLC+08]. Their approach consists on finding stops (i.e., places in which a user remains more than a time threshold within a certain distance) to merge them as POIs. Specifically, for each user the algorithm looks as input a distance \(d\), a time \(t\) and a frequency \(f\) threshold. Then, it looks at consecutive mobility traces that are within a distance \(d\) more than \(t\) minutes. Next, a multiset union of user’s stops is performed to find the densest common areas. Finally, the number of stays in the selected areas is counted, if the count is greater than the \(f\) threshold the area is considered a POI. They were able that extract 3189 POIs for the dataset using 200 meters and 20 minutes as input parameters. Finally, a hybrid approach is presented by Browers and Whoehrle [BW11]: to extract POIs from GPS and WiFi mobility traces detected. This approach relies on the dwell time to generate the set of mobility traces that will be the input of a clustering algorithm.

2.1.2 Extraction through clustering algorithms

Another approach to extract important places (i.e., POIs) is to use clustering algorithms. For instance, Isaacman et al. [IBC+11] proposes a method to extract POIs using GSM traces. The authors sort GSM antennas based on the number of days they were contacted, before applying Hartigan’s leader clustering algorithm [Har75], which forms clusters by taking into account a distance \(d\) threshold. More precisely, to form a new cluster the algorithm takes as centroid the first antenna in the sorted list. Then, it looks over the data to measure the distance between the centroid and the following points until the distance, between the centroid and a given point in the list, exceeds the threshold \(d\). This process is repeated until all clusters centroids are found or no further mobility trace can be assigned. These centroids are computed as a weighted average of the number of days they were contacted. Finally, to determine the likelihood of a cluster being a point of interest, they use logistic regression [DWHL04], which computes the probability of a place to be important as function of the number of days, frequency and amount of events. In order to verify the relevance of their algorithm, the authors use a Call Data Record (CDR) dataset of 37 users recorded over 60 days in different states in USA. They found that 60% of the POIs were only one mile away from the real POIs (obtained from census data) and they succeed in finding important locations for 97.5% of the users of the dataset. Ashbrook and Starner [AS03] extract the points of interests frequently visited by the individual before building a mobility model. POIs are discovered using a variant of the \(k\)-means clustering algorithm on the individual’s mobility traces. The authors were able to extract 30 POIs from one user with four months data.

Zhou et al., proposes a technique to discover POIs based on a spatiotemporal version of DBSCAN [EpKSX96] named DJ cluster [ZFL+04]. This algorithm is adaptive in the sense that the number of POIs extracted depends on the mobility behavior of the individual studied, while in \(k\)-means the number of clusters to be discovered has to be fixed in advance. More precisely, DJ cluster takes only as input parameters an upper bound on the radius of clusters and the minimal number of mobility traces that should be contained in a cluster. Authors measure their results in terms of recall (83%), precision (71%) and surprise factor (14%), which is the ratio between found POIs that were not declared by the user at the beginning of the experiments and the total number of found POIs. The work of Kang et al. [KSBW04] proposes a time distance based cluster (TD cluster) that takes as input a distance \(d\), a time \(t\) and a frequency \(f\) thresh-
old to discover the POIs. Specifically, the algorithm takes the first mobility trace and measures the distance between it and the successive mobility traces until the distance exceeds \(d\). Once the distance threshold \(d\) is exceeded, a cluster is formed and the algorithm restarts the process with the rest of the mobility traces. Once all clusters are formed, the algorithm verifies that the time between the youngest and the oldest mobility traces surpasses the time threshold \(t\) to decide if it should keep them or not. Finally, it validates that the remaining clusters are visited frequently (e.g., more than the frequency threshold \(f\)). The authors use two datasets: one containing GPS traces of a user during 8 days, another one of 19 days and a log of users’ POIs (i.e., ground truth). They achieve a precision of 87% and a recall of 100% for the first dataset, while for the second trial of mobility traces they have a precision and recall of 61% and 90%, respectively. Kikhia et al. [KBH+12] propose an automatic method to infer POIs from GPS logs in order to build a loglife system. Specifically, they apply the DBSCAN [EpKSX96] clustering algorithm, whose parameters are previously tuned using OPTICS algorithm, to extract POIs. Afterward, the found POIs are used as input of the convex Hull algorithm [PS85a] to determine geographical boundaries of POIs. The authors have tested their method on a dataset of mobility traces of 3 users over six months. Unfortunately they do not report on the quality of the POIs discovered in this study. Brouwers and Whoehrle [BW11] extract POIs from GPS, WiFi and GPS + WiFi mobility trace using four different clustering algorithms such as K-means algorithm variance [AS03], the clustering algorithm Ashbrook [AS03], DT cluster [HT04] and DJ cluster [ZFL+04]. The results are summarized in Table 2.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ashbrook GPS</td>
<td>58.59%</td>
<td>80.65%</td>
</tr>
<tr>
<td>Ashbrook WiFi</td>
<td>73.33%</td>
<td>94.62%</td>
</tr>
<tr>
<td>Ashbrook GPS+WiFi</td>
<td>60.00%</td>
<td>96.77%</td>
</tr>
<tr>
<td>Hariharan GPS</td>
<td>63.12%</td>
<td>95.70%</td>
</tr>
<tr>
<td>Hariharan WiFi</td>
<td>51.72%</td>
<td>96.77%</td>
</tr>
<tr>
<td>Hariharan GPS+WiFi</td>
<td>55.09%</td>
<td>98.92%</td>
</tr>
<tr>
<td>Timebased GPS</td>
<td>69.70%</td>
<td>74.19%</td>
</tr>
<tr>
<td>Timebased WiFi</td>
<td>70.49%</td>
<td>92.47%</td>
</tr>
<tr>
<td>Timebased GPS+WiFi</td>
<td>70.08%</td>
<td>95.70%</td>
</tr>
<tr>
<td>DJ cluster GPS</td>
<td>98.51%</td>
<td>70.97%</td>
</tr>
<tr>
<td>DJ cluster WiFi</td>
<td>96.63%</td>
<td>92.47%</td>
</tr>
<tr>
<td>DJ cluster GPS+WiFi</td>
<td>98.88%</td>
<td>94.62%</td>
</tr>
</tbody>
</table>

Table 2.1: Resume table of comparative study in [BW11].

In the next section, we will detail 3 spatiotemporal clustering algorithms: Density Joinable cluster, Density Time cluster and Time Density cluster as well as the Begin-end heuristic.
2.2 Clustering algorithms for extraction of points of interest

In this section, we detail 3 different clustering algorithms, which are adapted to process spatiotemporal data as well as heuristic method for identifying POIs.

2.2.1 Density Joinable cluster

DJ-Cluster [ZFL’04] is a clustering algorithm taking as input a minimal number of points \( minpts \), a radius \( r \) and a trail of mobility traces \( M \) (cf. Algorithm 4). This algorithm works in two phases. First, the preprocessing phase discards all the moving points (i.e. whose speed is above \( \epsilon \), for \( \epsilon \) a small value) and then, squashes series of repeated static points into a single occurrence for each series. Next, the second phase clusters the remaining points based on neighborhood density. More precisely, the number of points in the neighborhood must be equal or greater than \( minpts \) and these points must be within radius \( r \) from the medoid of a set of points. Then, the algorithm merges the new cluster with the clusters already computed, which share at least one common point. Finally during the merging, the algorithm erases old computed clusters and only keeps the new cluster, which contains all the other merged clusters.

2.2.2 Density Time cluster

DT cluster [HT04] is an iterative clustering algorithm taking as input a distance threshold \( d \), a time threshold \( t \) and a trail of mobility traces \( M \). First, the algorithm starts by building a cluster \( C \) composed of all the consecutive points within distance \( d \) from each other (cf. Algorithm 5). Afterwards, the algorithm checks if the accumulated time of mobility traces between the youngest and the oldest ones is greater than the threshold \( t \). If it is the case, the cluster is created and added to the list of POIs. Finally as a post-processing step, DT cluster merges the clusters whose mediids are less than \( d/3 \) far from each other.

2.2.3 Time Density cluster

Introduced [GKNndPC11a] under the name "GEPETO clustering" algorithm is a clustering algorithm inspired from DT Cluster, which takes as input parameters a radius \( r \), a time window \( t \), a tolerance rate \( \tau \), a distance threshold \( d \) and a trail of mobility traces \( M \). The algorithm starts by building iteratively clusters from a trail \( M \) of mobility traces that are located within the time window \( t \). Afterwards, for each cluster, if a fraction of the points (above the tolerance rate \( \tau \)) are within radius \( r \) from the medoid, the cluster is integrated to the list of clusters outputted, whereas otherwise it is simply discarded. Finally, as for DT Cluster, the algorithm merges the clusters whose medoids are less than \( d \) far from each other. See Algorithm 6 for a brief description of this method.

2.2.4 Begin-end heuristic

The objective of the begin and end location finder inference attack [GKNndPC10a] is to take as meaningful points the first and last of a journey. More precisely, this heuristic considers that the
Algorithm 4 Density joinable cluster

Require: Minimal number of points $minPts$, radius $r$ and trail of mobility traces $M$

Ensure: List $l$ of found POIs

% Preprocessing step
Remove all points whose speed $> \epsilon$
Remove all redundant sequential points

% End preprocessing step
$N = M.length$
List $l$ of found POIs = empty
List noise of points considered as noise = empty

for ($i = 0$ until $N - 1$) do
% Find neighbors
$numPts = 0$

for ($j = 0$ until $N - 1$) do
if ($i \neq j$) then
create a cluster $cl$ with $M[i]$
if (distance($M[i], M[j]$) $\leq r$) then
add $M[j]$ to cluster $cl$
$numPts = numPts + 1$
end if
end if
end for
if ($numPts \geq minPts$) then
% Join clusters sharing at least a mt
List of booleans of merged clusters of $l.length$ size
Initialize all elements of merged = false

for ($k = 0$ until $l.length$) do
joinable=false
if ($cl \cap l[i][0]$) then
join($cl$ to $l[i]$)
joinable=true
merged[$k$]=joinable
end if
end for
for ($m = 0$ until merged.length) do
if (merged[m]=true) then
l.erase($m$)
end if
end for
add $cl$ to $l$
else
add $M[i]$ to noise
end if
end for
return List $l$ of found POIs
2.2. Clustering algorithms for extraction of points of interest

Algorithm 5 Density time cluster algorithm

Require: Distance threshold $d$, a time threshold $t$ and a trail of mobility traces $M$

Ensure: List $l$ of found POIs

$$N = \text{M.length}$$
$$\text{sumTime} = 0$$

List $l$ of found POIs = empty

for $i = 0$ until $N - 1$

% It builds a cluster with the very first point

if $i$ equals to 0 then
    create cluster $cl$ with $M[i]$
end if

if distance($cl$.$\text{medoid}$, $M[i]$.$\text{position}$ $< d$) then
    $\text{sumTime} = \text{sumTime} + \text{timedifference}(M[i], M[i+1])$
else
    if $\text{sumTime} \geq t$ then
        add $cl$ to $l$
    end if
    create a new cluster $cl$ with $M[i+1]$
    $\text{sumTime} = 0$
end if
end for

Merge clusters of $l$ in which distance between medoid $< d/3$

return List $l$ of found POIs
Chapter 2. Extraction of points of interest

Algorithm 6 GEPETO or TD cluster clustering algorithm

Require: Trail of mobility traces \( M \), time window \( t \), radius \( r \), tolerance rate \( \tau \), distance threshold \( d \)

Ensure: \( N=M.length \)

Initialize list \( l \) of found POIs = empty

Create a empty cluster \( cl \)

for \( i = 0 \) until \( N - 1 \) do

\( cumulTime = cumulTime + \text{TimeDifference}(M[i+1].time - M[i].time) \)

if \( cumulTime \leq t \) then

Add the mobility trace \( M[i] \) to cluster \( cl \)

else

Compute the medoid of \( cl \)

\( nbPtsOutsideRadius = 0 \)

\( totalPoints = cl.nbPts \)

for \( j = 0 \) to \( cl.nbPts \) do

if distance(\( c[j] \), cl.medoid) > \( r \) then

\( nbPtsOutsideRadius = nbPtsOutsideRadius + 1 \)

end if

end for

if \( nbPtsOutsideRadius/\text{totalPoints} < \tau \) then

Add the cluster \( cl \) to \( l \)

end if

Reset \( cumulTime \) to 0 and create a new empty cluster \( cl \)

end if

end for

Merge clusters of \( l \) whose distance between medoids is less than \( d \)

return \( l \), the list of discovered POIs
beginning and ending locations of a user, for each working day, might convey some meaningful
information. This attack, illustrated in Algorithm 7, is a simple heuristic assuming that the first
and last recorded mobility traces in a working day correspond to the departure and arrival
points from a POI. The intuition is that when there is no mobility trace measured during a
period longer than a given time threshold $\tau$, this means that the individual had a “mobility
break” and the place in which they took this break is likely to be a POI. If $\tau$ is chosen sufficiently
large (e.g., 6 hours), this POI may for instance, be the home of individuals in which they went
to sleep after their work. Now that we have introduced the different algorithms we use to
Algorithm 7 Begin end attack

Require: Trail of mobility traces $M$, threshold $t$
Ensure: List $l$ of POIs found

\[ N = \text{M.length} \]
\[ \text{List } l \text{ of POIs found } = \text{empty} \]

for \( i = 0 \) until \( N - 1 \) do
  if ((TimeDifference($m[i].time$, $m[i+1].time$) $\geq t$) then
    add $M[i]$ and $M[i+1]$ to $l$
  end if
  \( i = i + 1 \)
end for

return $l$, the list of discovered POIs

extract the points of interest, we try to understand how these algorithms behave when the
input is a sanitized trail of mobility traces. Thus, in the next section we present the evaluation
of performance and resilience when a sanitized input has been provided.

2.3 Evaluation of performance and resilience to sanitization

The four algorithms (the Begin-end heuristic, DJ cluster, DT cluster and TD cluster (GEPETO
clustering algorithm)) described in Section 2.2 were implemented within a tool named GEo
Privacy Enhanced TOolkit (GEPETO) (cf., Section 3.3) and applied to the San Francisco cabs
dataset [PSDG09] for identifying POIs. We used both the original and sanitized versions of
the taxi dataset to evaluate the resilience of the inference attacks against sanitization. More
precisely, we applied both sampling and perturbation techniques (cf. Section 1.6) with various
ranges of parameters. In each situation, we evaluate the recall and precision of the produced
POIs. This recall-precision evaluation requires to be able to judge whether or not a POI is “cor-
correct’. To automate this process, we defined 6 areas in San Francisco, because we do not know
the ground truth, that make good candidates for real POIs and, which are at the same time
generic enough: the taxi company parking lot, the main train station, the airport, the city cen-
ter and three entertainment areas (the Castro district, Fisherman’s Wharf and the Golden Gate
recreational park). The precision is defined as the ratio between the number of correct POIs and
the total number of POIs returned by the algorithm. In our experiments, a POI is considered
“correct” if it falls inside one of the 6 ground truth areas. The recall is the ratio between the
number of area detected (i.e., hit by at least one POI) and the total number of areas. According to these definitions, an algorithm randomly generating many POIs would have a high recall and a low precision, as it would probably identify all the areas but many POIs would fall outside many of them. An “ideal” algorithm, displaying a high recall and high precision, would generate 6 POIs, one for each area. Figure 2.1 shows the measures like the recall-precision trade-off of the four algorithms against a sampling technique (cf. Section 1.6). In Figure 2.2, the evaluation is performed for the 4 algorithms against random perturbation (cf. Section 1.6). These experiments have shown that the Begin-end heuristic has an excellent recall, due to the high number of POIs generated and an average precision but is sensitive to sampling. Indeed,
2.3. Evaluation of performance and resilience to sanitization

Figure 2.2: Precision-recall of the four algorithms with different perturbation rates.

when the sampling rate reaches the size of the time window of the begin-end heuristic, it considers all the traces as POIs. On the other hand, the begin-end heuristic is not too impacted by perturbation and even under large distortion; it stays one of the most precise algorithms. DJ Cluster displays a terrible behavior in the presence of sampling, and even a worst one with respect to distortion. Indeed, the first phase of the algorithm removes the moving traces to focus on those in which the individual is not moving. With sampling, the probability is high that the sanitization process removes static traces. Moreover, when applying a perturbation operator to the data, each single trace implies some movement. Henceforth, all the traces are removed during the first phase of the algorithm and DJ Cluster does not output any POI. DT
Cluster is highly resilient against sampling, with a high recall and the best precision, but displays a low recall against distortion. However, the precision of the remaining POIs is still good under moderate perturbation. Finally, GEPETO cluster (TD cluster) seems to be a good compromise. For instance, its behavior is comparable or just below DT cluster in the presence of sampling with average to good recall and precision. Moreover, under distortion, it seconds the Begin-end heuristic with an average recall and a high precision. To summarize, the efficiency of the inference attack depends strongly on the sanitization process that has been performed on the target data. For instance, in the presence of sampling, the DT cluster algorithm offers a high recall and a good precision, but its performance degrades significantly with respect to distortion. Therefore, if the adversary knows \textit{a priori} that the sanitization process is only based on sampling, then he can directly choose the DT cluster algorithm for performing an efficient inference attack. On the other hand, GEPETO cluster (TD cluster) seems to be a reasonable alternative for an adversary having no knowledge of the sanitization technique applied as its performance remains good under both sampling and distortion. In the following subsection we discuss about how to quantify the quality of the results of the clustering algorithms.

\section{Cluster quality index}

One aspect that has to be taken into account while performing extraction of POIs inference attacks, is the quality of the obtained clusters, which affects the precision and recall of the attack. In the following subsection we describe some metrics to quantify how accurate or “how good” a clustering that has been obtained is. Intuitively, a good clustering is one that identifies a group of clusters that are well separated one from each other, compact and representative. Table 2.2 summarizes the notation used in this section.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C )</td>
<td>An ensemble of clusters.</td>
</tr>
<tr>
<td>( c_i )</td>
<td>The ( i )th cluster of ( C ).</td>
</tr>
<tr>
<td>( n_c )</td>
<td>The number of clusters in ( C ).</td>
</tr>
<tr>
<td>( m_i )</td>
<td>The medoid point of the ( i )th cluster.</td>
</tr>
<tr>
<td>( d(x, y) )</td>
<td>The Euclidean distance between ( x ) and ( y ) point.</td>
</tr>
<tr>
<td>(</td>
<td>c</td>
</tr>
<tr>
<td>( m' )</td>
<td>The closest point to the medoid ( m_i ).</td>
</tr>
<tr>
<td>( m'' )</td>
<td>The second closest point to the medoid ( m_i ).</td>
</tr>
<tr>
<td>(</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 2.2: Summary of notations

\subsection{Intra-inter cluster ratio}

The \textit{intra-inter} cluster ratio \cite{Hil12} measures the relation between compact (Equation 2.1) and well separated groups (Equation 2.3). More precisely, we first take the inter-cluster distance,
2.4. Cluster quality index

which is the average distance from each point in a cluster \( c_i \) to its medoid \( m_i \).

\[
DIC(c_i) = \frac{1}{|c_i| - 1} \sum_{x_j \in c_i, x_j \neq m_i} d(x_j, m_i)
\]  

(2.1)

Then, the average intra-cluster distance (DIC) is computed using Equation 2.2.

\[
AVG_{DIC}(C) = \frac{1}{n_C} \sum_{c_i \in C} DIC(c_i)
\]  

(2.2)

Afterward, the mean distance among all medoids (DOC) in the cluster \( C \) is computed, using Equation 2.3.

\[
DOC(C) = \frac{1}{|C| \times (|C| - 1)} \sum_{c_i \in C} \sum_{c_j \in C, i \neq j} d(m_i, m_j)
\]  

(2.3)

Finally, the ratio intra-inter cluster \( r_{ii} \) is given by the Equation 2.4 as the relationship between the average intra cluster distance divided by the inter-cluster distance.

\[
r_{ii}(C) = \frac{AVG_{DIC}(C)}{DOC(C)}
\]  

(2.4)

The intra-inter ratio has an approximate linear complexity in the number of points to be computed and gives low values to well separated and compact cluster.

2.4.2 Additive margin

Ben-David and Ackerman [BDA08] propose a metric to estimate how well centered clusters are, which relies on differences instead of ratios. To evaluate this quality measure, the \( k \)-additive Point Margin (K-AM) is computed. This method measures the difference between the medoid \( m_i \) and its two closest points \( m_i' \) and \( m_i'' \) of a given group \( c_i \) belonging to a cluster \( C \) (Equation 2.5).

\[
K - AM(c_i) = d(m_i, m_i'') - d(m_i, m_i')
\]  

(2.5)

Since the average of the \( k \)-additive point margins for all groups \( c_i \) in a cluster \( C \) is computed, we take the ratio between the average \( k \)-additive Point Margin and the minimal inter-cluster distance (Equation 2.1) as shown in Equation 2.6.

\[
AM(C) = \min_{c_i \in C} \frac{1}{n_c} \sum_{c_j \in C} K - AM(c_i)
\]  

(2.6)

The additive margin method has a linear complexity in the number of clusters. This metric gives a high value for a well centered clusters.

2.4.3 Information loss

The information loss ratio is a metric inspired by the work of Sole et al. [SMMN12]. The basic idea is to measure the percent of information that is lost while representing original data only by a certain number of groups (e.g., when we represent the POIs by the cluster medoids instead
of the whole set of points). To evaluate the percent of information loss, we compute the sum of
distance of each point represented by $x_i$ to its medoid $m_i$ for all clusters $c_i \in C$ as we shown in
Equation 2.7.

$$SSE(C) = \sum_{c_i \in C} \sum_{x_j \in c_i} d(x_j, m_i)$$  \hspace{1cm} (2.7)

Then, we estimate the accumulated distance of all points of a trail of mobility traces in the
cluster $C$ to a global centroid ($M$) using the following equation Equation 2.8.

$$SST(C) = \sum_{x_i \in C} d(x_i, M)$$  \hspace{1cm} (2.8)

Finally, the ratio between aforementioned distances is computed using Equation 2.9, which
results in the information loss ratio.

$$IL(C) = \frac{SSE(C)}{SST(C)}$$  \hspace{1cm} (2.9)

The computation of this ratio has a linear complexity. The lowest is the value of this ratio, the
more representative the clusters are.

### 2.4.4 Dunn index

This quality index [Dun73, HBV01] attempts to recognize compact and well-separated clusters.
The computation of this index relies on a dissimilarity function (e.g. Euclidean distance $d$) be-
tween medoids and the diameter of a cluster (cf., Equation 2.10) as a measure of dispersion.

$$diam(c_i) = \max_{x,y \in c_i, x \neq y} d(x, y)$$  \hspace{1cm} (2.10)

Then, if the clustering $C$ is compact (i.e., the diameter tends to be small) and well separated
(distance between cluster medoids are large), the result of the index, given by the Equation
2.11, is expected to be large.

$$DIL(C) = \min_{c_i \in C} \left[ \min_{c_j \in C, j = i+1} \left[ \frac{d(m_i, m_j)}{\max_{c_k \in C} diam(c_k)} \right] \right]$$  \hspace{1cm} (2.11)

The greater is this index, the better the performance of the clustering algorithm is assumed to
be. The main drawbacks of this index is the computational complexity and the sensitivity to
noise in data.

### 2.4.5 Davis-Bouldin index

The objective of the Davis-Bouldin index (DBI) [DB79, HBV01] is to evaluate how well the
clustering was performed, using properties inherent to the dataset considered. First, we use a
scatter function within the cluster $c_i$ of the cluster $C$ (Equation 2.12).

$$S(c_i) = \sqrt{\frac{1}{n_{c_i}} \sum_{x_j \in c_i} d(x_j, m_i)^2}$$  \hspace{1cm} (2.12)
Then, a dissimilarity measure, given by Equation 2.13, between two different clusters $c_i$ and $c_j$ is computed.

\[ M(c_i, c_j) = \sqrt{d(m_i, m_j)} \quad (2.13) \]

Afterward, a similarity measure between two clusters $c_i$ and $c_j$, called $R$-similarity, is estimated, based on Equation 2.14.

\[ R(c_i, c_j) = \frac{S(c_i) + S(c_j)}{M(c_i, c_j)} \quad (2.14) \]

Then, the most similar cluster $c_j$ to $c_i$ is the one maximizing the result of the function $R_{all}(c_i)$, which is given by Equation 2.15 for $i \neq j$.

\[ R_{all}(c_i) = \max_{c_j \in C, i \neq j} R(c_i, c_j) \quad (2.15) \]

Finally, the DBI is equal to the average of the similarity between clusters in the cluster set $C$ (Equation 2.16).

\[ DBI(C) = \frac{1}{nc} \sum_{c_i \in C} R_{all}(c_i) \quad (2.16) \]

Ideally, the clusters $c_i \in C$ should have the minimum possible similarity to each other. Accordingly, the lower is the DB index, the better is the clustering formed. In the next section we evaluate the cluster algorithm aforementioned in Subsection 2.1.2 as well as the method to extract the meaningful places using the quality indices.

### 2.5 Selecting the optimal parameters for clustering

In order to establish how to select the best set of parameters for a given clustering algorithm, we have computed the precision, recall and F-measure of all users of LifeMap dataset [CC11a]. One of the unique characteristic of this dataset is that the POIs have been annotated by the users. Consequently, given a set of clusters $c_i \in C$ such that $C = \{c_1, c_2, c_3, \ldots, c_n\}$ and a set of points of interest (POIs) defined by the users $P = \{p_1, p_2, p_3, \ldots, p_n\}$ we were able to compute the precision, recall and f-measure as we detail in the next subsection.

#### 2.5.1 Precision, recall and F-measure

We are able to compute the precision by taking into account all the clusters $c_i$ of the clustering set $C$, the ground truth represented by the vector $P$ (which was defined manually by each user) and a radius in Km as inputs of the Algorithm 8. The algorithm computes the ratio of the number of good clusters compared to the total number of found clusters. With respect to recall, Algorithm 9 takes as input a clustering set $C$, a vector of POIs $P$ as well as a radius. To evaluate the recall, the algorithm computes the ratio between the number of good POIs and the total number of POIs. Finally, the F-measure is defined as the weighted average of the precision and recall as we can see in Equation 2.17

\[ F\text{-}measure = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (2.17) \]

We present the results of our experiments in the next subsection.
Chapter 2. Extraction of points of interest

Algorithm 8 Computation of the precision

Require: Set of clusters $C$, Vector of POIs $P$, $radius$

1: for each $c_i$ in $C$ do
2:     for each $p_j$ in $P$ do
3:         if $\text{distance}(c_i, p_j) \leq \text{radius}$ then
4:             count = count + 1;
5:         end if
6:     end for
7: end for
8: $\text{good} \_ \text{cluster} = \text{count}$;
9: $\text{total} \_ \text{cluster} = \text{c.size()}$;
10: $\text{precision} = \text{good} \_ \text{cluster/total} \_ \text{cluster}$;
11: return $\text{precision}$

Algorithm 9 Computation of the Recall

Require: Set of clusters $C$, Vector of POIs $P$, $radius$

1: for each $p_i$ in $P$ do
2:     for each $m_j$ in $C$ do
3:         if $\text{distance}(c_j, p_i) \leq \text{radius}$ then
4:             count = count + 1;
5:             break;
6:         end if
7:     end for
8: end for
9: $\text{good} \_ \text{poi} = \text{count}$;
10: $\text{total} \_ \text{poi} = \text{P.size()}$;
11: $\text{recall} = \text{good} \_ \text{poi/total} \_ \text{poi}$;
12: return $\text{recall}$
2.5. Selecting the optimal parameters for clustering

2.5.2 Experimental results

This subsection is composed of two parts, in the first part we compare the performance of the previously described clustering algorithms, with two baseline clustering algorithms namely \textit{k-means} and DBSCAN. In the second part, a method to select the most suitable parameters for a clustering algorithm is presented.

<table>
<thead>
<tr>
<th>Input parameters</th>
<th>Possible values</th>
<th>DBSCAN</th>
<th>DJ cluster</th>
<th>DT cluster</th>
<th>\textit{k-means}</th>
<th>TD cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance rate (%)</td>
<td>{0.75, 0.8, 0.85, 0.9}</td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minpts (points)</td>
<td>{3, 4, 5, 6, 7, 8, 9, 10, 20, 50}</td>
<td>✓✓✓✓✓</td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eps (Km.)</td>
<td>[0.01, 0.02, 0.05, 0.1, 0.2]</td>
<td>✓✓✓✓✓</td>
<td></td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge distance (Km.)</td>
<td>[0.02, 0.04, 0.1, 0.2, 0.4]</td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time shift (hour)</td>
<td>{1, 2, 3, 4, 5, 6}</td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K (num. clusters)</td>
<td>[5, 6, 7, 8, 9]</td>
<td>✓✓✓✓✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: Summary of input parameters for clustering algorithms.

In order to compare the clustering algorithms (i.e., \textit{k-means}, DBSCAN, DJ cluster, DT cluster and TD cluster), we have evaluated them in terms of precision, recall and F-measure obtained, as well as the average execution time, the number of input parameters and the time complexity. To evaluate these algorithms, we used the LifeMap dataset with POIs annotation and a set of different parameters configurations for each algorithm, which are summarized in Table 2.3. After running these configurations, we obtained the results shown in Table 2.4 for the different input values. For the sake of comparison, we use a radial diagram (Figure 2.3),

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Time(s)</th>
<th>Number of parameters</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSCAN</td>
<td>0.58</td>
<td>0.54</td>
<td>0.48</td>
<td>316</td>
<td>3</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>DJ cluster</td>
<td>0.74</td>
<td>0.52</td>
<td>0.52</td>
<td>429</td>
<td>3</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>DT cluster</td>
<td>0.38</td>
<td>0.47</td>
<td>0.39</td>
<td>279</td>
<td>3</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>\textit{k-means}</td>
<td>0.58</td>
<td>0.51</td>
<td>0.49</td>
<td>299</td>
<td>2</td>
<td>O(n)</td>
</tr>
<tr>
<td>TD cluster</td>
<td>0.43</td>
<td>0.54</td>
<td>0.44</td>
<td>362</td>
<td>4</td>
<td>O(n^2)</td>
</tr>
</tbody>
</table>

Table 2.4: The characteristics of the clustering algorithms.

which represents the normalized values of Table 2.4. It is possible to observe that the precision of DJ cluster outperforms the other clustering algorithms. In terms of recall DBSCAN and TD cluster perform the best but DJ cluster is just behind them. Moreover, DJ cluster has the best F-measure. Regarding the execution time, DT cluster is the fastest one while DJ cluster is the slowest algorithm due to the preprocessing phase. Despite the high computational time of DJ cluster, this algorithm performs well in terms of F-measure.
In the following, we describe our method to choose “optimal” parameters for obtaining a good F-measure. We have used some classical clustering algorithms such as $K$-means and DBSCAN, as well as more spatiotemporal adapted approaches like Density Joinable cluster, Density Time cluster and Time Density cluster with a different set of input parameters configurations for users with POIs annotations in the LifeMap dataset [CC11a]. Once clusters are built, we evaluate the clusters issued from different configurations of the distinct algorithms using the previously described quality indices (i.e. intra-inter index, additive margin, information loss, Dunn index and Davis-Bouldin index). Afterward, we were able to estimate the precision, recall and F-measure using the manual annotation of POIs by the users in the LifeMap dataset. As an example, we display Figure 2.4 the computed indices, for different setups of the DJ cluster algorithm, for a user in the LifeMap dataset. Regarding the relation between the quality indices and the F-measure, we studied the relationship between these factors, in order to identify the indices that are highly correlated with the F-measure, as can be observed in Figure 2.5. We observe that the best performing indices, except for $k$-means, are IL and DBI. The former shows a negative correlation with respect to the F-measure. While the latter, has
a positive dependency to the F-measure. Our main objective is to be able to identify the relationship between quality and F-measure among the previous evaluated clustering algorithms, we discard the inter-intra cluster ratio (RII) and the adaptive margin (AM), which only perform well when using $K$-means and the DJ clustering algorithms. Finally, we observe that the Dunn index has a poor performance. Based on these observations, we were able to propose an algorithm to automatically choose the best configuration of input parameters. The algorithm, depicted in Figure 2.6 in form of a flux diagram, takes as input a trail of mobility traces and a vector with the combinations of values for a given clustering algorithm. Once the clusters, for all the sets of parameters are computed the algorithm computes the Information loss (IL) and
the Davis-Bouldin index (DBI). Once all these values have been computed for each evaluated set of parameters, two cases are possible. In the first case, both IL and DBI agree on the same set of input parameters. In the second situation, both IL and DBI refer each one to a different set of parameters. In this case, the algorithm sorts the values by IL in the ascending order (i.e., from the smallest to the largest information loss value). Then, it chooses the set of parameters with the greatest DBI in the first quartile. For the sake of evaluation, the algorithm described in Figure 2.6 was tested using the LifeMap dataset to check if the chosen parameters are op-

Figure 2.5: Correlation of quality indices with the computed F-measure. Where A) is the correlation measured between the user annotation and the centroid at 20 m of radius B) at 35 m radius C) at 50 m radius, D) at 100 m radius and DBI=Davis-Bouldin index, DI=Dunn index, IL=Information loss, kAM=Additive margin and RII= Ratio intra-inter cluster.
2.5. Selecting the optimal parameters for clustering

We have tested the method with the 5 users of LifeMap that have annotated manually their POIs. The results are shown in Figure 2.7 on average the absolute difference between the selected parameter and the maximal F-measure is about 0.09. On figure 2.7 we can see that the method based on IL and DBI performs better than the method based on only one index. In general, we find the same pattern in all the clustering algorithms evaluated.
Resume is the average of all the results using different clustering algorithms.

Figure 2.7: Relation of F-measure and parameters selection based on schema in Figure 2.6. Where DBI=Davis-Bouldin index, IL=Information loss, IL_DBI= combination of IL and DBI and MAX is the maximal computed F-measure (taken as reference to compare with IL_DBI). Resume is the average of all the results using different clustering algorithms.

2.6 Summary

In the current chapter, we have presented the techniques to extract point of interests from mobility traces. First, we have described techniques relying on heuristics and clustering algorithms. Then, POI extraction algorithms, such as Density joinable cluster, Density time cluster, Time density (GEPETO) cluster and Begin-end heuristic, were detailed as well as a comparison and resilience study. Next, cluster quality indices such as, inter-intra cluster ratio, Additive margin, Information loss, Dunn index and Davis-Bouldin index, were introduced. Finally, a method to select optimal parameters for clustering algorithms based on the precision, recall
and F-measure were presented. In the current chapter, we have described clustering algorithms, whose objective is to explore and detect important frequented places from raw mobility traces. We have also discussed how to select the optimal parameters for the presented clustering algorithms and their role to maximize the number of extracted POIs. Once POIs have been extracted, it is possible to build a model to represent human mobility among their POIs. In the next chapter, we will describe a mobility model called the Mobility markov chain that we will use to perform subsequent inference attacks.
Chapter 3

Mobility Markov chain and GEPETO

“... A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant.”

— Manfred Eigen.

Mobility models have been widely studied for congestion forecasting [FHF07], epidemic dissemination [SD95] and mobile network protocols evaluation [CBD02]. Nevertheless, to the best of our knowledge there is nothing as a “global” human mobility model to evaluate privacy leakage in the context of ubiquitous systems. Existing mobility models used in the domain of privacy are usually used regarding a specific inference attack, such as [NSL+11], [HZL+10] or [IBC+12]. Accordingly, the model presented here, which relies on Markov chains for representing movements among users’ POIs, is able to perform all the inference attacks described in Section 1.5 namely the prediction of the next location, de-anonymization, semantic extraction and social network discovery. In this chapter, we introduce a versatile mobility model based on Markov chains called Mobility Markov Chain (MMC). MMC can be used to perform most of the inference attacks previously mentioned. This chapter is organized as follows. First, Section 3.1 reviews mobility model existing in the literature. Next, Section 3.2 introduces and details the MMC model. Then Section 3.3 and 3.4, describe respectively the tool named GEo-Privacy Enhanced Toolkit (GEPETO) developed to perform inference attacks and sanitization techniques using the MMC as well as the parallelized version of GEPETO using Mapreduce paradigm. Finally, Section 3.5 summarizes the main outcomes.

3.1 Mobility Models

Mobility models have been used for example for traffic simulation, epidemic transmission and mobile network protocols. For instance, Random Walk, Random Waypoint, Random Direction, Boundless Simulation Area, Gauss-Markov or City Section mobility models [CBD02] have been widely studied. Nonetheless, with the advent of ubiquitous systems that generate huge amounts of geolocated traces (from GPS, WiFi or GSM technologies), new and more realistic mobility models have been proposed. For instance, Noulas et al. [NSL+11] propose a mobility model based on the concept of rank, which relies on density of commodities. More precisely,
the rank for each transition between two places A and B is the inverse of the number of places (of the same kind) between those points. For example, if Bob wants to go shopping from his house (A) to a mall (B), the rank of the transition from home to the mall is given by the inverse of the number of stores and malls between A and B. In order to evaluate their model, the authors use a dataset issued from Foursquare collected between May and November 2010. This dataset is composed of 35,289,629 check-ins from 925,030 unique users on 3 different cities (Houston, San Francisco and Singapore). The authors found a positive correlation between the probability density function of the transitions and the rank values. Huang et al. [HZL+10] introduced a global mobility model named MEtropolitan TAxis Extracted (META). The authors use three parameters for their model: the turn probability, which is the probability that a vehicle turn left, right or go straightforward in a given road intersection, the road section speed is the average speed of a road section and the pattern represented as a transition matrix between zones formed by a grid of $1Km^2$. In order to extract the above-mentioned parameter values, the authors analyzed GPS traces issued from 4000 taxis in Shanghai between February and April in 2007. Then, for validation purpose the authors generate synthetic traces, from the mobility models obtained, for comparing them to real mobility traces in terms of the probability of the number of nodes that pass across a road section, finding a difference of 15% as well as the top 10 most frequented road segments. In the same spirit Kim et al. [KKK06] use mobility traces extracted from the WiFi Syslog at Dartmouth College, over a year in June 2003, using a Kalman filter. They rely on pause time, speed, direction, dense regions and the time distribution of devices when they log in or log out from the network. Once they extract all the values for the parameters, they produce a synthetic mobility traces to compare them with the real traces. In order to assess the accuracy of the model, the authors define 5 regions based on user density. Then, they compute the number of users in a given region over time for both, synthetic and real mobility traces. Finally, they compute the average relative error of the 5 regions, finding 17% of relative error between both datasets. Isaacman et al. [IBC+12] propose a model named “Work and Home Extracted REgions” (WHERE) to represent individual and global mobility models. Their model depends on home, work and commute distance probability distribution as well as density distribution per hour and time distribution per user per class. These distributions are extracted from anonymized and aggregated Call Detail Records (CDRs) of Los Angeles and New York as well as publicly available census. Once distributions are extracted, the authors study the accuracy of their model to represent individual and global behavior. First, the authors found two different classes of users (those who calls frequently and those who does not) based on temporal patterns of calls by applying the X-means clustering algorithm [DP00]. Then, they evaluate if their model is able to represent individual behavior of user, with only 2 or 3 POIs, compared to Random Waypoint (RWP) and Weighted Random Waypoint (WRWP) models and real traces by the means of heat maps and Earth Mover’s distance [LB01] showing that WHERE has an error of 0.5 miles with respect to real traces. Next, to quantify errors between global distributions generated from WHERE, RWP, WRWP and real traces authors relies on the daily range (i.e., diameter of a convex hull) [PS85b] to show that WHERE has a more realistic behavior, obtaining an average distance of 14.7, 13.1, 2.4 and 3.2 miles, respectively. At last, Ashbrook and Starner [AS03] propose a mobility model based on Markov chains. More precisely, the authors extract POIs using a $k$-means clustering algorithm. Then, a transition matrix representing the
3.2 Mobility Markov chain

A Mobility Markov Chain (MMC) [GKNndPC11a] models the mobility behavior of an individual as a discrete stochastic process (i.e., an ergodic regular Markov chain) in which the probability of moving to a state (i.e., point of interests) depends only on the previously visited state (or states) and the probability distribution on the transitions between states. More precisely, a MMC is composed of:

- A set of states \( P = \{p_1, \ldots, p_n\} \), in which each state is a point of interest. POIs are usually learnt by running a clustering algorithm on the mobility traces generated by the GPS of an individual or by taking the list of visited ID antennas from GSM mobility traces. These states generally have an intrinsic semantic meaning and therefore semantic labels such as “home” or “work” can often be inferred and attached to them.

- Transitions, such as \( t_{i,j} \), represent the probability of moving from state \( p_i \) to state \( p_j \). A transition from one state to itself is possible if the individual has a non-null probability for moving from one state to an occasional location before coming back to this state. For instance, an individual can leave home to go to the pharmacy and then come back to his home. In this example, it is likely that the pharmacy will not be extracted as a POI by the clustering algorithm, unless the individual visits this place on a regular basis. Figure 3.1 depicts an example of MMC of a user of the ARUM dataset, where POIs are numerated from 0 to 13.

- The stationary vector \( \pi \), which is known also as the steady state or the stationary distribution, is a column vector obtained by multiplying it by the transition matrix repeatedly until convergence. The meaning of the vector depends on whether the used dataset contains mobility traces issued from GPS or GSM. In the former case, it means that \( \pi \) represent the percentage of time spent in a certain POI, during the data gathering period. In the latter case, it is the probability to send or receive a call or SMS in a given POI. For instance the stationary vector of the MMC presented in Figure 3.1 is given by the vector: \( \pi = [0.01, 0.01, 0.01, 0.02, 0.02, 0.71, 0.04, 0.01, 0.01, 0.13] \).

- The order corresponds to a finite memory \( n \) to avoid a memoryless model, which “forgets” the previous visited locations [GKdPC12]. More precisely, the \( n \)-th order represents the \( n \) previous states the model recalls, from the current state and satisfies the property: \( Pr(p_{n+1} | p_n, p_{n-1}, p_{n-2} \ldots p_{n-m}) \) for \( n > m \). Namely, the next state \( p_{n+1} \) depends on the \( n \) previous states \( p_n, p_{n-1}, p_{n-2} \ldots \). For instance, when we use a second order model to forecast the next locations visited by an individual, we observe his actual position “Bar” and the previous one “Work”. Hypothetically, we can infer that this user will go to
“Home” due to the fact that he came from “Work” to “Bar”. An example of a second order MMC is depicted in Figure 3.2, in which the format of a label of a state means [previous state]_[actual state]. For instance, we take the state at top of the graph 0_2, which represents the fact that user has come from POI 0 to get into POI 2 and the transition weighted with 0.5 to state 2_3 denotes the user’s transition from POI 2 to POI 3.

- A time slice \((t)\) is a segment of time in which a \(n\)-th order mobility Markov chain is build [GKNndPC11b]. This abstraction allows us to construct a richer model by introducing the notion of time into the MMC to capture mobility within a time window. For example, Figure 3.3 represents a MMC with time slices \((t = 8)\), which illustrates the mobility of a person in weekdays and weekends in time windows from 0h to 5h59, 6h to 11h59, 12h to 17h59 and 18h to 23h59. On this example, the user only has mobility traces registered between 12h to 23h59 on weekdays and from 18h to 23h59 on weekends, which make only 3 time windows in the figure. In this particular example, the format of the states is given by the POI number and a letter for a given time slice. Thus, we have the POI 2 on weekdays from 12h to 17h59 (2_C), from 18h to 23h59 (2_D) and on weekends from 12h to 17h59 (2_G).
3.2. Mobility Markov chain

Figure 3.2: A 2nd order MMC of a user of the ARUM dataset.

Note that many mobility models relying on Markov chains have been proposed [AS03, GKNndPC11a, DMDBP08], including the use of hidden Markov models for performing inference attacks [YCP’11b]. In a nutshell, building a MMC is a two step process. During the first phase, an algorithm is run to extract the POIs from the mobility traces. Depending on the input data, if mobility traces are
from GSM antennas then POIs (antennas ID) are already extracted. In the case that mobility traces are from GPS a clustering algorithm is needed to perform the extraction task. See Chapter 2 for a description of the various clustering algorithms. In our study [GKNndPC11a], a clustering algorithm called Density Joinable cluster (DJ-Cluster) was used, but of course other clustering algorithms are possible. In the second phase, the transitions between those POIs are computed. To realize this, the trail of mobility traces is examined by chronological order and each mobility trace is tagged with a label that is either the number identifying a particular state of the MMC or the value “unknown”. Finally, when all the mobility traces have been labeled,
3.2. Mobility Markov chain

the transitions between states are counted and normalized by the total number of transitions in order to choose the parameters values. More formally, Algorithm 10 details the sequence for building a MMC. First, the process takes as input a trail of mobility traces, a set of boolean variables to indicate if traces are issued from GPS or GSM and if consecutive repeated traces must be squashed, the number of time slices $t$ and an array of the orders for each time slice. Then, we verify if traces are issued from GPS in order to extract POIs using a clustering algorithm or the antennas ID. Next, a $n$-th order MMC model is created for each time slice $t$. After that, consecutive repeated traces could be squashed before assigning traces to the corresponding time slice. Finally, the transition matrix and stationary vectors for each time slice as well as for the general MMC are computed. For the sake of clarity, we detail how the transition matrix is obtained. We use the MMC of Figure 3.3 for the illustration. Given the trail of mobility traces $MT = \{7_C, 2_C, 2_D, 7_D\}$, in which “C” and “D” represent two different time slices, we take the first two traces $7_C$ and $2_C$, as both correspond to the same time window, we mark the transition in the transition matrix belonging to the time slice “C” (local transition matrix) as well as in the global transition matrix. Then, the next transition from $2_C$ to $2_D$ is marked only in the global matrix as a temporal transition from “C” to “D” time slices as the two traces are not in the same time window.

**Algorithm 10 Construction of a MMC**

**Require:** $D$ a trail of (mobility) traces, $t$ the number of time slices, $fromGPS$ indicating if the traces are issued from GPS, $n[t]$ array of the number of previous location kept, $squash$ is a boolean variable to squash traces.

Extract POIs $ListPOI$ using Clustering algorithm

for each element $n$ in $n[]$ do
    Create the MMC in $ListMMC$
end for

Label $D$ spatiotemporal $ListSpatiotemporalLabels$

if $squash$ then
    Delete repeated consecutive traces in $ListSpatiotemporalLabels$
end if

for each element $mmc$ in $ListMMC$ do
    Assign traces from $ListSpatiotemporalLabels$ which correspond to $mmc$ time slice
    Compute transition matrix of $mmc$
    Compute stationary vector of $mmc$
end for

Compute global transition matrix $g – mmc$

Compute global stationary vector of $mmc$

return the Mobility Markov chain computed

Note that MMC can be either represented as a transition matrix (cf. Table 3.1) or in the form of a graph (cf. Figures 3.1, 3.2, 3.3) in which nodes correspond to states and arrows represent the transitions between states with their associated probability. When the MMC is represented as a transition matrix of size $n \times n$, the rows and columns correspond to states of the MMC while the value of each cell is the probability of the associated transitions between the corresponding
states. Therefore, our model is composed of a general transition matrices and as many local transitions as time slices. The general transition matrix represents spatio (i.e., from one state to another) and temporal (i.e., from one time slide to another) transitions. Local transition matrices represent spatial transitions between POIs within a given time slice (e.g., from 8 AM to 14 PM on weekdays).

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

Table 3.1: 12 × 12 transition matrix of the graph represented in Figure 3.1.

Transition matrix could contain many zeros, which correspond to unobserved transition between those two states in the trail of mobility traces. However, a zero value does not denote the impossibility to go from one POI to another, just the absence of observed transitions. Another interesting feature of this model is the analysis at different granularities. We use Figure 3.4 to explain the concept of granularity. At the top of the figure we observe the MMC of Bob, who travels mostly among four different cities (Aix-en-Provence, Cusco, Rennes and Toulouse). The parameters of the clustering algorithm for building a MMC allow to descend in a fine granularity as shown at the bottom of the Figure 3.4, which is a zoom over Toulouse, where we can observe the POIs and even associate a semantic to a POI like home, old home, work and shopping. The Mobility Markov chain described in the current chapter, the extraction of POIs (cf. Chapter 2), the next location prediction (cf. Chapter 4), the semantic extraction of POIs (cf. Chapter 4) and the de-anonymization inference attack (cf. Chapter 5) were implemented in a framework to provide a tool for privacy evaluation. This framework is described in the next section.

3.3 GEoPrivacy Enhanced TOolkit (GEPETO)

The global objective of GEPETO is to provide researchers concerned with geoprivacy with means to evaluate various sanitization techniques and inference attacks on geolocated data. GEPETO provides an interface for the management of geolocated data and offers several ways to manipulate this data such as sampling mechanisms, sanitization algorithms, inference attacks and a visualization tool to display this data on a world map. The main idea is to offer a generic and flexible tool so that anyone can easily plug a new sanitization technique or a
smart inference algorithm to attack geoprivacy. Moreover, the utility and visualization components provide means to evaluate the benefits of sanitization with regard to the success of inference attacks. In the same vein, other tools that are similar to GEPETO. For instance, M-Atlas is a knowledge discovery process system proposed by Trasarti et al. [TRP+10], which provides mobility traces ($MT$) storage, spatiotemporal queries over $MT$, trajectory models, patterns construction and storage to be reused and/or combined. Finally, new mining algorithms (e.g., inference attacks) could be added. This system also proposes a Data Mining Query Language [TGN+11]. The Location-Privacy Meter [STLBH11] designed by Shokri et al. quantifies the location privacy of users. In a nutshell, this privacy metric is related to the success of an adversary in performing an inference attack. More precisely, the tool gets as input a tuple composed of a set of users $U$, the trail of mobility traces $MT$ of $U$, the sanitization algorithm (applied to the $MT$), the sanitized trail of mobility traces $MT'$, the adversary’s model and a metric to evaluate the reached privacy. Then, in order to evaluate the privacy level, the adversary (modeled as a Hidden Markov chain) makes an inference attack over the $MT'$ to find a probability distribution of locations that depends on the objective of the attack e.g., tracking attack or meeting disclosure. Finally, the metric to evaluate the correctness of the attacker is the distance between the output of the attack and the ground truth. Similarly, Lee et al. [LGY13] propose a framework taking into account 3 dimensions: user, location privacy mechanism and evaluation metric. In this model, the movements of a user are protected by a location privacy mechanism (e.g., anonymity and obfuscation) and effectiveness of the attack is measured by the uncertainty and inaccuracy of the attack. These two last tools have been implemented after GEPETO. In the following subsections, we detail the design and architecture of GEPETO.
3.3.1 GEPETO Design

GEPETO [GKNndPC10] is designed following a multi-layer architecture with the intended goal of making the system functional, efficient, scalable, easily modifiable and reliable (c.f., Figure 3.5). First, the data layer is a set of classes, which manages the communication with the database server for inserting, updating and deleting location data. A control layer is in charge of the presentation, the local management and control of the data and provides a model of the data. The application layer manages the utility functions, the inference attacks and sanitization techniques are implemented. Finally, the visualization layer constitutes the graphical user interface of GEPETO in which the user can load data, apply algorithms and visualize the results. This layered architecture is targeted to provide a good separation of concerns between data access and data presentation, so that it is easy to implement new algorithms in the application layer, access and visualize data using the services of the control and the presentation layers. In GEPETO, the presentation layer uses external web-services for the visualization of the data such as Open Street Maps or Yahoo Maps. The design choices behind this architecture imply both benefits and drawbacks: GEPETO cannot be used offline as it needs access to the database server as well as to the Internet in order to visualize data or to have a local maps repository (i.e., an Open Street Map mirror), but the implementation and maintenance are handled more easily this way, with a clear separation between the database and the visualization parts.

3.3.2 GEPETO Implementation

GEPETO [GKNndPC10a] is an open source software\(^8\) implemented in Java (JSDK 6.0) to make it independent from the operating system and designed following an object-oriented methodology with an iterative approach during development. The final design includes 20 packages,

---

\(^8\) \url{homepages.laas.fr/mnunezde/gepeto/code_source/Gepeto_small.zip}
with a total of 143 classes. Currently, we have implemented the following 5 clustering algorithms, 5 sanitization techniques and 4 inference attacks:

• Clustering algorithms. For more details about clustering algorithms we refer the reader to Chapter 2.
  – $k$-means clustering on a single trail of traces that can be parameterized by $k$ the target number of clusters;
  – Density Joinable cluster receives as input a minimal number of points and a radius;
  – Density Time cluster, which takes as input a radius, a merging distance as well as a tolerance rate.
  – Time Density cluster takes the same inputs of the Density Time cluster in addition to a tolerance rate.
  – DBSCAN takes as input a minimal number of points and a radius.

• Sanitization techniques. For more details about sanitization techniques refer to Section 1.6
  – Downsampling, with the time window in seconds as an input parameter;
  – Pseudonymization, in which a seed to the pseudo-random generator can be given (so that an experiment can be repeated under the same conditions);
  – Random perturbation (with Gaussian noise), the standard deviation for the perturbation and a seed for the pseudorandom generator are taken as input parameters;

• Inference attacks. For more details about inference attacks we refer the reader to Chapter 4 and Chapter 5.
  – Begin and end location finder, parameterized by the duration of a break in seconds;
  – Point of interest extraction, which depends on clustering algorithms (cf., Chapter 2).
  – Prediction of the next location relies on Density Joinable cluster algorithm and the order of the model.
  – De-anonymization attack based on Density Joinable cluster.

• Mobility model. For more details about the MMC mobility model we refer the reader to Section 3.2.
  – Mobility Markov chain, which parameters depend on the clustering selected algorithm.

GEPETO has been explicitly designed to be extendable and it is easy to add new classes, which implement another sanitization or inference algorithm. The scalability is highly dependent on the memory available to run the algorithms and to process the given load of data. Indeed, the larger the volume of data to process is, the larger the computational resources need to be. Currently, we have worked with 4Gb of memory in order to run the implemented algorithms on a dataset with about 10 millions of mobility traces. The database, which stores the location data,
was implemented in MySQL 5.0.37. The sanitization process is often performed in several incremental steps starting from the original data and then applying a first sanitization algorithm “A”, storing the intermediate result, and then applying a second algorithm “B” on the resulting data. For instance, the user may first pseudonymize the data, then perform a downsampling, before perturbing it and clustering it. The visualization part of GEPETO allows to get a clear picture of the data evolution from the original geolocated data to a desired sanitized data. As depicted in Figure 3.6, GEPETO is able to process location data (i.e., mobility traces) issued from GPS or CDR. Then, this data is parsed using JavaCC, in which Table 3.2 summaries the fields we use in the different datasets to be inserted in our database. The privacy analysis tool is able to manipulated data in the DB in order to run previously mentioned sanitization, attack or clustering algorithms. In the next section, we present a MapReduced version of GEPETO that can handle millions of mobility traces.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arum</td>
<td>Mac address, Latitude, Longitude, Altitude, Timestamp</td>
</tr>
<tr>
<td>Geolife</td>
<td>User ID, Latitude, Longitude, Timestamp</td>
</tr>
<tr>
<td>Nokia</td>
<td>User ID, Latitude, Longitude, Timestamp</td>
</tr>
<tr>
<td>LifeMap</td>
<td>User ID, Latitude, Longitude, Timestamp, Important place</td>
</tr>
<tr>
<td>San Francisco Yellow Cabs</td>
<td>Cab ID, Latitude, Longitude, Altitude, Timestamp</td>
</tr>
<tr>
<td>Borlange</td>
<td>Vehicle ID, Latitude, Longitude, Timestamp</td>
</tr>
</tbody>
</table>

Table 3.2: Extracted data fields from different datasets.
3.4 Mapreducing GEPETO

Efficiently analyzing large geolocated datasets composed of millions of mobility traces requires both distribution and parallelization. In particular, partitioning the dataset into independent small chunks assigned to distinct nodes will speed up the execution of algorithms implemented within GEPETO. Therefore, we believe that these algorithms (in particular the clustering ones) represent good candidates to be abstracted in the MapReduce formalism. In the work of Gambs et al. [GKMndPC13], we describe how some algorithms of GEPETO can be adapted to the MapReduce programming model. Basically, this adaptation requires to structure the algorithm/application into Map/Reduce phases (which may be difficult task), before implementing it on top of Hadoop. More precisely, a developer must define three classes that extend the native Hadoop classes and interfaces:

- The Mapper class implements the map phase of the application,
- The Reducer class deals with the reduce phase of the application and
- The Driver class specifies to the Hadoop framework how to run and schedule the MapReduce processes.

The mapreduced algorithms are $k$-means clustering, DJ cluster, downsampling and R-Tree, which are described in the following subsections.

3.4.1 Clustering algorithms

In the current subsection we present the mapreduce version of the classical $K$-means and the DJ cluster algorithms implemented in the mapreduce version of GEPETO.

$k$-means clustering

$k$-means algorithm [M*67] is a classical clustering algorithm partitioning a set of objects (e.g., datapoints) into $k$ clusters of objects that are similar, by trying to minimize the average distance between the objects in a cluster and its centroid, hence the name $k$-means. An object is generally represented as a vector of $d$ attributes, thus corresponding to a data point in a $d$-dimensional space. The algorithm takes as input a dataset composed of $n$ points and $k$ the number of clusters returned by $k$-means. The output of $k$-means is the $k$ clusters as well as their respective centroids. The parameter $k$ has to be specified by the user or inferred by cross-validation. The user also defines a distance metric that quantifies how close or far are two points relative to each other. Typical examples of distance include the Euclidean distance and the Manhattan distance (i.e., L1 norm). The generic sketch of $k$-means is the following:

1. Randomly choose $k$ points from the input dataset as initial centroids of the $k$ clusters (initialization phase).
2. For each point, assign it to the cluster corresponding to the closest centroid (assignment step).
3. For each cluster, compute the new centroid by averaging the points assigned to this cluster (update step).

4. Repeat from step 2) until convergence (i.e., clusters are stable or after a bounded number of iterations).

In general, averaging the points assigned to a cluster is done directly by computing the arithmetic mean of all the points dimension by dimension (i.e., attribute by attribute). The algorithm has been proven to converge after a finite number of iterations. The clustering generated by $k$-means is influenced by parameters such as the distance used, the method for choosing the initial centers of the clusters as well as the choice of the parameter $k$ itself. While simple, the $k$-means algorithm can have a high computational time complexity when applied on large datasets $O(ndk + \log(n))$. In which $n$ is the total number of mobility traces and $d$ is the dimension of the data (2 in our case). Other limitations of the algorithm include the possibility of being trapped in a local minimum (i.e., finding the optimal $k$-means clustering is NP-Hard) and its sensitivity to changes in the input conditions. For instance, for two different random initializations of the clusters’ centers, it is highly probable that the two associated clustering outputted by the algorithm will be significantly different, which makes $k$-means non-deterministic. In addition, if it is manually tuned, $k$ the number of clusters must be known before the clustering begins, which may be difficult for users to specify in advance for some type of data. Finally, another drawback of using the mean as the center of the cluster instead of the median, is that outliers can have a sensible impact on a generated center. This can be a significant problem when the input is the geolocated dataset, as a single noisy and uninteresting mobility trace may pull the center of the cluster towards it much more than it should. Consider the situation in which the input dataset is a geolocated one. MapReducing the $k$-means algorithm amounts to MapReducing each iteration of the algorithm, thus implementing each $k$-means iteration as a MapReduce job. The initialization phase of the algorithm randomly picks $k$ mobility traces as initial centroids of the clusters. This phase requires no distribution because it is computationally cheap and can be performed by a single node. In contrast, the two steps of an iteration, the assignment step and the update step, are perfect candidates for being MapReduced. More precisely, the map phase is in charge of assigning each mobility trace to the closest centroid while the reduce phase computes the new centroid of each cluster previously built by the mappers. The program iterates over the input points and clusters, outputting a new directory “clusters-$i$” containing clusters files for the $i^{th}$ iteration. This process uses a mapper/reducer/main as follows:

- **$k$-means Mapper (Algorithm 11)**
  1. Read the input clusters during the *setup()* method from the clusters file.
  2. Assign each input trace to the nearest cluster by using the distance specified by the user.
  3. Output: (intermediate key, intermediate value) (centroid identifier, assigned trace).

- **$k$-means Reducer (Algorithm 12)**
3.4. Mapreducing GEPETO

1. Receive all traces assigned to a cluster (as specified by the intermediate key of this reducer).
2. Compute the new centroid of the cluster by averaging all the traces assigned to the cluster.
3. Verify if the updated centroid differs from the previous one to detect convergence.
4. Write the cluster and its new centroid to the clusters file.
5. Output the new centroid: (output key, output value) (centroid identifier, new centroid).

- **k-means Main (Algorithm 13)**

1. Randomly select $k$ traces as initial centroids and write them to the clusters file.
2. Iterate until all output clusters have converged or until a predefined maximum number of iterations has been reached.
3. For each iteration, a new clusters directory “clusters- i” is produced and the clusters outputted by the current iteration are used as input for the next one.

All these steps as well as the way centroids are updated during each iteration are summarized in Figure 3.7. In the following, we report on the performance obtained when running our

MapReduced implementation of $k$-means on the GeoLife dataset. We designed several testing scenarios in order to measure how varying the input parameters impacts the performance of $k$-means. For each scenario, we tuned parameters such as the dataset size and the distance

Figure 3.7: Workflow for the MapReduced version of $k$-means.
Table 3.3: Arguments for $k$-means.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>11</td>
</tr>
<tr>
<td>maxIter</td>
<td>150</td>
</tr>
<tr>
<td>convergenceDelta</td>
<td>0.5</td>
</tr>
<tr>
<td>distanceMeasure</td>
<td>Squared Euclidean or Haversine</td>
</tr>
</tbody>
</table>

used. For measuring the distance between points, we considered two metrics: the squared Euclidean distance and the Haversine distance. The squared Euclidean distance uses the same formula as the standard Euclidean distance, but without computing the square root part. As a consequence, clustering with the squared Euclidean distance is faster than clustering with the regular one while preserving the order relationship between different points. The Haversine formula [Sin84] computes the distance between two points over the earth’s surface by taking into account the shape of the earth. The metric used to assess the performance of the implementation is the time required to complete one iteration. Note that the number of iterations required by $k$-means to converge depends on the initial selection of centroids. In our experiments, the number of iterations reported corresponds to a rounded average of 3 to 5 trials. The runtime arguments for $k$-means in our implementation are shown in Table 3.3. The $k$-means algorithm was run on two datasets that are actually subsets of the original GeoLife dataset; the first dataset is composed of 90 users and has a size of 66 MB while the bigger one is composed of 178 users and is of size 128 MB. The experiments were carried out on the Parapluie cluster of the Grid’5000 platform. For Hadoop, the namenode was deployed on a dedicated machine, the jobtracker on another machine, and finally the datanodes and the tasktrackers on the remaining nodes (with one entity per machine), leading to a total of 7 nodes overall. Table 3.4 summarizes our results obtained under different testing scenarios, corresponding to different values for the runtime arguments. The fifth column of the table contains the time (measured in seconds) required to run a $k$-means iteration with our MapReduce implementation.

Algorithm 11 $k$-Means Mapper

```java
setup(Configuration conf)
    centroids = load from file

map(K key, V value)
    trace = value
    min = MAX_VALUE
    for center 2 centroids do
        if distance(trace, center) < min then
            min = distance(trace, center)
            assign = center
            emit Intermediate(assign, trace)
        end if
    end for
```

62
3.4. Mapreducing GEPETO

Algorithm 12 $k$-Means Reducer

```java
setup(Configuration conf)
  centroids = load from file

reduce(K key, V[] value)
  for v values do
    if distance(trace, center) < min then
      new_centroid = average(v)
      assign = center
      emit (key; new centroid)
    end if
  end for
```

Algorithm 13 $k$-Means Main

```java
randomCenters(Configuration conf)
  centers = randomly choose k centroids
  write to file

main()
  i = 0
  while true do
    submit MapReduce job for iteration i
    i = i + 1
    if hasConverged or i = maxIter then
      stop
    end if
  end while
```
Chapter 3. Mobility Markov chain and GEPETO

<table>
<thead>
<tr>
<th>Data (MB)</th>
<th>Nb of traces</th>
<th>Distance</th>
<th>Chunk (MB)</th>
<th>Iter. time (sec)</th>
<th>Nb of iter. to converge</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>1.050.000</td>
<td>Haversine</td>
<td>64</td>
<td>57</td>
<td>73</td>
</tr>
<tr>
<td>66</td>
<td>1.050.000</td>
<td>Squared Euclidean</td>
<td>64</td>
<td>48</td>
<td>72</td>
</tr>
<tr>
<td>66</td>
<td>1.050.000</td>
<td>Squared Euclidean</td>
<td>32</td>
<td>41</td>
<td>70</td>
</tr>
<tr>
<td>66</td>
<td>1.050.000</td>
<td>Haversine</td>
<td>32</td>
<td>45</td>
<td>73</td>
</tr>
<tr>
<td>128</td>
<td>2.033.686</td>
<td>Squared Euclidean</td>
<td>64</td>
<td>51</td>
<td>85</td>
</tr>
<tr>
<td>128</td>
<td>2.033.686</td>
<td>Squared Euclidean</td>
<td>32</td>
<td>45</td>
<td>83</td>
</tr>
<tr>
<td>128</td>
<td>2.033.686</td>
<td>Haversine</td>
<td>32</td>
<td>48</td>
<td>89</td>
</tr>
<tr>
<td>128</td>
<td>2.033.686</td>
<td>Haversine</td>
<td>64</td>
<td>60</td>
<td>93</td>
</tr>
</tbody>
</table>

Table 3.4: Result of the MapReduce $k$-means experimentations.

The experiments were carried out on the Parapluie cluster of the Grid’5000 platform. For Hadoop, the namenode was deployed on a dedicated machine, the jobtracker on another machine, and finally the datanodes and the tasktrackers on the remaining nodes (with one entity per machine), leading to a total of 7 nodes overall. Table 3.4 summarizes our results obtained under different testing scenarios, corresponding to different values for the runtime arguments. The fifth column of the table contains the time (measured in seconds) required to run a $k$-means iteration with our MapReduce implementation. We observe that a crucial parameter having a big influence on the computational time is the chunk size. Usually, in Hadoop the chunk size can be set to two values: 32 MB and 64 MB. A smaller chunk size leads to a larger number of chunks, which in turn generates more map tasks. Obviously, a higher number of mappers working in parallel will improve the computational time. Moreover as expected, the Haversine distance increases the execution time of an iteration compared to the squared Euclidean distance due to the more complex computations that the Haversine formula requires.

Density Joinable

DJ-Cluster is detailed in Section 2.2.1. Each of the phases (preprocessing, processing and merging) of DJ-cluster can be expressed in the MapReduce programming model. Thereafter, we describe how each phase can be implemented as one or several MapReduce jobs. The first phase of DJ-Cluster preprocesses the mobility traces by applying two filtering techniques that remove the mobility traces that are not interesting or might even perturb the clustering process. These two filtering techniques have been implemented in the form of two MapReduce jobs executed in pipeline. More precisely, the output of the first job constitutes the input of the second one. During the first MapReduce job, stationary traces are kept while moving ones are discarded. Identifying the moving traces amounts to measuring the speed of each trace and then removing the traces whose speed is higher than a predefined threshold $\epsilon$ (for $\epsilon$ a small value). The speed of a trace is computed as the distance traveled between the previous and the next traces divided by the corresponding time difference. The computation of the speed for each trace can be performed only by map tasks. Each mapper reads its corresponding data chunk and outputs only the traces whose speed is less than. As no aggregation or additional filtering is required,
3.4. Mapreducing GEPETO

<table>
<thead>
<tr>
<th>Sampling rate</th>
<th>Unfiltered</th>
<th>Filter moving traces</th>
<th>Remove duplicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 min</td>
<td>155260</td>
<td>86416</td>
<td>85743</td>
</tr>
<tr>
<td>5 min</td>
<td>41263</td>
<td>23996</td>
<td>23894</td>
</tr>
<tr>
<td>10 min</td>
<td>23596</td>
<td>14207</td>
<td>14174</td>
</tr>
</tbody>
</table>

Table 3.5: Number of traces in the sampled datasets after the preprocessing

the implementation of this part does not include a reduce phase. The second filtering technique removes redundant consecutive traces, which correspond to mobility traces that have (almost) the same spatial coordinate but different timestamps. Similarly to the first filtering technique, only the map phase is needed. The role of the mapper is simply to output the first trace from a sequence of traces that are redundant. Figure 3.8 shows the two pipelined MapReduce jobs implementing the preprocessing phase of DJ-Cluster. These two filtering techniques reduce considerably the amount of data that needs to be processed by the clustering algorithm. In order to measure this reduction of the dataset size, we ran a set of experiments that apply the preprocessing phase on the sampled datasets at different sampling rates of respectively 1, 5 and 10 minutes (see Table 3.5). Table 3.5 shows the number of traces remaining after the preprocessing phase. The value of, the threshold speed, was set to 0.72 km/h (which is equivalent to 0.2 m/s). The main challenge of the second phase of DJ-Cluster is to design an efficient method for discovering the neighbors of a data point. We chose to rely on a data structure known as R-Tree [Gut84], as computing the neighborhood of a point with such a data structure can be done in $O(n \log n)$, for $n$ the size of the dataset. The construction of an R-Tree is described as a part of the third phase of DJ-clusters in the followings paragraphs. The computation of the neighborhood for each trace in the dataset is a good candidate for parallelization and distribution. Indeed, splitting the dataset into small chunks processed by different nodes is likely to speed up the computation process. Before processing its chunk, a mapper first loads the R-Tree from the distributed cache while executing its setup method. For each trace in the chunk, the mapper computes the set of neighbors within a distance $r$ from the current trace. More precisely, the mapper searches for the nearest neighbors of a trace by relying on the R-tree. This searching process traverses mainly the branches of the R-Tree in which neighbors may be located. If the size of the computed neighborhood has less than $\text{MinPts}$ elements, the mapper marks the current trace as noise. The (key; value) pair outputted by the mapper corresponds to a trace and its associated neighborhood. Algorithm 14 describes more precisely this pro-

![Figure 3.8: Workflow for the MapReduced version of DJ cluster.](image)
A single reducer implements the last phase of the algorithm as the merging of joinable neighborhoods must be done by a centralized entity that obtains a knowledge about all current neighborhoods built by mappers. In the intermediate (key; value) pair emitted by mappers, the key field is set to a constant value such that all pairs are collected by a single reducer (i.e., all intermediate pairs that have the same key field are redirected towards the same reducer). This reducer collects all neighborhoods outputted by the mappers and then proceeds to building the clusters. Merging all joinable neighborhoods does the construction of clusters. By definition, two neighborhoods are joinable if there exists at least one trace such that both neighborhoods contain it. The reducer merges all joinable neighborhoods with existing clusters or creates new clusters if a neighborhood cannot be joined with any of the existing clusters. Thus, by the end of the clustering process, each trace is either assigned to a cluster or marked as noise. In addition, the clusters are assured to be non-overlapping and to contain at least $MinPts$ mobility traces. The output of this reduce phase, which is also the output of the algorithm, consists in the computed clusters. All the above steps are shown in Algorithm 15. The third phase consist

**Algorithm 14 DJ-Cluster Mapper.**

```java
setup(Configuration conf)
rTree = load from file in distributed cache

map(K key, V value)
trace = value
neighborhood = rTree.kNN(trace, MinPts, r)
if neighborhood.size < MinPts then
    trace.markAsNoise
    emitIntermediate| (const, neighborhood)
end if
```

**Algorithm 15 DJ-Cluster Reducer**

```java
setup(Configuration conf)
clusters = ∅

reduce(K key, V[] value)
for neighborhood ∈ values do
    for cluster ∈ clusters do
        if neighborhood.intersects(cluster) then
            cluster = cluster.merge(neighborhood)
        else
            new_cluster = neighborhood
        end if
    end for
end for
for cluster ∈ clusters do
    emit (clusterId, cluster)
end for
```
of building a R-Tree to merge clusters. The R-Trees [Gut84] are data structures commonly used for indexing multidimensional data. In a nutshell, an R-Tree groups datapoints into clusters and represents them through their minimum bounding rectangle in their upper level in the tree. At the leaf level, each rectangle contains only a single datapoint (i.e., each rectangle is a point) while higher levels aggregate an increasing number of datapoints. When querying an R-Tree, only the bounding rectangles intersecting the current query are traversed. The indexing of large datasets into an R-Tree structure can be implemented as a MapReduce algorithm [Gut84]. In previous work, each point in the dataset is defined by two attributes: a location in some spatial domain used to guide the construction of the R-Tree and a unique identifier used to reference the object in the R-Tree. The construction of an R-Tree is a process that can be split into three phases:

1. The partition of datapoints into \( p \) clusters.
2. The indexing of each cluster into a small R-Tree.
3. The merging of the small R-Trees obtained from the previous phase into a final one indexing the whole dataset.

The first two phases are MapReduced while a single node due to its low computational complexity executes the last phase sequentially. Figure 3.9 illustrates how these three phases are executed sequentially. More precisely, the first phase computes a partitioning function assigning each datapoint of the initial dataset to one of the \( p \) partitions. This partitioning function should yield equally-sized partitions and preserve at the same time data locality (i.e., points that are close in the spatial domain should be assigned to the same partition). In order to satisfy these constraints, the partitioning function has to map multidimensional datapoints into an ordered sequence of unidimensional values. In practice, this transformation is performed with the help of space-filling curves that precisely map multidimensional data to one dimension while preserving data locality. In our R-Tree construction, we implemented and tested two types of space-filling curves: the Z-order curve and the Hilbert curve [LK00]. The MapReduce design for this phase consists of several mappers and one reducer. Each mapper (Algorithm 16) samples a predefined number of objects from its data chunk and outputs the corresponding single-dimensional values obtained after applying the space-filling curve. Afterwards, the reducer (Algorithm 17) collects a set of single dimensional values from all mappers, orders this set and then determines \((p - 1)\) partitioning points in the sequence (partitioning points delimit the boundaries of each partition). The second phase concurrently builds \( p \) individual R-Trees by indexing the partitions outputted by the first phase. The mappers (Algorithm 18) partition the dataset into \( p \) partitions using the space-filling curve computed during the first phase. Each mapper processes its chunk and assigns each object it reads to a partition identifier. The intermediate key is represented by the partition identifier such that all datapoints sharing the same key (i.e., belonging to the same partition) will be collected by the same reducer. Then, each reducer (Algorithm 19) constructs the R-Tree associated with its partition, leading to a total of \( p \) reducers building \( p \) small R-Trees. Finally, the last phase merges the small R-Trees into a global one that indexes all the datapoints of the initial dataset. For values of \( p \) not exceeding thousands, this process can be executed by a single node. For building the small R-Trees and
then merging them into a global one, we use an existing off-the-shelf implementation. More precisely, the Java Spatial Index library (JSI) is an open-source implementation of an R-tree in Java whose main objective is to provide fast intersection query performance. JSI contains Java classes and structures defining nodes, rectangles and R-trees. It also provides procedures for adding and deleting a node to/from an R-tree and for finding the neighbors within a specific distance from a datapoint. JSI relies on the Trove library for defining high speed, regular and primitive collections for Java. Trove is a fast and lightweight implementation of the java.uti.Collections API. The main idea behind Trove is to use customized hashing strategies. In particular, it is possible to save time with a customized hashing function by skipping portions of the query that remain invariant.

**Algorithm 16** R-Tree First Phase Mapper.

```plaintext
map(K key, V value)
    sample ← randomly sample objects in chunk
    scalar ← space_filling_curve(sample)
    emitIntermediate(const, scalar)
```

Figure 3.9: Workflow for the MapReduced version of $d$. 
3.4. Mapreducing GEPETO

Algorithm 17 R-Tree First Phase Reducer

```java
reduce(K key, V[] value)
    order(values)
    step = total_samples \div R
    count = 0
    for i \in values.size do
        if i mod step = 0 then
            emit (count, values[i])
            count = count + 1
        end if
    end for
```

Algorithm 18 R-Tree Second Phase Mapper(partitionId, value).

```java
setup(Configuration conf)
    partitions = load ouput of first phase
    map(K key, V value)
        scalar = space_filling_curve(value)
        for i \in [0 : R] do
            if partitions[i] < scalar < partitions[i + 1] then
                partitionId = i
                break
            end if
        end for
        emit Intermediate(assign, trace)
```

Algorithm 19 R-Tree Second Phase Reducer

```java
reduce(K key, V[] value)
    partitionId = key
    tree = build_RTree(values)
    emit (tree, treeRoot)
```
3.4.2 Sanitization techniques

(Down)sampling is a form of temporal aggregation in which a set of mobility traces that have occurred within a time window are merged into a single mobility trace, which is called the representative trace. Thus, sampling summarizes several mobility traces into a single one. Considering a time window of size $t$ composed of a sequence of mobility traces that have occurred during this time window, we have implemented two sampling techniques. The first sampling technique takes the trace closest to the upper limit of the time window as the representative one (Figure 3.10) while the second technique chooses the trace closest to the middle of the time interval (Figure 3.11). Both sampling techniques have been implemented as MapReduce applications consisting only of map phases. The reduce phase is not necessary as sampling represents a computationally cheap operation and can be performed in a single pass. Each map task reads its input chunk and processes each line of the chunk, corresponding to a mobility trace. In GeoLife, each trace is composed of the location and the timestamp, plus additional information about the speed associated to this trace. All the mapper tasks process in parallel their respective chunks by executing the same code. In a nutshell, for each time window, the mapper artificially generates a “reference trace”, which is either the trace located at the end or at the middle of the time window depending on the used sampling technique. Afterwards, the current mobility trace read from the chunk is compared against the reference trace. Then, the
result of this comparison determines if this trace is kept or not, as only the trace closest to the reference trace is outputted by the mapper. The initial GeoLife dataset of 1.61 GB is split into 64 MB-size chunks, leading to a total of 26 mapper tasks. The experimental testbed consisted of 7 nodes on the Parapluie cluster, thus each node executes approximately 4 mapper tasks. The user can specify as input parameters: the size of the considered time window and the desired sampling technique as well as the input and output folders. For a time window of 10 seconds, the completion of the sampling process on the overall dataset takes 1 minute and 4 seconds. Table 3.6 summarizes how the size of the dataset changes after sampling at different rates (namely 1 minute, 5 minutes and 10 minutes). It can be observed that the number of traces decreases drastically even when downsampling with a rate of 1 minute, which is not really surprising due to the dense nature of the GeoLife dataset in which the GPS logs were collected every 1 to 5 seconds.

<table>
<thead>
<tr>
<th>Initial dataset</th>
<th>1min sampling</th>
<th>5min sampling</th>
<th>10min sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>2033686</td>
<td>155260</td>
<td>41263</td>
<td>23596</td>
</tr>
</tbody>
</table>

Table 3.6: Number of traces in the GeoLife dataset under different sampling conditions: no sampling, sampling rates of 1, 5 and 10 minutes.

In this chapter, we have proposed to adopt the MapReduce paradigm in order to be able to perform a privacy analysis on large scale geolocated datasets composed of millions of mobility traces. More precisely, we have developed a complete MapReduce-based approach to GEPETO, a software that can be used to design, tune, experiment and evaluate various sanitization algorithms and inference attacks on location data as well as to visualize the resulting data. Most of the algorithms used to conduct an inference attack represent good candidates to be abstracted in the MapReduce formalism. For instance, we have designed MapReduced version of sampling as well as the k-means and the DJ-Cluster clustering algorithms and integrate them within the framework of GEPETO. These algorithms have been implemented with Hadoop and evaluate on a real dataset. Preliminary results show that the MapReduced versions of the algorithms can efficiently handle millions of mobility traces.

### 3.5 Summary

In the current chapter, we have presented the existing mobility models in the literature. Then, our mobility model called Mobility Markov chain has been introduced. Finally, the tool that implements inference attacks and sanitization algorithms, GEPETO, as well as its Mapreduce version are described. Since we have defined our mobility model, which can be consider as the adversary model to extract new information through inferences, it is possible now to use the MMC model to implement inference attacks previously highlighted in Section 1.5. Thus, the implementations of those inferences attacks are detailed in the next chapters.
Chapter 4

Next place prediction and semantic inference

"Prediction is difficult, especially about the future."

— Niels Bohr.

The current chapter has a twofold objective. First, we address the issue of predicting the next location of an individual based on the observations of his mobility behavior over some period of time and the recent locations that he has visited. More precisely, we base our attack on the Mobility Markov Chain (MMC) mobility model (cf. Chapter 3). We design a novel algorithm for next location prediction based on this mobility model. This work has several potential applications such as the evaluation of geo-privacy mechanisms, the development of location-based services anticipating the next movement of a user and the design of location-aware proactive resource migration. The evaluation of the efficiency of our algorithm on three different datasets demonstrates an accuracy for the prediction of the next location in the range of 70% to 95% as soon as \( n = 2 \). Another complementary attack to the point of interest extraction presented in Chapter 2 is discussed in the current chapter. This attack is the semantic extraction of the points of interest of people. The semantic extraction we present is accomplish trough heuristics and based on the MMC structure. The remainder of this chapter is organized as follows. First, we present the next place prediction inference attack as well as some related works and experiments in Section 4.1. Afterward, in Section 4.2 we discuss about the semantic extraction and finally Section 4.3 summaries the chapter.

4.1 Prediction

In this section, we address the issue of predicting the next location of an individual based on the observations of this mobility behavior over some period of time and the recently visited locations. This inference has several potential applications such as the evaluation of geo-privacy mechanisms, route recommendation, 3D maps download for vehicles navigation, location-aware proactive resource migration. The evaluation of the efficiency of our algorithm on three different datasets range of 70% to 95%.
4.1.1 Related Work

In this subsection, we describe related work on location prediction based on Markov models [AS03, AMSS11], raw trajectories [KH06, MW04] and semantic trajectories [YLWT11], as well as compression techniques [SQBB10]. We also discuss the results of studies comparing location predictors [PBTU06, SQBB10] and review the literature on the modeling of human mobility.

Markov models

There is a family of predictors that represents the mobility behavior of an individual as a Markov model and predicts the next location based on the previously visited locations [AS03, AMSS11, VK96]. For instance, Ashbrook and Starner [AS03] have built a method for predicting future movements that first extracts POI using a variant of the \( k \)-means clustering algorithm on the individual’s mobility traces. Afterward, a Markov model is computed in which each node is a POI and the transition between two nodes corresponds to the probability of moving from one POI to another. This work is very similar in spirit to our contribution presented in Gambs et al. [GKNdPC11a, GKdPC12]. However the major difference between this previous work and the study of Gambs et al. relies on the clustering algorithm used to discover the POIs. Indeed, Ashbrook and Starner have used the standard \( k \)-means algorithm while Gambs et al. relies on a clustering algorithm tailored for location data called DJ-cluster [GKdPC12] depending on the number of states the MMC is able to recall (i.e., the order of the Markov chain). The experiments on three different datasets show that the accuracy of this prediction algorithm ranges from 70% to 95%. Finally, while Ashbrook and Starner proposed a method for learning a model for next place prediction, they did not actually assess its accuracy on a real dataset. A variant of Markov model called the Mixed Markov-chain Model (MMM) [AMSS11] has recently been proposed for next place prediction. This approach considers that standard Markov Models (MM) and Hidden Markov Models (HMM) are not generic enough to encompass all types of mobility. Therefore, the concept of MMM was proposed as an intermediate model between individual and generic models. The prediction of the next location is based on a Markov model belonging to a group of individuals with similar mobility behavior. This approach clusters individuals into groups based on their mobility traces and then generates a specific Markov model for each group. The prediction of the next location works by first identifying the group a particular individual belongs to and then inferring the next location based on this group model. This approach was tested on a synthetic dataset of pedestrian movements simulated through a pedestrian-flow algorithm as well as on a real dataset of 691 participants moving in a mall. These experiments show an accuracy for the prediction task (as measured by the ratio between the correct and total number of predictions) of 74,1% for MMM, between 16,9% to 45,6% for MM and between 2,41% to 4,2% for HMM.

Raw and semantic trajectories

Ying et al. have proposed to integrate semantic information about the places visited by an individual in addition to its location data in order to enhance the accuracy of the prediction about his future location [YLWT11]. The proposed approach relies on the notion of semantic trajectories, which represents the mobility of an individual as a sequence of visited places tagged with
semantic information. For instance, the semantic tags can be “home”, “favorite restaurant” or “sport center visited on Wednesday and Friday”. To support the prediction of next location based on semantic trajectories, the authors have developed a framework called SemanPredict, which is composed of two modules. The offline mining module extracts the semantic trajectories from raw data by first computing the stop points of a trajectory [ABK’07b], which corresponds to POIs in which the user has stayed more than a certain amount of time. Afterward, the offline module queries a Geographic Semantic Information Database (GSID) storing information of landmarks collected via Google Maps in order to attached semantic information to the stop points. The stop points and semantic trajectories are then stored in a tree-like structure. After that, the semantic trajectories are clustered using an algorithm based on the Maximal Semantic Trajectory Pattern (MSTP) similarity [YLL’10]. The online prediction module is responsible for matching the current trajectory of a user with the closest trajectory in the database by relying on the geographical and semantic features. This module computes a similarity measure combining a geographical and semantic score quantifying the closeness between two trajectories. A partial matching strategy is applied on the tree-like structure in order to identify the closest trajectory. Finally, the prediction of the next location is simply the child node of the candidate trajectory with the highest similarity. For their experiments, authors use MIT reality dataset, which contains communication, proximity, location, and activity information from 100 subjects at MIT over the course of the 2004-2005 academic year. Finally, to evaluate the model they compare the precision, recall and f-measure (cf. Subsection 2.5.1) of the “SemanPredict” model face to Geographic-Only (GO) and Full-Matching (FM) prediction strategies. The former is an adaptation of “SemanPredict”, which skips the semantic mining step to generate a baseline metric. The later baseline metric is generated from traditional matching method instead of partial-matching. Results are summarized in table 4.1. Some other works also rely on the notion of trajectories to predict the next location. For instance, Krumm and Horvitz have developed a tool called Predestination [KH06], which aims at predicting the destination of a trip based on the trajectory information gathered so far. In a nutshell, this method divides the spatial area into a grid in which each cell has a surface of 1 km$^2$ and then counts the number of times an individual has visited each cell. From this information, Predestination computes a probability distribution representing the likelihood of visiting a particular cell. This probability distribution is then used later to predict the most likely destination of a trip. The experiment involved GPS driving data gathered from 169 different subjects who drove 7 335 different trips during 2 weeks in Seattle [KH05]. The authors employ a training dataset to find a probability distribution (model) and test dataset to measure the mean error in kilometers. Thus, the mean

<table>
<thead>
<tr>
<th>Model-Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemanPredict</td>
<td>0.8</td>
<td>0.9</td>
<td>0.85</td>
</tr>
<tr>
<td>GO</td>
<td>0.68</td>
<td>0.8</td>
<td>0.73</td>
</tr>
<tr>
<td>FM</td>
<td>0.68</td>
<td>0.4</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of the precision, recall and F-measure between SemanPredict, Geographic-Only and Full-Matching.
error found with the whole population was 1Km. In the same line of research, Froehlich and Krumm have designed an approach predicting the next location based on the trip similarity [FK08], which displays an accuracy of 85%. The preprocessing step of this method extracts the trajectory by removing mobility traces that correspond to movements (as measured by the speed computed from the previous and the next mobility trace), as well as redundant, erroneous or out of bounds mobility traces.

### Compression techniques

Some predictors are based on compression techniques such as the Lempel-Ziv family of algorithms [FMG92, AMSS11]. These methods use an incremental parsing algorithm [ZL78] normally used in the context of text compression. Other methods such as the prediction by partial matching combine at the same time on compression techniques and Markov models [CIW84]. To predict the next location, this method creates different Markov models incorporating the knowledge of the previous \( n \) locations visited. Sampled Pattern Matching (SPM) prediction [JSA00] is similar in spirit to the Markov chain models that takes as input the \( n \) last locations to predict the next one. In SPM, \( n \) is based on the longest sequence found. \( n \) is modulated by a constant \( \alpha \), which can take values between 0 and 1. For instance, let us consider a set of locations (i.e., GSM cell IDs) \( L_{id} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \) an the input parameters \( n=8 \) and \( \alpha=0.5 \) for the location sequence 54372214 62535437 8255437 91543722762535437. Thus, as \( n=8 \), the longest sequence found is [62535437]. Then, we have considered \( \alpha = 0.5 \), which means that the “flag” is the half of the whole sequence 5437. Afterward, we enumerate places visited after the flag 5437, ' which in our example are 2, 8, 9, 2 respectively. Finally, the predicted location after the flag is 2 because is the most frequented place in this example.

### Classification

The task of predicting the next location can be rephrased as a classification task. For instance, Etter et al. [EKK12] use a 2-layers Artificial Neural Networks (ANN) to compute the probability of the next whereabouts of a user. More precisely, the authors pass as input to the ANN the concatenation of 3 vectors (place, hour and day) for each user, of the Nokia challenge dataset [Kiu09], to obtain as output the probability distribution of places that user can potentially visit. The accuracy of this method was 60,83%. In the same spirit, Wang and Prabhala [WP12] use support vector machines (SVM) to predict next location. In more detail, they elaborate a set of binary vectors taking into account information about the time such as the day of the week and the hour of day with the next place to visit (i.e., label) as training set. Using this technique the authors obtained an average accuracy of 55,6%. However, the work of Gao et al. [GTL12] shows that the probability of being in a certain location depends on the current time and not on only previous positions. Then, they make the hypotheses that time can be divided in hour of a day and day of the week variables, which are independent. Thus, the authors model the probability of visiting a given location using Gaussian distributions for hours and days. Accordingly, the probability of being a certain location depends on the previous visited place and the distributions of the hour and day. To compute the aforementioned probability, they apply the Hierarchical Pitman-Yor language model [Teh06] obtaining an accuracy of 50,53%.
### 4.1. Prediction

<table>
<thead>
<tr>
<th>Technique</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Bayesian network</td>
<td>78.85%</td>
</tr>
<tr>
<td>multilayer perceptron</td>
<td>76.45%</td>
</tr>
<tr>
<td>Elman net</td>
<td>79.68%</td>
</tr>
<tr>
<td>Markov predictor</td>
<td>76.53%</td>
</tr>
<tr>
<td>state predictor</td>
<td>70.89%</td>
</tr>
<tr>
<td>Markov predictor with counter</td>
<td>81.14%</td>
</tr>
<tr>
<td>State predictor with counter</td>
<td>81.88%</td>
</tr>
</tbody>
</table>

Table 4.2: Accurate comparison of different methods

Comparing the efficient of different predictors

Song et al. have conducted a study comparing four different families of location predictors [SQBB10] tested on a dataset gathered by the Dartmouth College from WiFi users between April 2001 and March 2003. Based on the results obtained, the authors conclude that more complex predictors are not necessarily significantly more accurate than Markov predictors. They also established that Markov predictors beyond the second order (i.e., basing their predictions on 3 or more locations) are less precise. In another study Petzold et al. compare 7 prediction methods [PBTU06] dynamic Bayesian network, multilayer perceptron, Elman net, Markov predictor and state predictor. They tested the methods on a dataset that contains movements of four persons through an office building. The movement data were measured from July 2003 to January 2004 at the fourth floor of the building of the Institute of Computer Science at the University of Augsburg. In order to value all techniques authors uses two dataset, one for training (summer season) and another one for test (fall season), to computes the accuracy of each method by counting the number of correct prediction divided by the total number of predictions. Authors compare the accuracy of the above mentioned techniques, which are resume in the table 4.2. Chon and co-authors [CSTC12] evaluate 3 mobility models: location dependant based on Markov model taking into account the n last visited places. Then, location independent method uses nonlinear time series. The basic idea is that people tend to repeat temporal patterns such as leaving places at 6PM for example. Afterward, the third model based on feature added techniques is a set of methods composed of time-aided scheme, which assumes that duration of stay in a POI is dependent on arrival time, the fallback mechanism that adapts the number of places to remember when using the Markovian predictor and the return-probability-aided scheme, which relies on temporal periodicity of location visits. For instance, people go back home at 9PM no matter where they were before. To evaluate these models, the authors used GPS traces from 12 students, at Yonsei University in Seoul, Korea over 2 months [CC11a]. The metric to evaluate the model is the accuracy. The authors show that first order Markov model obtains 68% of accuracy (higher-order models generate significantly fewer predictable gain). While the time-aided and return probability-aided schemes show robust predictability of about 60% of accuracy. On the theoretical side, Barabasi et al. [GHAL08] have analyzed the predictability of the human mobility using three different entropy measures. Their approach first constructs a graph in which each node is associated with the percentage of time spent in a cell. Then,
Chapter 4. Next place prediction and semantic inference

three entropy measures (random $S_{ran}$, temporal-uncorrelated $S_{un}$ and actual entropy $S$) are computed for each user. Afterward, the probability distributions of the three proposed entropy measures $P(S_{ran})$, $P(S_{un})$ and $P(S)$ are computed in order to characterize the predictability of the population. Finally, a predictability score is computed, which represents the accuracy of the prediction of future whereabouts. This predictability score is derived from the Fano’s inequality and quantifies the entropy of a particular user moving between $n$ locations. To understand the high potential predictability, authors segmented each week in hours (168 h per week) to identify the most visited places per user per hour. On one hand, they measure each user regularity $R$, which is the probability to find a user in the most visited location during that hour. On the other hand, they compute the number of locations visited per hour $N(t)$. In their experiments, the authors have used a sample of 45 000 users of mobile phones registered during a period of 3 months. This dataset was collected by an American phone company for billing purpose, recording the user in a cell when he uses his phone. After that, this information was anonymized for the study. The space used for this study was discretized using coverage areas of cell towers (of about $3Km^2$ each). From the combination of the empirically measured entropy and the Fano’s inequality, the authors conclude the human mobility can be predicted with a probability of success of 93% on average.

4.1.2 Next Place Prediction

In order to predict the next location based on the $n$ last positions in the MMC model, we compute a modified form of the transition matrix whose rows represent the $n$ last visited positions. To illustrate the concept of prediction based on a MMC, Table 4.3 and Figure 4.1 respectively show the transition matrix and graphical representation of a 2-MMC learnt on the trail of mobility traces taken from a user of the ARUM dataset [KRT10] that we simply name as Bob to preserve his anonymity. This 2-MMC consists of three different states: “Home” (H), “Work” (W) and “Others” (O) and the goal is to predict the next location based on the two previous locations (i.e., $n = 2$). Thus, the rows of the transition matrix denote all possible combinations of pair of previous locations ($HH$, $HW$, $HO$, $WH$, $WW$, $WO$, $OH$, $OW$, $OO$) while a column represents the next position in the MMC. For instance, if the previous position was $H$ and the current position is $W$, the prediction on the next location will be home $H$ and a transition will occur from state $HW$ to state $WH$, thus updating the previous location to $W$ and the current one to $H$.

The prediction algorithm (Algorithm 20) requires as input the $n$ previous visited locations as well as the mobility model MMC and works in the straightforward following manner. For instance, the input could be a transition matrix such as Table 4.3 and the two previous locations $HO$. Then, we look for the row in the transition matrix corresponding to the set of previous states (see Algorithm 20 line 1). The algorithm finds the row corresponding to these $n$ previous locations and searches the most probable transition (ties are broken arbitrarily). In our example, as the previous locations are $HO$, the prediction is $H$ with a probability of 66%. If there are more than one column with the same maximum probability, we choose randomly (cf. Algorithm 20 lines from 6 to 18). Thus, in our case there is only one column with the greatest probability,
4.1. Prediction

<table>
<thead>
<tr>
<th>Source/Dest.</th>
<th>H</th>
<th>W</th>
<th>L</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>H W</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>H L</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>H O</td>
<td>0.66</td>
<td>0.34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>W H</td>
<td>0.00</td>
<td>0.84</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>L H</td>
<td>0.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.50</td>
</tr>
<tr>
<td>O H</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>O W</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.3: Transition matrix of Bob.

![Graphical representation of the 2-MMC from Bob.](image)

which is the column for home (H), this means that the individual has visited home, as previous location, and he stays at "other place" as the current location. Then, the next place he will go to is H. This can be observed directly from the graphical representation in Figure 4.1 where the node labeled H O is connected to the node O H, which represents the above described transition.

4.1.3 Experimental Evaluation

In this section, we report on experiments conducted to evaluate the accuracy of our prediction algorithm and the theoretical predictability of users. In these experiments, we used three different datasets: ARUM (Phonetic), Geolife and Synthetic, whose characteristics are summarized in Table 1.2 in Chapter 1. In order to assess the efficiency of our location predictors, we compute two metrics: the accuracy and the predictability. The accuracy Acc is the ratio between the number of correct predictions $p_{correct}$ over the total number of predictions $p_{total}$:

$$\text{Acc} = \frac{p_{correct}}{p_{total}}. \quad (4.1)$$
Chapter 4. Next place prediction and semantic inference

Algorithm 20 Prediction using a n-MMC

Require: Transition matrix $M$ of a n-MMC, $n$ previous visited locations

//Search the row $r$ in $M$ corresponding to the $n$ previous visited locations
for each columns $c$ in $TM$ do
    Search the maximal value $\text{max\_value}$ in $TM[index\_line][c]$
end for

Break ties arbitrarily (for instance by randomizing over them)

if there is only a $\text{max\_value}$ column then
    for each columns $c$ in $TM$ do
        search the $\text{index\_column}$ of the $\text{max\_value}$
    end for
    return the prediction $\text{index\_column}$
else
    Create an array of index set $\text{index\_set}$
    for each columns $c$ in $TM$ do
        stock $\text{index\_column}$ of the $\text{max\_value}$ in $\text{index\_set}$
    end for
    Choose randomly the $\text{index\_column}$ in $\text{index\_set}$
    return the prediction $\text{index\_column}$
end if

The predictability $Pred$ is a theoretical measure representing the degree to which the mobility of an individual is predictable based on his MMC (in the same spirit as the work of Barabasi and co-authors [SQBB10]). For instance, if the location predictor knows that Bob was previously at work ($W$) and he is currently at home ($H$), the probability of making a successful guess is theoretically equal to the maximal outgoing probability transition, which is 84% for this particular example (see Table 4.3). More formally, the predictability $Pred$ of a particular MMC (and thus a particular individual) is computed as the sum of the product between each element of the stationary vector $\pi$ of the MMC model, which corresponds to the probability of being in a particular state (for $l$, the total number of states of this MMC) and the maximum outgoing probability ($P_{\text{max\_out}}$) of the $k^{th}$ state:

$$Pred = \sum_{k=1}^{l} (\pi(k) \times P_{\text{max\_out}}(k, *))$$.

In our experiments, we split each trail of mobility traces into two sets of same size: the training set, which is used to build the MMC, and the testing set, which is used to evaluate the accuracy of the predictor. Finally, we also compute the average predictability score for each user based on the MMC -learnt from his training dataset. Figure 4.2 shows the results obtained for a user from the Geolife dataset with $n$ ranging from 1 to 4. As expected, the accuracy first improves as $n$ increases but then seems to stabilize or even decrease slightly as soon as $n > 2$. Moreover, while unsurprisingly the prediction accuracy is usually better on the training set that on the testing set, this difference is not significative. This seems to indicate that the mobility behavior of an individual is similar in the second part of his trail of traces (the testing set) to the first part.
of the traces (the training set), which may not necessarily be the case if the mobility behavior of a user naturally drift due to an important change in his life. Finally, Figure 4.3 displays the results obtained for all users of the three different datasets. To summarize, the results consistently show that the accuracy and predictability are optimal (or almost optimal) when \( n = 2 \), with an accuracy and predictability ranging from 70\% to 95\%. As we can see in both metrics in Phonetic and Geolife datasets tend to follow the same form of curves described in Figure 4.2 having the best score when we use a second order MMC model and decreasing when \( n \) tends to increase. Nevertheless, for the synthetic dataset the score of both metrics increase as \( n \) becomes bigger because we are observe the same particular events. Up to this point, we

![Figure 4.2: Accuracy and predictability measured for a single user of Geolife.](image)

![Figure 4.3: Accuracy and predictability on the three datasets.](image)

have described the prediction of future locations. Another interesting inference, which gives more meaningful to the prediction, is the extraction of the semantics of POIs. This attack is presented in the next section. Nevertheless, if we add semantic labels to the prediction, then we increase the harm of the attack. The extraction of semantics of POIs is an inference attack itself and is presented in the next section.


Chapter 4. Next place prediction and semantic inference

4.2 Extraction of the semantics of POIs

This kind of inference is a complementary attack to the point of interest extractions. More detailed, the semantic extraction attack allows the adversary to have a meaningful knowledge about user point of interests in the MMC. Nevertheless, semantic extraction is not a trivial task and depends on the knowledge of the adversary about a geographic region. However, if adversary has not a complete or any knowledge about the meaning of found POIs, they can rely on heuristics in order to find the semantic of a place using the model structure or can use an external database such as Open Street Maps, Google maps, Yahoo maps, etc. as a source of semantic information. In the next subsection we present some works in the literature about semantic extraction based on heuristics and queries to external sources.

4.2.1 Related Work

In this subsection we present the three different sources to extract semantics for the discovered point of interest like: external semantic knowledge, heuristics and structure of the model. These sources are detailed in the next subsections.

Learning semantics through external knowledge

Nowadays, the availability of semantic information about locations has increased significantly with the advent of projects like *OpenStreetMap* (OSM) [RTC13], *Google Maps* [Goo12c] and *Yahoo Maps* [Yah12], which are a source of rich semantic. There exist projects in order to build formal geo-ontologies based on OSM like *LinkedGeoData* [Lin13] and *OSMonto* [OSM13], as well as to extract semantics of places [BBW12] in form of Resource Description Framework (RDF) graph [LS99] to make operations over the semantic of places. Furthermore, the work of Ballatore and Bertolotto [BB11b] aims to improve the location semantic inferred by performing web queries over rich semantic websites (e.g., Wikipedia). Cao et al. [CCJ10] propose a technique to extract the semantic of location places from GPS traces using external knowledge such as Google Maps and Yellow pages. More precisely, authors discover public common POIs from the mobility traces of 119 cars over 15 months, using a clustering algorithm (e.g. the *k*-means or OPTICS). Once POIs are discovered, the address of mobility traces contained in clusters are queried in order to verify if they share the same semantic. If this is not the case, clusters are split. Afterward clusters with the same semantic are merged using a similarity function based on the frequency of visits, average stay duration, average time of a day and the list of semantics that may apply to the cluster.

Extracting semantic through heuristics

Experience-based techniques can be used to infer semantic meaning from previous extracted points of interests. For instance, Gambs et al. [GKNndPC10a] propose a heuristic to infer homes of taxi drivers from the Yellow Cabs [PSDG09] dataset. Specifically, the first and last GPS traces of a given day are identified, based on the fact that taxi driver turn-on and turn-off GPS, when they leave and arrive home. Isaacman et al. [IBC+11] propose a method to identify home and work using POIs extracted from GSM traces. The authors sort GSM antennas based on the
4.2. Extraction of the semantics of POIs

number of days they were contacted. Then, the Hartigan’s leader clustering algorithm [Har75] is used to extract significant places. Afterward, logistic [DWHL04] regression is applied to determine if a POI corresponds to “home” or “work”, which computes the probability of a place to be important as function of the number of days, frequency and amount of events. Finally, the POIs with the highest probability during weekdays from 7PM to 7AM and from 1PM to 5PM are considered to be “home” and “work”, respectively. In order to verify the relevance of the results outputted, the authors use a Call Detail Record (CDR) dataset of 37 users over 60 days in different states of USA. Similarly, the work of Girardin et al. [GVGB09] quantifies attractiveness and popularity of POIs using density analysis over pre-defined POIs in order to establish a ranking of important places. The authors use call data records (CDR) from cities of Manhattan and Brooklyn over a year and geotagged publicly available photos on Flickr in New York to compute attractiveness and popularity. The former index is computed by using the Comparative Relative Strength (CRS), which is the ratio of the number of visits of a period divides by the number of visitors of the precedent period. The latter index is calculated using PlaceRank, inspired in the Google’s PageRank\(^9\) algorithms, which is a normal indicator to list the most visited pages.

Inferring semantic from the model

The POIs semantics can some times be inferred from the structure itself. For instance, in our work [GKNndPC11a] we propose a mobility model called mobility Markov chain, in which POIs are the states of the model and transitions are weighted with the probability to go from one state to another. Once POIs are sorted by density, the two densest places are likely to be “home” and “work”. Liao et al. [LFK07] design a method to extract semantics and actions from GPS mobility trace using conditional random field (CRF) [LMP01], represented by a graph of three layers (semantic places, activities and mobility traces). In more details, they take as input a trail of mobility traces, as well as features learned from activities and places. Then, they sample mobility traces by taking one representative point for each time windows of 10 minutes. Once the mobility traces have been segmented (first layer), the maximum a posteriori probability (MAP) is computed for assigning the activities and the feature vector in order to label activities and places. The former allows assigning activities (second layer) to a given segment of a trail of mobility trace while the latter links activities with places (third layer). For instance, a POI (first layer) can be connected to an activity, such as cooking (second layer) and binded to a location such as home (third layer). The method was tested using trails of mobility traces from 4 persons over 7 days. The drawback of this method is that feature vectors are not constructed automatically and thus their pertinence depends on the experts that have elaborated them. Ye et al. [YSL’11] propose a method to automatically annotate semantically of places using binary support vector machine (SVM) from users’ check-ins features. More precisely, the features for the SVM algorithm are extracted from implicit similarity (which is a similarity matrix of labeled and unlabeled places) and explicit patterns (e.g. coffee). The former is based on similarity of regularity of check-in. The latter is characterized by the distinct number of visits and time distribution of a given place. Features are obtained from generated check-ins of users in a

network and the binary SVM algorithm classifies unknown places with labels corresponding to categories extracted from Yelp\textsuperscript{10}. The researchers have tested their system with a dataset of 53432 places and 199 types of tags extracted from 5892 users of Whrrl\textsuperscript{11} over a month. The training set for this experiment was composed of user with more than 40 check-ins. While, the testing set is derived from the training set by randomly erasing 20% of labels. To evaluate the performance of the classification authors use the Hamming loss, which is the number of misclassified objects, one-error measures the number of times the first predicted label is correct, coverage is the average number of predicted labels before obtaining the correct one and average precision is the number of correct predicted labels [SS00, ZZ05]. The authors obtained as results an average precision of 80%.

4.2.2 Experimental results

Our implementation of the semantic extraction inference is twofold. On one hand, we rely on an heuristic called begin-end, in order to find home [GKNndPC10a]. The attack, performed over the San Francisco Yellow Cabs, consists on analyzing the first and last GPS traces in a given day, based on the fact that taxi driver turn-on and turn-off GPS, when they leave and arrive home. As we can see in the Figure 4.4, using this attack we were able to find the houses of some taxi driver. The draw back of the dataset is the absence of ground truth. Accordingly, we should verify manually the house of taxi drivers, in order to find evidence of the pertinence of the attack. In the bottom of the Figure 4.4 we found, using Google street view, a taxi parked in a impasse where we have found a taxi driver’s house. This piece of evidence as well as the logic behind the begin end attack demonstrate the attack capability. Nonetheless, the mobility Markov chain (MMC), introduced in Chapter 3, could be used to infer the home and work of the users. For the sake of explanation, we take the MMC in Figure 4.5, which is Bob mobility model. On one hand, we appreciate the state “2” is a central state from where Bob can reach almost all states. At the same time, from almost all other states it is possible to go to state

\textsuperscript{10} Social networking service [www.yelp.com]
\textsuperscript{11} This location based social networks does not exist anymore
4.3 Summary

In this chapter, we have presented an algorithm for next place prediction based on a mobility model of an individual that keeps track of the \( n \) previous locations visited. Experiments on

\[\text{From: 0:00 to 23:59:59 on ALL}\]

```
Huatanay(river)(0_A)
Home(1_A)
BusStop(2_A)
Saphi(river)(3_A)
BusStop(9_A)
Sauna(4_A)
Huatanay(5_A)
Courthouse(6_A)
Work(School)(7_A)
```

Figure 4.5: Simple MMC with semantic of a user in ARUM dataset.

"2". As consequence we can infer that this state is home. On the other hand, we take the two biggest values of the stationary vector, \( \pi = [0.02, 0.01, 0.48, 0.02, 0.02, 0.02, 0.01, 0.37, 0.05] \), of POIs and then these two places are mostly home and work. In this case POI “2” is home and POI “7” work. The last resort to discover the meaning of the other states could be by querying an external data source, such as the ontology of Open Street Map (OSMonto\(^\text{12}\)) More precisely, to obtain the semantic of a POI represented by its centroid, which is a latitude and longitude coordinate, we use the Cloudmade API\(^\text{13}\) to query OSMonto in order to obtain the semantic, as it is illustrated in the MMC of the Figure 4.5. In the next section we summarize the current chapter.

4.3 Summary

In this chapter, we have presented an algorithm for next place prediction based on a mobility model of an individual that keeps track of the \( n \) previous locations visited. Experiments on

\(^{12}\) wiki.openstreetmap.org/wiki/OSMonto
\(^{13}\) cloudmade.com
three different datasets show that the accuracy of this prediction algorithm ranges from 70% to 95%. Moreover, while the accuracy of the prediction grows with $n$, choosing $n > 2$ does not seem to bring an important improvement at the cost of a significant overhead in terms of computation and space for the learning and storing of the mobility model. In order to further improve the accuracy of the prediction, we have noticed that the train and test sets should be as homogeneous as possible in order to build a good model. For instance, if we construct a model based on the vacation period, the predictor issued from this dataset will not be able to forecast the next place in scholar or work period. Another limitation is the cluster selection parameters, which are important to have enough significant places. In the last part, we have shown the method to derive semantic from the model, as well as the external source to obtain POI semantic. One of the drawbacks is the absence of a dataset with the semantic labels for users’ POIs to verify how accurate the method is. The same problem was stated in a semantic trajectories survey done by Parent et al. [PSR’13], where they mention also the lack of ground truth to validate results. Predicting future whereabouts and extracting semantics from POIs could be worst if the adversary is able to know the identity of the victim. Thus, in the next chapter another attack with the aim to reverse anonymity from datasets is presented.
Chapter 5

De-anonymization and linking attack

"Once you have lost your privacy, you realize you have lost an extremely valuable thing."

— Billy Graham.

In this chapter, we focus on a particular form of inference attack called the de-anonymization attack, by which an adversary tries to infer the identity of a particular individual behind a set of mobility traces. More precisely, we suppose that the adversary has been able to observe the movements of some individuals during a non-negligible amount of time (e.g., several days or weeks) in the past during the training phase. Later, the adversary accesses a different geolocated dataset containing the mobility traces of some of the individuals observed during the training phase, plus possibly some unknown persons. Then, the objective of the adversary is to de-anonymize this dataset (called the testing dataset) by linking it to the corresponding individuals contained in the training dataset. Note that simply replacing the real names of individuals by pseudonyms before releasing a dataset is usually not sufficient to preserve the anonymity of their identities because the mobility traces themselves contain information that can be uniquely linked back to an individual. In addition, while a dataset can be sanitized before being released by adding spatial and temporal noise, the risk of re-identification through de-anonymization attack nevertheless still exists. Thus, in order to be able to assess the importance of this risk, it is important to develop a method to quantify it. In this chapter, we propose a novel method to de-anonymize location data based on a mobility model called Mobility Markov Chain (MMC) [GKNndPC11a]. A MMC (cf. Chapter 3) is a probabilistic automaton, in which each state corresponds to one (or possibly several) Point Of Interest (POIs), characterizing the mobility of an individual and an edge indicates a probabilistic transition between two states (i.e., POIs). Each state can have a semantic label attached to it such as “home”, “work”, “leisure”, “sport”, ... A MMC is built out of the mobility traces observed during the training phase and is used to perform the de-anonymization attack during the testing phase. More precisely, the mobility of each individual both from the training and testing sets is represented in the form of a MMC. Afterwards, a distance is computed between possible pairs of MMCs from the training and testing sets in order to identify the closest individuals in terms of mobility. In short, the gist of this method is that the mobility of an individual can act as a signature, thus playing the role
of a quasi-identifier [BWj05]. Thus, if the adversary knows a target user Alice and a signature of her mobility (e.g., he has learnt her MMC out of the training set), he can try to identify her by finding a matching signature in the testing set. The outline of the paper is the following. First, in Section 5.1, we review some related work on de-anonymization attacks and mobility models. Afterward in Section 5.2, we present the distance metrics between MMCs that we designed in order to quantify the closeness between two mobility behaviors, while in Section 5.3 we describe how to build predictors (which we call de-anonymizers) based on these distances to efficiently and accurately de-anonymize location data. Finally, we evaluate experimentally the efficiency of the proposed attack on real geolocated datasets in Section 5.4 before concluding in Section 5.5.

5.1 Related work

An inference attack corresponds to a process by which an adversary that has access to some data related to individuals (and potentially some auxiliary information) tries to deduce new personal information that was not explicitly present in the original data. For instance, a famous inference attack was conducted by Narayanan and Shmatikov on the “Netflix dataset” [NS08]. This dataset is a sparse high dimensional data containing ratings on movies of more than 500 000 subscribers from Netflix that was supposed to have been anonymized before its release for the Netflix competition\(^\text{14}\). However, Narayanan and Shmatikov have performed a de-anonymization attack that was able to successfully re-identify more than 80% of the Netflix subscribers by using the Internet Movie Database\(^\text{15}\) (IMDB), a public database of movie ratings, as auxiliary knowledge.

Inference attacks have also been developed specifically for the geolocated context. For instance, Mulder et al. [DMDBP08] have proposed two methods for profiling users in a GSM network that can also be used to perform a de-anonymization attack. The first method is based on constructing a Markov model of the mobility behavior of an individual while the second considered only the sequence of cell IDs visited. Once POIs have been extracted for each user, an agglomerative hierarchical clustering algorithm is used to group users according to a similarity measure called the cosine similarity [TSK06]. Their first method is relatively similar to ours with two major differences due to the fact that their dataset comes from a cellular network. Thus, it relies on the static GSM cells as the states of the Markov model, while we dynamically learn the POIs from the mobility traces of an individual. Therefore, in our setting two individuals do not necessarily have any POI in common, whereas with GSM cells, individuals living in the same area have a high probability of sharing some POIs (i.e., cells). As a consequence, the second main difference between their work and ours is that the transitions are only possible between neighboring GSM cells, as it is impossible to “jump” from one cell to another if they are not adjacent. Their attack was validated through experimentations using cell locations of 100 users from a MIT Media Lab dataset\(^\text{16}\) [ESP06] that includes information such as call logs, Bluetooth devices in proximity, cell tower IDs, application usage and phone status. During the

\(^{14}\) http://www.netflixprize.com

\(^{15}\) http://www.imdb.com

\(^{16}\) http://reality.media.mit.edu/dataset.php
experiments, the authors have observed that if the currently profiled user belongs to a cluster of other similar users, there is a high chance of making a mistake about the identity inferred among all the users of this cluster when performing the de-anonymization attack. The success rate of the re-identification attack varies from 37% to 39% using the Markov model, against 77% to 88% when the sequence of cells visited is used.

Zang and Bolot [ZB11] have performed a study of the top $n$ most frequently visited places by an individual in a GSM network and show how they can act as quasi-identifiers to re-identify anonymous users. Their study was performed on the Call Data Records (CDRs) from a nationwide US cellular provider collected over a month and contains approximately 20 millions users. From this dataset, the authors have identified the top $n$ most frequent places for each user at different levels of spatial (i.e., sector, cell, zip code, city, state and country) and time granularity (i.e., day and month). Their inference attack was able to successfully re-identify 35% of the population studied when the adversary has no auxiliary knowledge and even up to 50% when the adversary can use the knowledge of the social network of users as auxiliary information. The social network was constructed by creating a social relationship between two individuals that have called each other at least once in the past. In their analysis, the authors emphasize that the distance between home and work can be an indicator of the privacy level for an individual. In particular, the larger this distance, the higher the risk that this individual can be de-anonymized.

Ma et al. [MYYR10] proposed an inference attack to de-anonymize users in a geolocated dataset along with a metric to quantify the privacy loss of an individual. Two datasets were used in this study, one recording the movements of San Francisco YellowCabs [PSDG09] and the other related to the movements of Shanghai city public buses\footnote{To the best of our knowledge, this dataset is not publicly available contrary to the other one.}. Two types of adversary models were considered: the passive one, collecting the whereabouts of individuals from a public source (possibly sanitized) and the active one that can deliberately participate or perturb the data collection in order to gain additional knowledge about the location of some specific individuals. To retrieve the identities of individuals, the authors imagine four different estimators that the adversary can use to measure the similarity between mobility traces (for instance between the original traces and the sanitized ones). Namely, the Maximum Likelihood Estimator relies on the Euclidean distance to compute the similarity between mobility traces. The Minimum Square Approach computes the negative sum of the square of the absolute value of the difference between the original traces and the sanitized ones. The Basic Approach assumes that the traces have been perturbed by uniform noise. Thus with this approach, a mobility trace will be considered to be similar to another trace if it is contained within a sphere of radius $r$ centered on the original trace. Finally, the Weighted Exponential Approach is similar to the Basic Approach, with the exception that no assumption is made on the type of noise generated. These methods approach a success rate of de-anonymization of 80% to 90% on the San Francisco YellowCabs dataset and between 60% and 70% on the Shanghai bus dataset, and this even when the data is sanitized through the addition of spatial noise. However, contrary to our work, these inference attacks were conducted on the whole dataset (there was no split between a training set and a testing set). In particular, the authors assume that both type of adversaries (i.e., passive and active ones) pick the information they need to build the mobility model from
Chapter 5. De-anonymization and linking attack

the same dataset on which the success of the de-anonymization attack is tested. This induces an overly strong bias in the re-identification results obtained with this approach.

The work of Freudiger, Shokri and Hubaux [FSH12] also focuses on re-identifying users of geolocated datasets. These experiments have been conducted on two datasets. The first dataset contains the GPS traces of 24 users from the city of Borlange recorded in a two-year period (1999-2001) while the second one is due to Nokia and is composed of the GPS traces of 150 users from the city of Lausanne recorded over a year. In this work, the pair of POIs “home/work” is used as pseudo-identifiers to de-anonymize users. First, a variant of the k-means algorithm is used to extract POIs from the mobility traces. Then, the POI in which the individual considered stays the most often between 9PM to 9AM is identified as “home” while the POI in which the individual stays the most often between 9AM to 5PM is labelled as “work”. The training phase consists in applying this method on the raw data to extract the pairs “home/work” for all individuals, and then to conduct the same attack on sampled traces in order to assess how much the pair “home/work” can still be inferred even when the dataset released has been sanitized by applying sampling. The success of this method depends on how sampled traces have been generated. Indeed, the authors have proposed several sampling schemes whose bias towards selecting home/work locations or other POIs can be parametrized. The authors have shown that when 100 samples are observed, the de-anonymization rate is approximately 67% for the Borlange dataset and 70% for the Nokia one. Unfortunately, as most of the previous works presented, this study does not split the available data into a training and testing set during its evaluation but rather generates the samples that will constitute the testing set directly from the training one. This introduces a major bias in the evaluation of the techniques, which we further discuss in Section 5.4.4.

The work of Xiao et al. [XZLX10] relies on the notion of Semantic Location Histories (SLH) to compute the similarity between users. In a nutshell, a SLH is simply the sequence of semantic locations frequently visited by an individual. Like several previous approaches, this work first uses a hierarchical clustering algorithm to extract POIs out of mobility traces. Then, using as external knowledge a database that can associate semantics to a location, each POI is linked with a semantic tag for each level of the hierarchy (e.g., “Italian restaurant” and then “restaurant” at an upper level). Finally, the SLH is computed by analyzing the sequence of POIs visited by a user and taking into account their semantic labels. The similarity measure designed by the authors is based on the notion of maximal travel match, which counts the number of similar semantic locations visited (not necessarily in the same order) by two different SLHs within a predefined time interval. This metric is computed for each layer of the cluster hierarchy before being summed over all possible layers, possibly by weighting the influence of a particular level (i.e., the deeper the level, the bigger the influence). Finally, the proposed approach was evaluated on the Geolife dataset [ZLC+08]. Contrary to previous work, the success of the de-anonymization attack is quantified in terms of the normalized discounted cumulative gain, a metric originated from information retrieval [JK02]. In a nutshell, the objective of this metric is to rank all the possible candidates to de-anonymization with respect to how close their mobility is to the behavior of the user considered. In the experiments conducted, the success of

---

18 http://icapeople.epfl.ch/freudiger/borlange.zip
19 To the best of your knowledge, this dataset is not publicly available.
the attack as measured by this metric was between 0.7 and 0.9. Basically the closer this value is to 1, the more effective the attack is. Note that, due to the different metric that was used, this method is not directly comparable to other previous works.

Shokri et al. [STLBH11] inferred the correspondence between pseudonymized traces of 40 randomly chosen users of the YellowCabs dataset [PSDG09], in which each position is a cell of 8×5 grid over the San Francisco Bay area, and user profiles represented in the form of hidden Markov model. Their attack computes a matching probability between pseudonymized traces and user profiles by using the classical forward-backward algorithm [RN09]. This matching probability basically represents the likelihood that a particular set of traces correspond to a specific user. Once these probabilities are computed, a bipartite graph of traces and users is formed whose edges are weighted according to the associated matching probability between pairs of user traces. This bipartite graph is then given as input to the Hungarian algorithm that assigns a particular set of traces to each user. As the paper mainly focuses on solving the task of “individual tracking”, which corresponds to being able to predict at which location a user was located at a particular moment in time, their results are not directly comparable to our work. However, we believe that the framework they have developed is generic enough in terms of the attacks it encompasses that it could also model the de-anonymization attack.

The study of Srivatsa and Hicks [SH12] de-anonymizes users based on the contact graph, generated from mobility traces and information about their social network. The authors use three different datasets: the Saint Andrews dataset, which contains 18 241 contact traces from 27 different users issued from WiFi access points, the Smallblue dataset is formed by 240 665 contact traces of 125 users representing messages generated by instant messaging, and finally the Infocom 06 dataset that consists of 182 951 contact traces from 78 users extracted from Bluetooth devices used during the Infocom 2006 conference. In addition, the authors have used some auxiliary knowledge in the form of social networks extracted from Facebook for the Saint Andrews and Smallblue datasets and from DBLP for the Infocom 06 dataset. Their attack relies on the structural similarities between contact and social graph, which is measured in terms of standard graph similarity measures such as the graph edit distance (i.e., minimal number of edges to erase or add to have a perfect match), the maximum common subgraph (i.e., the number of vertices in the largest common subgraph) and the distribution of node degrees. The de-anonymization attack works in two steps. During the first step, the n most important POIs are extracted from both contact and social graphs based on the node centrality metric [CLP06]. The second step is the de-anonymization process itself that links the identities of users from the social graph with the corresponding users of the contact graph. The authors have used three different mapping techniques: distance vector (82% of de-anonymization rate), spanning trees (88% of de-anonymization rate) and subgraph matching (80% of de-anonymization rate).

Sharad and Danezis [SD13] have de-anonymized the communication graph datasets of the D4D challenge20. Their attack exploits the topology of the contact graph to break the anonymity by looking for subgraphs that are isomorphic and form the largest subgraphs. More precisely, the authors compute the 1-hop node neighborhood degree distribution for each node of the subgraphs generated from the graph. This distribution is used as a similarity metric to identify the closest node with the same characteristics in the other graph. Afterwards, for matching the

20 http://www.d4d.orange.com
Chapter 5. De-anonymization and linking attack

2-hop nodes, the authors rely on two metrics: the neighbor match, which is the number of common 1-hop nodes they are associated to, and the signature match, which is a metric computed from the in and out degree distributions. To evaluate their methodology, Sharad and Danezis have used the EU mail communication dataset [LKF07] and artificially generated a dataset with similar characteristics to the communication graphs of the D4D challenge obtaining between 56% to 96% of re-identification rate.

A recent study of De Montjoye and co-authors [dMHVB13] has shown that if an individual has a unique pattern in the anonymized dataset then this is enough to characterize him even if a particular dataset does not contain personal information such as name, age and address. The objective of this study was precisely to measure the uniqueness of 1.5 millions users in an anonymized mobile phone dataset collected from April 2006 to June 2007. By taking 2 and 4 random locations to characterize the mobility patterns of individuals, the authors were able to identify respectively 50% and 95% of the users in the anonymized dataset. However, this study has some limitations as for instance it is not clear how the knowledge that is learnt can be used to later in the future de-anonymize an individual. More precisely, the study shows that during a particular period (e.g., a day or a week) if we consider the sequence of a constant number locations visited by an individual compared to the rest of the population, this sequence is likely to be unique. Nonetheless, this do not preclude the possibility that other individuals might visit the same sequence of locations during another period (e.g., the next day or the next week). Thus, it is not obvious how to build an efficient de-anonymizer from the results of this study. Moreover as for other previous works, there is no separation being made between the training and the testing sets in this study.

Finally, a recent work by Unnikrishnan and Naini [UN13] shows that revealing statistics about the movements of users (instead of directly their mobility traces) can also be used to de-anonymize efficiently geolocated datasets and thus may also be harmful to privacy. To perform the de-anonymization attack, the authors computed through the Hungarian algorithm the minimum weight matching [Gut89] in a bipartite graph, composed of an anonymous and labeled graph, in which nodes represent users and the weights on edges correspond to the distance in term of entropy between the spatial and temporal distribution of nodes across different places. The dataset used is a two weeks logs obtained from the access points from the EPFL campus, using the first week for training and the second one for testing. Using this technique, the authors obtained 70.5% of success for the de-anonymization.

In this section, we have reviewed the previous work on de-anonymization attacks in the geolocated context, which we summarize in Table 5.1. Note that the results from some of these studies cannot be directly compared due to the difference in nature of the datasets used, the way the success metrics are defined for an attack and also the fact that in most of these studies the data has not been split in proper training and testing sets. We discussed further this fundamental issue in Section 5.4.4. In the following sections, we will present our approach to de-anonymization. More precisely, we first introduce the distances metrics to measure the similarity between two MMCs, before describing the de-anonymizers for linking the users using the distances we propose. Finally, we demonstrate experimentally how to use these distance metrics to perform a de-anonymization attack.
5.2 Distances between mobility Markov chains

In this section, we propose four different distances quantifying the similarity between two mobility Markov chains. These distances are based on different characteristics of the MMCs and thus give different but complementary results. We will rely on these distances in the following sections to perform the de-anonymization attack.

5.2.1 Stationary distance

The intuition behind the stationary distance is that the distance between two MMCs can be evaluated as to the sum of the distances between the closest states of both MMCs. In order to compute the stationary distance, the states of the MMCs are paired in order to minimize this distance. As a result, it is possible that a state from the first MMC is paired with several states of the second MMC (this is especially true if the MMCs are of different size). Furthermore, the computation of the stationary distance heavily relies on the stationary vectors of the MMCs. In a nutshell, the stationary vector of a MMC is a column vector \( V \), obtained by multiplying repeatedly a vector initialized uniformly \( V_{\text{init}} \) by the MMC transition matrix \( M \) until convergence (i.e., until the distribution of values in this vector reaches the stationary distribution of the MMC).

The stationary distance is directly computed from the stationary vectors of two MMCs (hence its name). More precisely, given two MMCs, \( M_1 \) and \( M_2 \), the stationary vectors, respectively \( V_1 \) and \( V_2 \), of each model are computed. Afterwards, Algorithm 21 is run on these two stationary vectors. For each state in \( V_1 \), the algorithm searches for the closest state in \( V_2 \) (lines 5 to 11, Algorithm 21) and then multiplies the distance between these two states by the corresponding probability of the stationary vector of the state of \( V_1 \) currently considered (line 12, Algorithm 21).

Once the algorithm has taken into account all states from \( V_1 \), the current value computed represents the distance from \( M_1 \) to \( M_2 \) (distance \( AB \) in line 1, Algorithm 22). This distance is not symmetric as such and therefore in order to symmetrize it, Algorithm 21 is called once again,
but on $V_2$ and $V_1$ in order to obtain the distance from $M_2$ to $M_1$ ($\text{distance}_{BA}$ of line 2, Algorithm 22). The result is made symmetrical by computing the average of these two distances (line 3, Algorithm 22).

**Algorithm 21** Stationary distance($V_1, V_2$)

1. $\text{distance} = 0$
2. for $i = 1$ to $n_1$ (the number of nodes in $V_1$) do
3. $\text{MinDistance} = 100000$ kilometers
4. Let $p_i$ be the $i^{th}$ node of $V_1$
5. for $j = 1$ to $n_2$ (the number of nodes in $V_2$) do
6. Let $p_j$ be the $j^{th}$ node of $V_2$
7. $\text{CurrentDistance} = \text{Euclidean\_Distance}(p_i, p_j)$
8. if ($\text{CurrentDistance} < \text{MinDistance}$) then
9. $\text{MinDistance} = \text{CurrentDistance}$
10. end if
11. end for
12. $\text{distance} = \text{distance} + \text{Prob}_{V_1}(p_i) \times \text{MinDistance}$
13. end for
14. return $\text{distance}$

**Algorithm 22** Symmetric stationary distance($V_1, V_2$)

1. $\text{distance}_{AB} = \text{Stationary\_distance}(V_1, V_2)$
2. $\text{distance}_{BA} = \text{Stationary\_distance}(V_2, V_1)$
3. $\text{distance} = (\text{distance}_{AB} + \text{distance}_{BA})/2$
4. return $\text{distance}$

5.2.2 Proximity distance

The intuition behind the proximity distance is that two MMCs can be considered close if they share “important” states. For instance, if two individuals share both their home and place of work they should be considered as being highly similar. Moreover, the importance of a state is directly proportional to the frequency at which it is visited. Therefore, the first states ordered by decreasing order of importance are compared, then the second ones, then the third ones, and so on. The proximity score obtained for sharing the first states is considered twice as important as the score for sharing the second states, which is itself twice as important as the sharing of the third states, and so forth.

Given two MMC models $M_1$ and $M_2$ (ordered in a decreasing manner with respect to their stationary probabilities), this distance is parametrized by a threshold $\Delta$ and a rank. The objective of the rank is to quantify the importance of matching two states at a specific level. In particular, the higher is the value of the rank, the bigger is the weight that will be given to these POIs. For instance for the first pair of POIs, we have set rank $= 10$.

Algorithm 23 starts by verifying for each pair of nodes between $M_1$ and $M_2$ if the Euclidean distance between them is less than the threshold $\Delta$ (line 8, Algorithm 23). If this condition is
met, the value of \( rank \) is added to the score value (line 9, Algorithm 23). Afterwards, \( rank \) is divided by two (lines 11-14, Algorithm 23). Once all the pair of nodes have been processed, the global distance is set to be the inverse of the global score if this score is non-null (lines 17-19, Algorithm 23). Otherwise, the distance outputted is set to a large value (e.g., 100 000 kilometers).

**Algorithm 23** Proximity\_distance\((V_1, V_2, \Delta, rank)\)

1: Sort the states of \( V_1 \) by decreasing order of frequency
2: Sort the states of \( V_2 \) by decreasing order of frequency
3: \( score = 0 \)
4: \textbf{for} \( i = 1 \) to \( \min(n_1, n_2) \) \textbf{do}
5: \hspace{1em} Let \( p_a \) be the \( i^{th} \) node of \( V_1 \)
6: \hspace{1em} Let \( p_b \) be the \( i^{th} \) node of \( V_2 \)
7: \hspace{1em} \( distance = \text{Euclidean\_distance}(p_a, p_b) \)
8: \hspace{1em} \textbf{if} \ (distance < \Delta) \ \textbf{then}
9: \hspace{2em} \( score = score + rank \)
10: \hspace{1em} \textbf{end if}
11: \hspace{1em} \( rank = rank/2 \)
12: \hspace{1em} \textbf{if} \ (rank = 0) \ \textbf{then}
13: \hspace{2em} \( rank = 1 \)
14: \hspace{1em} \textbf{end if}
15: \hspace{1em} \textbf{end for}
16: \( distance = 100000 \)
17: \textbf{if} \ (score > 0) \ \textbf{then}
18: \hspace{1em} \( distance = 1/score \)
19: \textbf{end if}
20: \textbf{return} \( distance \)

The stationary distance is composed of the sum of the Euclidian distances of some pairing of the states while the proximity distance is completely different as it is based on the semantics behind the MMCs. Indeed, the first state in the model is inferred as being very representative of the mobility of an individual (e.g., home), the second as quite representative (e.g., the place of work), and two individuals are considered as very similar if they share these two places. Thereafter, we will see how these distance can be used to build de-anonymizers and how their diversity can be leveraged and combined to enhance the success of the de-anonymization attack.

### 5.2.3 Matching distance

The matching distance is similar to the stationary distance in the sense that this distance also corresponds to the sum of distances between the states of these two MMCs. However, the pairing of the states between the MMCs is done in a different way as each state from the first MMC is paired with one and only one state of the second MMC.

The matching distance is based on the Hungarian method [TP04], which is a polynomial-
Chapter 5. De-anonymization and linking attack

Algorithm 24 Matching_distance($V_1, V_2$)

1: Let $D$ be the distance matrix $D$ and $Min$ the minimization matrix
2: Let $n_1$ be the number of nodes in $V_1$
3: Let $n_2$ be the number of nodes in $V_2$ (we suppose that $V_2$ is the stationary vector with the
   fewest number of states if the number of states between $V_1$ and $V_2$ is different)
4: Add fake states to $V_2$ if necessary to ensure that $n_2 = n_1$
5: for $i = 1$ to $n_1$ do
6:   for $j = 1$ to $n_2$ do
7:     $D_{i,j} = \text{Euclidean} \_\text{distance}(p_i, p_j)$
8:     if ($p_i$ or $p_j$ is fake) then
9:        $Min_{i,j} = 100000$
10:      else
11:         $Min_{i,j} = D_{i,j}$
12:     end if
13:   end for
14: end for
15: $Index = \text{HungarianAssignment}(Min)$
16: $Dist = 0$
17: for $i = 0$ to $n_1$ do
18:   $row = Index_{i,0}$
19:   $col = Index_{i,1}$
20:   if ($V_2.node(col)$ is not fake) then
21:      $proba = (V_1(row) + V_2(col))/2$
22:      $Dist = Dist + D_{row,col} \times proba$
23:   else
24:      $proba = (V_1.stationaryVector(row) + 0)/2$
25:      $Dist = Dist + \text{NearestState}(V_1.node(row), V_2.nodes)$
26:   end if
27: end for
28: return $Dist$
5.2. Distances between mobility Markov chains

time combinatorial optimization algorithm for assignment problems. More precisely, the method works as follows. Given two MMCs, $M_1$ and $M_2$ and their corresponding stationary vectors $V_1$ and $V_2$, Algorithm 24 first verifies if the number of nodes in both models is the same. If not, we assume without loss of generality that $M_2$ is the MMC with the fewer number of states. Before continuing the computation of the distance, “dummy” states are added to $M_2$. Each dummy state is a copy of the centroid of the states of $M_2$ (line 4, Algorithm 24). Once this process is completed, the number of states in $V_1$ and $V_2$ is the same.

Afterwards, the distance matrix $(D_{i,j})$ and the minimization matrix $(Min_{i,j})$ are computed. The distance matrix contains the Euclidean distance between each pair of nodes in $M_1$ and $M_2$ (lines 5-14, Algorithm 24), in which nodes in $M_2$ form the rows of the matrix and nodes in $M_1$ correspond to the columns of $D_{i,j}$. For instance, the distance $D_{i,j}$ is the Euclidean distance between the node $i$ in $M_2$ and the node $j$ in $M_1$. The minimization matrix is equivalent to the distance matrix, except that each distance $Min_{i,j}$ in which node $i$ is a dummy is assigned a large default value, such as 100 000 kilometers (line 9, Algorithm 24). In short, the objective of this large value is to guide the Hungarian method towards giving priority to real states instead of dummy ones.

The minimization matrix, which is squared, is passed as input to the Hungarian method. The Hungarian method then computes the optimal assignment between states from $M_1$ and $M_2$ that minimizes the global sum of distances between pair of states and such that each state from $V_1$ is exactly matched with one state of $V_2$ (and vice-versa). This optimal matching is returned as a matrix $Index$ composed of two columns. Each column contains respectively the indexes of rows and columns of the optimal assignment for the minimization matrix $Min_{i,j}$. Figure 5.1 illustrates graphically an optimal assignment between two MMCs (one for Alice and one for Bob). The matrix $Index$ corresponding to Figure 5.1 is $I = \{ \{1, 1\}, \{2, 3\}, \{3, 2\}, \{4, 3\}, \{5, 4\} \}$.

![Alice MMC.](image1.png) ![Bob MMC.](image2.png)

Figure 5.1: Example of the optimal matching of two MMCs. The number in each node corresponds to the numbers of traces that falls inside this state.

For each pair of the matching obtained, the distance between these two (non-dummy) states is multiplied by the average of probabilities of stationary vectors $V_1$ and $V_2$. However, when one of the state in $M_2$ is “dummy”, the nearest real state in $M_2$ is identified, and the distance
between these two states is then multiplied by the probability corresponding to the “orphan” state in $V_1$ divided by two (lines 17–27, Algorithm 24). The distance returned is the sum of the values computed for each pair of the matching (line 28, Algorithm 24).

### 5.2.4 Density-based distance

Similarly to the stationary and the matching distances, the density-based distance is simply the sum of the distances between pairs of MMCs states. However, the MMCs states are paired according to their rank once they are sorted using their corresponding probabilities in the stationary vector.

First, given two MMC models $M_1$ and $M_2$, the nodes in both models are sorted by decreasing density (lines 1 and 2, Algorithm 25). Therefore, the first node will be the one with the highest stationary probability. Afterwards, the first node of $M_1$ is matched with the first node of $M_2$ and the same goes for the rest of the nodes. Finally, the sum of all distances between matched nodes is computed (line 4 to 8, Algorithm 25) and the algorithm outputs the total distance.

**Algorithm 25** Density-based_distance($V_1, V_2$)

1. Sort the states of $V_1$ by decreasing order of density
2. Sort the states of $V_2$ by decreasing order of density
3. $distance = 0$
4. for $i = 1$ to $\min(n_1, n_2)$ do
5. Let $p_a$ be the $i^{th}$ node of $V_1$
6. Let $p_b$ be the $i^{th}$ node of $V_2$
7. $distance = distance + Euclidean\_distance(p_a, p_b)$
8. end for
9. return $distance$

The stationary distance, the matching distance and the density-based distance are all numerical, in the sense that they are formed of the sum of the Euclidian distance between some pairing of states. In contrast, the proximity distance is different as it is based on the semantics behind the MMCs. Indeed, the first state in the model is inferred as being very representative of the mobility of an individual (e.g., home), the second as quite representative (e.g., the place of work), and two individuals are considered as very similar if they share these two places. Table 5.2 summarizes the main characteristics of these distances. Thereafter, we will see how these distance can be used to build de-anonymizers and how their diversity can be exploited and combined to enhance the success of the de-anonymization attack.

### 5.3 De-anonymizers

In this section, we rely on the four distances proposed in the previous section to build statistical predictors in order to perform a de-anonymization attack. We call such a predictor, a de-anonymizer in reference to its main objective. A de-anonymizer takes as input the MMC representing the mobility of an individual and tries to identifies within a set of anonymous
5.3. De-anonymizers

<table>
<thead>
<tr>
<th>Name</th>
<th>Strategy</th>
<th>Type</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary distance</td>
<td>Stationary vector</td>
<td>Numerical</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Proximity distance</td>
<td>Proximity of MMCs states</td>
<td>Relative</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Matching distance</td>
<td>Hungarian algorithm</td>
<td>Numerical</td>
<td>(O(n^3))</td>
</tr>
<tr>
<td>Density-based distance</td>
<td>Weight of MMC states</td>
<td>Numerical</td>
<td>(O(n))</td>
</tr>
</tbody>
</table>

Table 5.2: Summary of MMC distances. In this table, the complexity is measured with respect to \(n\) the number of individuals in the datasets that has to be de-anonymized (for simplicity the training and testing datasets are assumed to contained the same number of individuals).

MMCs, the one that is the most similar (i.e, the closest in terms of distance). For example, a de-anonymizer may learn from the training set a MMC representing the mobility of Alice and later look for the presence of Alice in the testing set. A de-anonymizer can be based on one distance or a combination of them.

- **The minimal distance** de-anonymizer considers that in each row, the MMC with the minimal distance (i.e., the column) is the individual corresponding to the identity of the row. We have considered four instantiations of this de-anonymizer, one for each distance described previously (i.e., stationary distance, proximity distance, matching distance and density-based distance).

- **The maximal-gap** de-anonymizer takes as input several distance matrices corresponding to different distance metrics. For each distance matrix, the minimal-distance anonymizer outputs two predictions (instead of one) for each row, which corresponds to the first and second smallest values of the distances in this row. Afterwards, the gap (i.e., difference) between these two values are computed. The higher the gap, the more confident we can be in the prediction made by the de-anonymizer on this particular distance matrix. Therefore, the distance metric with the highest gap among all the ones considered will be the candidate considered for the de-anonymization.

- **The simple vote** de-anonymizer receives as input a list of candidates, which corresponds to the identities of the \(n\) minimal values of a particular row in at least two different distance matrices. The candidate that receives the highest number of votes will be the one considered for the de-anonymization. For example, in Table 5.3, each column corresponds to different distance metrics while each row contains the proposed candidates at this position. In the first row, Bob gets two votes against one for Alice and therefore he is considered as the most likely candidate for de-anonymization.

<table>
<thead>
<tr>
<th>points</th>
<th>(m_1)</th>
<th>(m_2)</th>
<th>(m_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Bob</td>
<td>Alice</td>
<td>Bob</td>
</tr>
<tr>
<td>2</td>
<td>Charlie</td>
<td>Bob</td>
<td>Alice</td>
</tr>
<tr>
<td>1</td>
<td>Alice</td>
<td>Charlie</td>
<td>Charlie</td>
</tr>
</tbody>
</table>

Table 5.3: Illustration of the voting method.
Chapter 5. De-anonymization and linking attack

- The weighted vote de-anonymizer, much like the simple vote one, takes as input a list of candidates coming from different distance matrices. However, the voting method weights each possible candidate depending on the rank he has obtained for the different distance metrics. For instance, if each distance metric proposes $n$ candidates, the first one is given a weight $n$, while the second receives a weight of $n - 1$, and so on. For example, in Table 5.3, the first column contains the weights for each candidate. The outcome of the weighted voting method is that Bob gets 8 votes, Alice 6 and Charlie 4. Therefore, Bob is considered as the “winner” (i.e., most likely candidate) for the de-anonymization.

- The stat-prox de-anonymizer behaves exactly like the minimal stationary distance de-anonymizer, except when the stationary distance is above a given threshold and the proximity distance is below its maximum value (i.e., 100 000 kilometers). The intuition is that if the minimal stationary distance is very small, we should use it. Otherwise, we rely on the minimal proximity distance unless it gives no conclusive result, in which case we roll back to the minimal stationary distance.

5.4 Experiments

An overview of the process of de-anonymization attack over geolocated datasets used in our experiments is illustrated in Figure 5.2. Considering a particular geolocated dataset, we first sort the mobility traces of each user in a chronological order. Then, for each user his trail of
mobility traces is split into two disjoint trail of traces of same size, one for the training set and one for the testing set as explained in Section 5.4.1. The former is part of the auxiliary knowledge gathered by the adversary while the latter is the ground truth we use to assess the success of the attack. For each of this dataset, we learn a MMC for each trail of mobility traces. With respect to the MMCs learnt from the training set, the adversary knows the correspondence between these models and the corresponding identities of the users. Afterwards, the distances described in Section 5.2 are used to compute a distance matrix between the MMC models resulting from the training and testing sets. Subsequently, using this distance matrix as input to one of the de-anonymizer described in Section 5.3, the objective of the de-anonymization attack is to map back the users of the testing set to their true identities by linking their models in the testing set to the corresponding ones in the training set. Finally, the success rate of the attack is computed by measuring the ratio between the number of correct predictions over the total number of guesses.

In order to estimate the robustness of the de-anonymization attack when the sampling conditions change, we have also performed some downsampling on the data. In a nutshell, a sampling mechanism summarizes several mobility traces into fewer traces. This is generally done by representing a set of subsequent traces that have occurred within the same time window into a single trace (e.g., the average or median trace).

We evaluate the efficiency of the de-anonymization attack introduced in the previous section on five different datasets described in Section 5.4.1. Then, we describe the method used to fine-tune the parameters of the clustering algorithms in Section 5.4.2. Afterwards, we report the results of those experiments that were conducted by using the distances and de-anonymizers described in the previous sections. More precisely, we evaluate the accuracy of the de-anonymization attack relying on either a single predictor or a combination of them (Section 5.4.3). Finally, in Section 5.4.4, we compare our work with the performance reported in related works, highlighting in particular the bias in some of the experimental settings of these previous works.

5.4.1 Analysis of the characteristics of the datasets

In this subsection we present the datasets used for the experimentation for analyzing. In more detail, we describe each dataset, we analyze the distribution of users to explore the number of cooperative users, we try to understand the minimal number of points needed to build a representative MMC model and we evaluate the spatio temporal pertinence of the training and test sets.

The datasets used in the experiments are described thereafter.

1. The Arum dataset [KRT10] is composed of the GPS traces of 5 researchers sampled at a rate of 1 to 5 minutes in the city of Toulouse from October 2009 to January 2011.

2. The Geolife dataset [ZLC08] has been gathered by researchers from Microsoft Asia and consists of GPS traces collected from April 2007 to October 2011, mostly in the area of Shanghai city. This dataset contains the mobility traces of 178 users captured at a very high rate of 1 to 5 seconds.
3. The *Nokia dataset* [Kiu09] is the result of a data collection campaign performed the city of Lausanne for 200 users started in September 2009 and that lasted for more than two years. The rate at which the location has been sampled varies depending on the current battery level.

4. The *San Francisco Cabs (SFC) dataset* [PSDG09] contains GPS traces of approximately 500 taxi drivers collected over 30 days, between May and July 2008, in the San Francisco area.

5. The *Borlange dataset* [FSH12] has been collected as a part of traffic congestion experiment over two years from 1999 to 2001. The public version of this dataset contains the GPS traces of 24 vehicles.

Table 1.2 summarizes the main characteristics of the datasets described above, namely the total number of users in the dataset, the collection period measured in the number of days as well as the average number of traces per user and the total number of traces in the dataset.

Conducting an analysis of the distribution of the average number of traces per user, we have observed that there is a high variance between the users with a small number of traces and those with a high number of traces. For instance, Figure 5.3, shows the distribution of the users according to their number of traces for the Geolife and Nokia datasets. While these distributions are peaked around the category of users that have between 1000 and 10000 traces, the other categories of users also contains a significant number of users. With respect to the other datasets (Arum, SFC and Borlange), they are close to be uniformly distributed with the exception of the SFC dataset, which also displays a peak distribution (cf. Figure 5.4).

![Figure 5.3: Distribution of the users according to their number of traces for the Geolife and Nokia datasets (the biggest datasets).](image)

Regarding the number of mobility traces per user, we found that there is considerable differences between the users that are more collaborative and those who are less cooperative (i.e. collaborative with respect to data collection or participative in a LBS). This phenomenon is shown in Table 1.2, we observe, for instance, that the difference for the GeoLife and Nokia dataset between the user with the minimal number of traces and the one with the maximal number...
of traces is really importanty. This issue affects the performance of the attack because non-cooperative users are more difficult to find due to the lack of location data. We illustrate this problem in Figure 5.3 and 5.4 where we observe that the most collaborative users represent only 25% for the GeoLife, 20% for the Nokia datasets and 40% for the San Francisco Cabs.

Concerning the dataset separation, we split his trail of mobility traces (chronologically ordered) into two disjoint parts of approximately the same size for each individual. The first half of the original data forms the training set, and will be used as the adversary background knowledge, while the second half constitutes the testing set on which the de-anonymization attack is conducted. For instance, if the original trail of one individual is composed of \( n \) mobility traces \( \{mt_1, mt_2, \ldots, mt_n\} \), it will be split into a training set \( \{mt_1, mt_2, \ldots, mt_{n/2}\} \) and a testing set \( \{mt_{n/2+1}, mt_{n/2+2}, \ldots, mt_n\} \) (for illustration purpose we assume that \( n \) is an even number). Therefore, the objective of the adversary is to de-anonymize the individuals of the testing set by linking them to their corresponding counterparts in the training set.

For simplicity reason, we assume thereafter that the training and the testing sets are composed of the same number \( N \) of persons while in general this might not be the case (e.g., the testing set could only contain a small fraction of the users of the training set). Of course, dividing the training and test sets based on the number of traces may not guarantee that the length (i.e., number of days) of the period they covered is exactly the same. Nonetheless, dividing the training and test sets based on the number of traces may not guarantee that training and test datasets have the same gathering period (i.e., number of days). For instance, a user of the Arum dataset, whose gathering begins on March 4 2010, ends on March 5 2011 and the \( n/2 \) trace is on November 20 2010 has a training and a test sets over a period of 256 and 125 days respectively. In addition, when gathering period is over a long period of time, like in GeoLife dataset, the spatiotemporal behavior could be different (i.e., they could move out, change job, etc…) impacting in both training and test sets. Accordingly, we need to evaluate the spatial and temporal pertinence of the training set to be used as input knowledge to perform the attack.

With reference to spatial behavior of users in the training and test sets. We measure the dis-
Chapter 5. De-anonymization and linking attack

Figure 5.5: Number of traces per hour in training, test and complete (all) dataset.

tance between both sets for each user in the GeoLife and Arum dataset. For instance, we split
the trail of mobility traces of Bob in training and test sets. Afterward, we make two MMCs us-
ing training and test sets for applying Stationary and Proximity distances in order to evaluate
the spatial behavior of Bob in both sets. Table 5.4 summaries the results of the spatial comport-
ment, for Arum we observe an average difference of 3.8 Km, for GeoLife 1.8 Km and for Nokia
0.5 Km. With respect to the Proximity distance, we observe that 60%, 70% and 95% of the users
have similar training and test sets in Arum, GeoLife and Nokia datasets, respectively (c.f., Table
5.4).

<table>
<thead>
<tr>
<th>Distance</th>
<th>Arum</th>
<th>GeoLife</th>
<th>Nokia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary, average in Km.</td>
<td>3,835</td>
<td>1,841</td>
<td>0.5235</td>
</tr>
<tr>
<td>Proximity (Percent of user with similar training and test sets)</td>
<td>60%, 70%, 95%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Spatial distance between training and test set using 3 different datasets.

With regard to user behavior over time, we measure the dissimilarity between training and
test sets. For this task, we rely on the Kullback-Leibler divergence [KL51], which is a non-
symmetric information theory metric to quantify the difference between two distributions. We
use the empirical distributions of the average number of trace per hour of the day in the origi-
nal, training and test sets, illustrated in Figure 5.5 to compute the divergence between training
and test sets of the different aforementioned locations datasets, which is summaries in Table
5.5. In our case the most important result is the divergence from training to test because we
use the training set to learn the MMC model and perform the attack over the test set. Con-
5.4. Experiments

sequently, the less dissimilarity or divergence the better the training set represents the test set. We observe in Table 5.5 that all values are close to zero, which means a good similarity between both training and test sets.

<table>
<thead>
<tr>
<th>Divergence from:</th>
<th>GeoLife</th>
<th>Nokia</th>
<th>Arum</th>
<th>San Francisco Cabs</th>
<th>Borlange</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training to test</td>
<td>0</td>
<td>0.0005</td>
<td>0.0082</td>
<td>0.0011</td>
<td>0.0042</td>
</tr>
<tr>
<td>Test to training</td>
<td>0</td>
<td>0.0005</td>
<td>0.008</td>
<td>0.0011</td>
<td>0.00391</td>
</tr>
</tbody>
</table>

Table 5.5: Kullback-Leibler divergence between training and test sets of location datasets.

In order to evaluate the impact of the size of the training set on the construction of a MMC, we have varied its size between 10% to 50% of the total number of traces, trying all slices by range of 10%. The value of 50% corresponds to the original “full” training dataset composed of the first half of the trail of traces. We choose to use the stationary distance (cf. Section 5.2.1) as a direct measure of the similarity between the MMC learnt on the reduced training set and the one learnt on the full training set. More precisely, a small stationary distance between the reduced training MMC and the full training MMC means that the two models are quite similar and thus that there was enough information contained in the considered training set to build a “representative MMC”. From Figure 5.6, we can observe that the more traces are used to build the training MMC, the smaller the stationary distance becomes between the reduced training MMC and the full training MMC. In particular, it seems that we need at least 30% of the total of the mobility traces to build a compact and representative MMC model.

![Figure 5.6: Stationary distance between the “reduced” training set and the “full” training set for the Geolife and the Nokia datasets, when the size of the training set varies between 10% to 50% of the total number of traces.](image)

In the following, we first focus on the Geolife dataset in order to analyze and understand the behavior of the de-anonymizers and distances.
Chapter 5. De-anonymization and linking attack

5.4.2 Fine-tuning clustering algorithms and de-anonymizers

The states of a MMC are extracted by running a clustering algorithm on the mobility traces of an individual. Therefore, the MMC generated (and by extension the success of the de-anonymization attack) is highly dependent on the clustering algorithm used and the accuracy of this algorithm, which may itself vary depending on the values of its parameters. The first step of our analysis consists in determining the parameters that leads to the best accuracy for the DJ-clustering algorithm.

Depending on the chosen values for the parameters, not all users of the original Geolife dataset will lead to the generation of a well-formed MMC. For instance, when the parameters of the clustering algorithm are too “conservative”, some users will not have enough mobility traces to identify frequent POIs, which results in their MMC being composed of only one state. On the contrary, choosing parameters that are too “relaxed” leads to the identification of a high number of POIs thus conducting to a MMC with too many states, which is detrimental to the success of the de-anonymization attack. Thus, the main objective of the tuning phase is to find the good set of parameters for the clustering algorithm that maximize the number of users in the training set whose MMCs does not consist in only one state while keeping the average number of POIs identified per user in an acceptable range.

First, we vary the three parameters of the clustering algorithm ($MinPts$, $r$ and the minimal number of days) and count the number of MMC generated that have more than one state. We found that these parameters are themselves correlated with the duration of the collection process and the sampling rate used. Table 5.6 summarizes the founded values of the clustering parameters used in the following experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>$MinPts$</th>
<th>$r$ (km)</th>
<th>Minimal nb of days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geolife</td>
<td>20</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>Nokia</td>
<td>10</td>
<td>0.05</td>
<td>10</td>
</tr>
<tr>
<td>Arum</td>
<td>40</td>
<td>0.05</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 5.6: Clustering parameters validated.

Once the parameters of the clustering algorithm have been tuned, we have analyzed the behavior of the de-anonymizer. In particular, the efficiency of the simple and weighted voting de-anonymizer needed to be assessed, which was the main objective of the following experiments. For the simple voting method, we first studied the influence of the number of candidates proposed by the minimal distance de-anonymizer used on each distance metric (stationary, matching, proximity and density-based). Each instance of this de-anonymizer proposes the $n$ candidates that have the smallest distances in a row, sorted in increasing order.

The simple vote de-anonymizer is applied to this list of candidates in order to output a single prediction. Figure 5.7 illustrates the success rate of this attack as a function of the number of candidates (more precisely as the percentage of true positives obtained over the total number of individuals) and the number of users that have well-formed MMCs (i.e., MMCs that have more than one state). This experiment was conducted on the Geolife dataset with the sampling rate varying in the range {10s, 30s, 60s, 120s}. 

106
5.4. Experiments

From Figure 5.7, we can observe that the number of users considered that have a well-formed MMC decreases as the sampling rate increases, because a low sampling rate results in fewer mobility traces to build MMCs. We can also observe that the success rate of the attack decreases as the number of candidates increases, which is not surprising as a higher number of candidates renders the task of the de-anonymizer more complex than when they are few candidates. For instance, the success rate of the attack when 4 candidates are generated by each distance metric is never more than 15%. Therefore, we can conclude that for the simple vote method considering only one candidate per distance is sufficient.

However, in some situations it is helpful to consider more than one candidate per distance metric but their weight should be set to be different values, which is the main idea behind the weighted vote de-anonymizer. In our experiments, we have compared two different ways to weight candidates, one based on linear weights and the other on exponential ones to determine the most effective one (Figure 5.8). In a nutshell, the linear method assigns weights in a decreasing linear form and the exponential method assigns weights in a decreasing exponential form starting at $2^n$, $n$ being the number of candidates. Table 5.7 illustrates the assigned weights.
weights for these two methods, while Figure 5.8 compares the success rate of the weighted vote de-anonymizer using both weighting systems for different sampling rates. From these experiments, we can observe that the exponential method seems to be more efficient as its success rate is about 5% better than with the linear method. Moreover, this de-anonymizer seems to be robust to data sampling with different rates.

Table 5.7: The linear and exponential weighting schemes.

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Linear weight</th>
<th>Exponential weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>2</td>
<td>$n - 1$</td>
<td>$2^{n-1}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$n$</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

5.4.3 Measuring the efficiency of de-anonymizers

To measure the success rate of the proposed de-anonymizers, we have sampled the Geolife dataset at different rates and observed the influence of the sampling on the success rates of the de-anonymizers. Figure 5.9 shows that the success rate of the attack with the minimal stationary distance and the minimal proximity distance varies from 20% to 40%, but that the best performing predictor is stat-prox with results ranging from 35% to 40%. By studying how successfully identified users tend to be the same across the different experiments (cf. 5.8), we have observed that a significant fraction of them tends to be re-identified with success independently of the sampling conditions. We believe that for these particular users their mobility behaviors is so regular that they are highly resilient to downsampling.

Figure 5.9: Success rate of the combined de-anonymizer.

At this point of the experiments, it seems important to be able to compare precisely the de-anonymizers. Indeed, the success rate of a de-anonymization attack is not the only aspect that should be considered. For instance, for an adversary a possible strategy is to focus on weak individuals that offer a high probability of success for the attack rather than being able...
5.4. Experiments

Figure 5.10: ROC curve for the Geolife dataset.

<table>
<thead>
<tr>
<th>Table 5.8: Some users are resilient to sampling. For this experiment we take into account down-sampling of 5s, 10s, 30s and 60s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal number of generated models</td>
</tr>
<tr>
<td>Number of users found 4 times</td>
</tr>
<tr>
<td>Number of users found 3 times</td>
</tr>
<tr>
<td>Number of users found twice</td>
</tr>
<tr>
<td>Number of users found only once</td>
</tr>
<tr>
<td>Number of users never found</td>
</tr>
</tbody>
</table>

to de-anonymize the entire dataset. Measuring the probability of success of the inference attack for a given individual is similar to have some kind of confidence measure for a given de-anonymization candidate. Deriving this confidence measure is quite intuitive for our de-anonymizers. Indeed, for the minimal distance ones, the smaller is the distance, the higher the confidence.

In order to compare the performance of the de-anonymizers, we rely on the notion of Receiver Operating Characteristic (ROC) curve [Faw06]. In a nutshell, a ROC curve is a graphical plot representing the sensitivity (i.e., as measured by the true positives rate versus false positives rate) for a classifier. In our particular case, the ROC curve is built based on the confidence of the de-anonymizer. More precisely, for each anonymous candidate, an identity is assigned as well as the distance between the MMCs that was used to assign the identity to the candidate. Thus, the smaller the distance (or the higher the votes) the more confidence one can have in the result. Consequently, to build the ROC curve the results are sorted by confidence in order to have the true positives at the beginning. The intuition behind this curve is that between two de-anonymizers achieving the same success rate, one should favor the one displaying the highest confidence. In Figure 5.10, the ROC curve shows the true positives rate (TPR) versus the false positives rate (FPR) for the best performing de-anonymizers, with the candidates sorted
Chapter 5. De-anonymization and linking attack

Figure 5.11: Success rate of the de-anonymizers on the Nokia dataset.

by ascending distance. This ROC curve further confirms that the stat-prox de-anonymizer is the best alternative among the de-anonymizers we designed.

Our approach performs fairly well for the Geolife dataset as the achieved success rate is between 35% and 45% for the stat-prox de-anonymizer (Figure 5.9). In order to further validate the approach, we applied it on the Nokia dataset. This dataset has 195 users, among which we can generate a “valid” MMC composed of more than one POI for 157 users using the parameters described previously. As shown on Figure 5.11, the success rate varies between 35% and 42%, with the best score obtained again by the stat-prox de-anonymizer.

5.4.4 Fair comparison with prior work

In this section, we have presented various experiments on de-anonymization attacks that lead to the definition of an heterogeneous de-anonymizer called stat-prox, which obtains a success rate between 42% and 45% on different datasets. While at first glance, this performance may seem to be poorer than the one achieved by the predictors of Ma et al. [MYYR10], which goes up to 60% to 90%, and the predictors of Freudiger and co-authors [FSH12] that have a re-identification rate of 70%, we believe that these results are not directly comparable because we clearly differentiate between the training set and the testing set, while these authors perform the learning and the testing on the same dataset, thus inducing a strong experimental bias.

Indeed, our mobility models are built out of the training set, which is disjoint from the test set, whereas one of the adversary model of Ma et al. directly extracts mobility traces forming the test set from the training set. Moreover in our case, the training data is temporally separated from the test data (i.e., the training and the test have been recorded at different non-overlapping periods of time) because the whole dataset has been split into two temporally disjoint parts, whereas the second adversary model of Ma et al. picks the information it uses to de-anonymize within the same period as the test data is recorded. Therefore, our approach is quite different from them as our attack consists first in collecting mobility data from an individual, before trying to identify this individual in a so-called anonymized dataset, while their attack aims at gathering location data at the same time at which the de-anonymization attack occurs. In
addition, one important parameter of their attack is the number of timestamped location data collected, which can be compared to the number of states we have in our mobility model. On average and depending on the dataset considered, we have between 4 and 8 states per MMC, which correspond to a compact representation of the mobility behavior of an individual. When restricted to such limited of information in terms of the number of timestamped location data, the attacks proposed by Ma et al. do not perform well, with a de-anonymization rate between 10% to 40%. Similarly in the work of Freudiger and co-authors, mobility samples are taken from home/work, POIs or frequent visited places without any distinction between training or testing sets, which again introduces an evaluation bias. For instance, when 90% of the samples are taken either from home or work, the success rate of the re-identification is around 75% while a uniform random sampling leads to a success rate for the re-identification of 10%.

For comparison purpose, we conducted the de-anonymization attack without separating the training and testing sets. The results obtained by related work and for the stat-prox de-anonymizer in this setting using different datasets are summarized in Figure 5.12. These experiments are, as expected, so biased that they lead to a success rate close to 100% for all the datasets. Once again, we do not pretend that our de-anonymization attack would achieve a success rate of nearly 100% and beat all the previous methods if tested in the same conditions (for instance in some settings the test set was only a subset of the training set as it was sampled from it). We are merely pointing out that for fairness issues, it is important to compare de-anonymizers using the same setting and that in order to reduce the experimental bias, the training and testing set should be clearly separated (which is not the case in almost all the previous works).

![Figure 5.12: Success rate of the stat-prox de-anonymizer when the training and testing sets are the same.](image)

**5.5 Summary**

In recent years, several privacy breaches related to location data have reached the headlines. For instance, the German deputy Malte Spitz sued Deutsche Telekom to obtain the last six months of location data generated from his phone [onl09]. Then, he published this data in
the form of an interactive map showing that the combination of location data with contextual information could lead to a serious privacy breach. Another example of privacy buzz was the article about telephone constructors [JA11] published in The Wall Street Journal revealing that Apple and Google collect on a large scale location data using unique identifiers in order to develop novel location-based services.

The classical argument used by data collectors use is that by itself location data is anonymous and thus can be collected from users without violating their privacy. Unfortunately, as shown by our work in this paper this argument is fallacious. More precisely, we have demonstrated that geolocated datasets gathering the movements of individuals are particularly vulnerable to a form of inference attack called the de-anonymization attack. More precisely, we have shown that the de-anonymization attack can re-identify with a high success rate the individuals whose movements are contained in an anonymous dataset provided that the adversary can used as background information some mobility traces of the same individuals that he has been able to observe during the training phase. From these traces, the adversary can build a MMC that models in a compact and precise way the mobility behavior of an individual. We designed novel distances quantifying the similarity between two MMCs and we described how these metrics can be combined to build de-anonymizers. The de-anonymization attack is very accurate with a success rate of up to 45% on large-scale real datasets and this even if the mobility traces are sanitized by downsampling them (e.g., every 2 minutes instead of every 10 seconds).

In summary, the mobility behavior of an individual is far from being random [GHAL08] and tends to be unique thus acting as a signature of an individual [dMHVB13]. For instance, even the pair home/work could act as a quasi-identifier [GKNndPC11a]. In addition to location data, the knowledge of the social network, as shown by the works of Srivatsa and Hicks and Sharad and Danezis, can be used as side information to help in performing the de-anonymization. However, it might be possible to mitigate this risk of re-identification by sanitizing the social graph before releasing it [GI13].
Chapter 6

Conclusion and Perspectives

"You lose your privacy, and sometimes, people do not see you as human."
— Shawn Wayans.

The use of geolocation application is becoming an everyday habit as smart cities are more and more connected to mainstream cell phones with GPS capabilities. Today is easy to use applications to check the bus schedule based on our current position, verify congestion level in highway before going back home, navigate to reach a new address, etc. Thus, a lot of mobility traces are generated in a high frequency and automatic way (raw big data). Accordingly, as brought up in the introduction of this dissertation, our study sought to answer these questions:

I. What kind of information could be inferred from this geolocated datasets?
II. How could these inferences be implemented?
III. What is the privacy impact?
IV. Is there a comprehensive classification or taxonomy of inference attacks over geolocated data?

The main empirical findings are chapter specific and were summarized within the respective chapters: Classification of inference attacks on geolocated data (cf. Chapter 1), Extraction of points of interest (cf. Chapter 2), Mobility Markov chain model (cf. Chapter 3) and inference attacks using MMC, such as prediction future locations, semantic extraction (cf. Chapter 4) and de-anonymization (cf. Chapter 5). The current chapter presents the conclusion of this dissertation in Section 6.1 as well as the perspectives of future works in Section 6.2.

6.1 Conclusion

This section will synthesize the findings to answer the study’s research questions.

I. What kind of information could be inferred from this geolocated datasets?
a. **Points of interest** detection is performed using clustering algorithm. We did a benchmarking, in terms of precision, recall, f-measure, time, complexity and number of needed parameters, among some algorithm like DBSCAN, k-means, DT cluster, TD cluster or DJ cluster. We have found that the best algorithm is DJ cluster, which performs this task with a precision of 74% and recall of 52%.

b. **Future locations** prediction was done using the mobility Markov chain (MMC) model, the accuracy of this prediction algorithm ranges from 70% to 95%. We were able also to establish someone’s predictability, which is a theoretical metric based on the MMC, to measure how predictable someone is.

c. **Identity** could be inferred through de-anonymization attack, which relies on the distance between mobility models due to our MMC acts as a signature. Then, we measure the similarity from one model to another in order to find the corresponding individual. This inference is accurate with a success rate of up to 45% on large-scale real datasets and even in presence of sanitized by downsampling mobility traces.

d. **Semantic of personal spots** is extracted from the structure of the mobility Markov chain model and/or using heuristics. We were not able to quantify the precision nor the accuracy of the inference car, to the best of our knowledge, there is not a publicly able dataset with the semantic labels of users in it.

II. How the inference process could be implemented?

a. **Mobility model:** the inference attacks presented in the current study does not rely on “Ad-hoc” mobility models. Conversely, the inference attacks were implemented using a single mobility model based on Markov chains, which is a compact and representative model able to perform all the attack we have establish in our classification.

b. **Training and test:** the way how dataset is split into training and test impacts directly the precision and accuracy of the inference.

c. **Parameter tune:** a methodology to maximize the points of interests extraction based on cluster quality indexes was proposed.

III. What is the privacy impact?

a. **Privacy** could not be measured in a global way. It should be done service per service. More precisely, it is not possible to evaluate in same way a service that provides weather reports based on user location, where a coarse grain mobility trace is needed, and a service to locate the nearest restaurant using as input a fine grain mobility trace.

b. **Anonymity:** simply replacing the real names of individuals by pseudonyms before releasing a dataset is usually not sufficient to preserve the anonymity of their identities because the mobility traces themselves contain information that can be uniquely linked back to an individual.

c. **Privacy metrics:** The theoretical (predictability) and empirical (accuracy, precision, recall and entropy) metrics could be use for further understanding about geoprivacy and privacy risk leakage to establish a method for privacy risk quantification in ubiquitous systems.
IV. Is there a comprehensive classification or taxonomy of inference attacks over geolocated data?

a. **Classification**: A first attempt to make a classification was done by Wernke [WSDR12]. Nevertheless, the main difference from previous work is that our classification is from the point of view of the adversary objectives and it is more complete.

The study has offered an evaluative perspective on inference attacks over geolocated data as well as a classification of these inferences based on the adversary objective. The study encountered a number of limitations, which need to be considered, such as:

- The lack of publicly available big datasets. In one hand, apart from the public datasets proposed by Crawdad and Geolife, to the best of our knowledge there are not big datasets with a GPS granularity. On the other hand, there were bigger datasets like those published by Nokia in the Mobility Data Challenge (MDC) [LGPA+12] and by Orange in the Data for Developmental (D4D) challenge [Ora13]. Nevertheless, it is not possible to test inference attacks over these datasets due to utilization contract agreement forbid it even if it is for scientific purposes.

- The absence of ground truth of semantics of points of interest. The only dataset we have found at GPS granularity, which have annotation of POIs was the LifeMap dataset. Nonetheless, the annotation is only binary where 1 represents mobility trace belonging to a personal spot. To the best of our knowledge there not exist a dataset at GPS granularity with semantic labels like: home, work, bar, etc..

- The time holes in Call Detail Records (CRD). When using mobility traces issued from a CDR we must confront the incompleteness of the location data due to in this kind of datasets position is sample when the telephone send or receive a SMS or a call. Thus in the records, we have the illusion that users teletransport from one position to another.

- Utility and privacy of location data is application dependent and could not be quantified in global way. For instance, if we use a weather forecasting application it is only necessary to reveal the location at a city granularity (e.g., Toulouse) but if we need to know the nearest ATM to our current location a GPS granularity is needed. As result, we could not evaluate privacy nor utility in the same way for both applications.

**6.2 Perspectives**

We are planning to extend the current work by following several avenues of research in four different axes:

I. Mobility Markov chain model.

- Establish the distribution of time for the transitions in the model.
- Calculate the probability distribution of leaving a state based on the time spent in that state.
Chapter 6. Conclusion and Perspectives

- Compute the average speed as well as the transportation mean for all the transition.
- Trajectory clustering to establish the different trajectories and the average time per group of similar trajectories to go from one state to another one.

II. Inference attack.

- Infer the points of interests from where someone comes to the actual one, in order to complete possible temporal gaps in the trail of mobility traces.
- Discover social networks based on MMC similarity.
- Infer demographic characteristics.
- Establish a classification of adversaries characteristics.

III. Sanitization technique.

- Use the mobility Markov chains to build synthetic mobility traces that meet the trade off between privacy and utility.
- Clustering of mobility patterns based on MMC distances to sanitize geolocated datasets.

IV. Privacy metric.

- Quantify privacy based on the metrics of the developed inference attacks.
- The theoretical (predictability) and empirical (accuracy, precision, recall and entropy) metrics could measure the level of privacy.

In the present work we have explored the risk of privacy leakage in the context of geolocation by building a classification of possible inferences, the mobility model to implement the inferences and the attacks that could be performed by an adversary who is in possession of a dataset constituted of trails of mobility traces of different individuals.
Appendix A

Actors of privacy
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Benefits</th>
<th>Constraints</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo-Masking</td>
<td>Uses modification of the geographic coordinates of original data points to add stochastic or deterministic noise, thus hiding a point’s original location associated with particular data</td>
<td>Allows for a case-by-case determination of appropriate balance between privacy and accuracy based on density or other factors</td>
<td>Accuracy of data and its analysis is heavily impacted by degree of anonymity desired and may be untrustworthy if not balanced correctly</td>
<td>[KCS04]; [LC06]</td>
</tr>
<tr>
<td>Cloaking</td>
<td>Individual user’s geographic data are reported at lower spatiotemporal resolutions than initially recorded in order to lower data precision and protect privacy</td>
<td>Cloaking both spatial and temporal data attributes provides a higher degree of privacy than might otherwise be obtained</td>
<td>Loss of precise data may create difficulties for accurate analysis</td>
<td>[GL04]; [CZBP06]; [GG03]</td>
</tr>
<tr>
<td>Mix Zone</td>
<td>User identities are anonymized by restricting locations of identification</td>
<td>Allows for short-term accuracy of position determination, thus allowing for effective analysis and service provision</td>
<td>Difficult to effectively anonymize in a sparse network</td>
<td>[BS04]; [FSH09]</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Combines individual or group-level data to reduce identification of individuals; location data may be aggregated over time or space</td>
<td>Reasonably simple and resource-efficient method of deidentifying data</td>
<td>May lead to a loss in usefulness depending on planned data uses and degree of aggregation</td>
<td>[CCP’01]; [Cas07]</td>
</tr>
<tr>
<td>Encryption</td>
<td>Sent or stored data are encrypted to render them essentially inaccessible to nontrusted parties</td>
<td>Raw data still are available in unencrypted form for research or analysis by trusted parties, thus minimizing loss of accuracy</td>
<td>May require trusted third party, increase in computing resources; decryption by malicious party is possible</td>
<td>[HM04]; [SLH’07]</td>
</tr>
<tr>
<td>Pseudonyms</td>
<td>Data are sent under assumed names or identities; often recommended that pseudonyms change periodically and be used in conjunction with mix zones</td>
<td>Allows location and other data to be gathered without directly linking the records to an individual user</td>
<td>Ineffective in sparse networks; may require linkage of actual identification to pseudonym</td>
<td>[BHV07]; [Ben11]</td>
</tr>
</tbody>
</table>

Table A.1: Responsible party: Developer
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Benefits</th>
<th>Constraints</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deidentification</td>
<td>Removal of personally identifying information (such as name, social security number, etc.) from a dataset so that there would be no reasonable basis to assume that the data subject could be uniquely identified</td>
<td>Existence of fairly standard methods for deidentification; may be conducted at the database level, thus does not need to be built into the front end of application development</td>
<td>May not be adequate to protect privacy depending on how well data are deidentified (for example, if location data are revealed, it may be possible to reidentify)</td>
<td>[Ohm10]; [Swe00]; [CWM08]</td>
</tr>
<tr>
<td>Access limitations</td>
<td>Limitations are set regarding who may have access to what degree of data; may be set via such methods as password protection, limitations on third-party sharing, or keeping data on local networks only</td>
<td>Allows raw data to be retained while ensuring some degree of protection; theoretically provides a “paper trail” in the event of accidental or malicious disclosure</td>
<td>Stored data still may be subject to misuse depending on trustworthiness of persons granted access; may limit usefulness of data depending on degree of restriction</td>
<td>[NBL*10]; [MFD03]</td>
</tr>
<tr>
<td>Secure storage</td>
<td>May involve physical security (such as keeping data in locked offices or on a non-networked machine), methods such as encryption, which prevent unauthorized users from accessing stored data, distribution of data across several storage units, or other methods</td>
<td>Provides physical evidence of data security; limits access to trusted parties; augments other forms of privacy protection</td>
<td>Potential for compromise may be large, for physical security may be violated; if data are available in raw format to some persons, confidentiality may be compromised if untrusted parties gain access</td>
<td>[AU99]</td>
</tr>
<tr>
<td>Statistical privacy</td>
<td>Protecting data so that processes such as statistical computing, query responses, and other process may be performed without revealing sensitive information</td>
<td>Allows for statistically valid analysis techniques while protecting individual identifiers</td>
<td>May be difficult to determine the actual level of protection, particularly if additional datasets are available for combination and mining</td>
<td>[Dwo08]; [FDCdVP*11]</td>
</tr>
</tbody>
</table>

Table A.2: Responsible party: Agencies (Public & Private)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Benefits</th>
<th>Constraints</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved design of privacy policies</td>
<td>Clarification of user rights and responsibilities indicated in clear language that maps to consumer expectations</td>
<td>May be applied consistently to policies for all types of applications and services</td>
<td>Improved design and clarification of policies will not guarantee that consumers will read or review privacy policies</td>
<td>[CP11]; [EAM+05]; [Miy08]; [CC11b]</td>
</tr>
<tr>
<td>Improved consumer privacy advocacy and rights</td>
<td>Increased attention to privacy advocacy groups and improved funding for privacy advocacy activities</td>
<td>Extends the reach of expertise of persons involved with and expert in privacy policies</td>
<td>May not have the funding or reach of commercial or retail lobbying groups, thus may be limited in effect</td>
<td>[Giv00]; [Ben11]</td>
</tr>
<tr>
<td>Consumer privacy education and awareness</td>
<td>Increased educational activities taking place around the need to effectively treat and manage data protection</td>
<td>Provides consumers with information needed to make effective decisions pursuant to protection of their personal data and information</td>
<td>Limited by the willingness of consumers to spend time managing their personal information settings and knowledge of need to do so</td>
<td>[Par11]; [WZ11]</td>
</tr>
<tr>
<td>Self-regulation</td>
<td>Placing emphasis on requirements that consumers thoroughly read privacy policies, set strong passwords, evaluate decisions to participate in services based on privacy preferences, etc.</td>
<td>Provides consumers with the responsibility to determine their own preferences and take accountability for their actions pursuant to the gathering of personal data</td>
<td>Limited by the complexities of privacy policies and attitudes and approaches toward privacy protections by various service providers</td>
<td>[RHGW09]; [PME09]</td>
</tr>
</tbody>
</table>

Table A.3: Responsible party: Users
Appendix B

Resume of the state of the art and datasets used to perform inference attacks
<table>
<thead>
<tr>
<th>Method</th>
<th>Technology</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dwelling time</td>
<td>GPS</td>
<td>YellowCabs [PSDG09]</td>
</tr>
<tr>
<td>Scoring function</td>
<td>WiFi</td>
<td>Arum [KRT10]</td>
</tr>
<tr>
<td></td>
<td>GSM</td>
<td>Nokia [Kiu09]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GeoLife [ZLC &quot;08]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Borlange city [FSH12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LifeMap [CC11a]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMLS [KH05]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIT reality [EPL08]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quinta [ABCDM09]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Statefair [RSH'09]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CenceMe [MLF '08]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dartmouth College WIFI access point [KHAY09]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ile Sans Fils [LGoP07]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Augsburg Indoor Location [Pet04]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDC [LGPA '12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D4D Orange [Ora13]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIT Media Lab dataset [ESP06]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContextPhone [ROPT05]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntetic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unpublish dataset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note</td>
</tr>
</tbody>
</table>

Table B.1: Identification of points of interests (POI).
<table>
<thead>
<tr>
<th>Method</th>
<th>Technology</th>
<th>GPS</th>
<th>WiFi</th>
<th>GSM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stochastic model</strong></td>
<td>YellowCab [PSDG09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Arum [KRT10]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nokia [Kiu09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Goode [ZLC08]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Borlang city [SH12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LifeMap [CC11a]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MMS [Kiu09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIT reality [EP08]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quinns [ABCdM09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StateFair [RSH10]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ConceMe [MLF08]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dartmouth College WiFi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>access point [KHA09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ile Sans Fil [LGp07]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Augsburg Indoor Location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Pwil04]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MDC [LGP12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D4D Orange [Pwil06]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIT Media Lab dataset [EPL06]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ContextPhone [ROY05]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synthetic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unpublish dataset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Markov model</strong></td>
<td>Chon et al. [CSTC12]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ashbrook et al. [AS03]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gambs et al. [GKdPC12]</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Mixed markov</strong></td>
<td>Asahara et al. [AMSS11]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Trajectory based</strong></td>
<td>Krumm and Horvitz [KH06]</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Froehlich and Krumm [FK08]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Semantic</strong></td>
<td>Ying et al. [YLWT11]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alvares et al. [ABK07b]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Compression techniques</strong></td>
<td>Asahara et al. [AMSS11]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feder et al. [FMG92]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Sampled pattern matching</strong></td>
<td>Jacquet et al. [JSA00]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Comparative studies</strong></td>
<td>Song et al. [SQBB10]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Petzold et al. [PBTU06]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Classification based</strong></td>
<td>Etter et al. [EKK12]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wang and Prabhala [WP12]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gao et al. [GTL12]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table B.2: Next place prediction.
<table>
<thead>
<tr>
<th>Technology</th>
<th>GPS</th>
<th>WiFi</th>
<th>GSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Geolife [ZLC08]</td>
<td>Borlange city [SH12]</td>
<td>LifeMap [CC11a]</td>
</tr>
<tr>
<td></td>
<td>MMLS [Kiu09]</td>
<td>MIT reality [PRT08]</td>
<td>Quinta [ABCAM09]</td>
</tr>
<tr>
<td></td>
<td>Statefair [RSH09]</td>
<td>CenceMe [MLF08]</td>
<td>CeWeMe [MLF08]</td>
</tr>
<tr>
<td></td>
<td>Ile Sans Fils [LGoP07]</td>
<td>Darmouth College WiFi access point [RAH109]</td>
<td>Augsburg Indoor Location [Pet04]</td>
</tr>
<tr>
<td></td>
<td>Dartmouth College WiFi access point [RAH109]</td>
<td>MIT Media Lab dataset [ESP06]</td>
<td>MIT Media Lab dataset [ESP06]</td>
</tr>
<tr>
<td></td>
<td>ContextPhone [ROPT05]</td>
<td>Dartmouth College WiFi access point [RAH109]</td>
<td>ContextPhone [ROPT05]</td>
</tr>
<tr>
<td></td>
<td>Syntetic</td>
<td>Unpublish dataset</td>
<td>Synthetic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Linking traces</th>
<th>De-anonymization attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linking traces</td>
<td>De Mulder et al. [DMDBP08]</td>
<td>✓</td>
</tr>
<tr>
<td>Linking traces</td>
<td>Zang and Bolot [ZB11]</td>
<td>✓</td>
</tr>
<tr>
<td>Linking traces</td>
<td>Ma et al. [MYYR10]</td>
<td>✓</td>
</tr>
<tr>
<td>Linking traces</td>
<td>Shokri et al. [STLBH11]</td>
<td>✓</td>
</tr>
<tr>
<td>Linking traces</td>
<td>Gambs et al. [GKNndPC13]</td>
<td>✓ ✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>From external sources</td>
<td>Cao et al. [CC]10</td>
<td>✓</td>
</tr>
<tr>
<td>From external sources</td>
<td>Ying et al. [YLWT11]</td>
<td>✓</td>
</tr>
<tr>
<td>From external sources</td>
<td>Alvares et al. [ABK07b]</td>
<td>✓</td>
</tr>
<tr>
<td>From external sources</td>
<td>Shokri et al. [SC13]</td>
<td>✓</td>
</tr>
<tr>
<td>Using heuristics</td>
<td>Gambs et al. [GKNndPC10a]</td>
<td>✓</td>
</tr>
<tr>
<td>Using heuristics</td>
<td>Isaacman et al. [IBC11]</td>
<td>✓</td>
</tr>
<tr>
<td>Using heuristics</td>
<td>Girardin et al. [GVGB09]</td>
<td>✓</td>
</tr>
<tr>
<td>From the model</td>
<td>Gambs et al. [GKNndPC11a]</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>From the model</td>
<td>Liao et al. [LFK07]</td>
<td>✓</td>
</tr>
<tr>
<td>From the model</td>
<td>Ye et al. [YSL11]</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table B.3: Linking records and semantic extraction.
<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Technology</th>
<th>GPS</th>
<th>WiFi</th>
<th>GSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detecting communities</td>
<td>Musolesi and Mascolo [MM07]</td>
<td>YellowCabs</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nunes [Nun12]</td>
<td>Arun [KRT10]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Boldrini [BP10]</td>
<td>Nokia [Kiu09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discover relations</td>
<td>Jedrzejczyk et al. [JPBN08]</td>
<td>GeoLife [ZLC08]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Eaglesa et al. [EPL09]</td>
<td>LabMap [KC11a]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Braga et al. [BTBM12]</td>
<td>MIT reality [EF108]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extract demographic attributes</td>
<td>Kelly et al. [KSC13]</td>
<td>Quinta [ABGdM09]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Gonzales et al. [GHAL08]</td>
<td>Statefair [RSH09]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Song et al. [SQBB10]</td>
<td>CenseMe [MLF08]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>de Montjoye et al. [dMQRP13]</td>
<td>Dartmouth College WiFi access point [RFAX09]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ile Sans Fils [LG07]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Augsburg Indoor Location [Pet04]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDC [LGPA12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAD Orange [Rou13]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIT Media Lab dataset [ESP06]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ContextPhone [ROP05]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Synthetic</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table B.4: Discover social links and demographic attributes.
Appendix B. Resume of the state of the art and datasets used to perform inference attacks
Bibliography


Allan J. Brimicombe. GIS Where are the frontiers now? In *ACM International Workshop on Advances in Geographic Information Systems*, pages 33–45, Manama, Bahrain, November 2002.


[Swe00] Latanya Sweeney. Uniqueness of simple demographics in the U.S. population, January 2000.


