FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MIXED SPATIAL AND NONSPATIAL PROBLEMS IN LOCATION BASED SERVICES

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Jaime Ballesteros

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Jaime Ballesteros, and entitled Mixed Spatial and NonSpatial Problems in Location Based Services, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Masoud Sadjadi

Christine Lisetti

Malek Adjouadi

Naphtali Rishe, Major Professor

Date of Defense: June 17, 2013

The dissertation of Jaime Ballesteros is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

Esta Disertacion va dedicada muy especialmente a mi familia. Primero mis padres, Elssy y Jaime, que me dieron todas las herramientas necesarias para poder cumplir mis metas. El amor, el sacrificio y dedicacion con que nos formaron a mis hermanos y a mi seran siempre motivo de inspiracion para continuar adelante. A mis hermanos Lili, Meli, Randy e Isabella, por apoyarme y darme muchas alegrias y motivos de orgullo.

Por supuesto, no podria faltar la dedicacion a mi hermosa esposa, Pily. Desde que comenzamos juntos este viaje, siempre ha estado conmigo compartiendo todos los momentos, lo mas felices y los mas dificiles y nunca ha desistido. Para ella, que es mi fuerza inspiradora y mi amor, va dedicada esta disertacion. Finalmente, quisiera dedicar este logro a mi Maestro. El sin duda me ha dado todo lo que tengo y esto es simplemente parte de su obra. Gracias por todo.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Naphtali Rishe for his great support and advise during all these years. Thank you for teaching me that only by working hard, great achievements and goals can be reached.

I also want to thank the other members of my Ph.D Committee, Dr. Christine Lisetti, Dr. Masoud Sadjadi and Dr. Malek adjouadi for their time taken in reading this dissertation and for their support and valuable feedback.

Special thanks goes to Dr. Bogdan Carbunar, who has been a great collaborator and teacher, and to Mr. Mahmudur Rahman. We all created such an exceptional team and we were able to contribute in several aspects in our research. Their valuable insights and feedback were the key to publish our work.

Finally, I want to thank all members of the High Performance Database Reseach Center. This lab was like my home during all these years and I made great friends along the way.

The material in this dissertation is based in part upon work supported by the National Science Foundation under Grant Nos. CNS-0821345, CNS-1126619, HRD-0833093, IIP-0829576, CNS-1057661, IIS-1052625, CNS-0959985, OISE-1157372, IIP-1237818, IIP-1215201, IIP-1230661, IIP-1026265, IIP-1058606, IIS-1213026, OISE-0730065, CCF-0938045, CNS-0747038, CNS-1018262, CCF-0937964.

ABSTRACT OF THE DISSERTATION

MIXED SPATIAL AND NONSPATIAL PROBLEMS IN LOCATION BASED SERVICES

by

Jaime Ballesteros

Florida International University, 2013

Miami, Florida

Professor Naphtali Rishe, Major Professor

With hundreds of millions of users reporting locations and embracing mobile technologies, *Location Based Services* (LBSs) are raising new challenges. In this dissertation, we address three emerging problems in location services, where geolocation data plays a central role. First, to handle the unprecedented growth of generated geolocation data, existing location services rely on geospatial database systems. However, their inability to leverage combined geographical and textual information in analytical queries (e.g. spatial similarity joins) remains an open problem. To address this, we introduce *SpsJoin*, a framework for computing spatial set-similarity joins. SpsJoin handles combined similarity queries that involve textual and spatial constraints simultaneously. LBSs use this system to tackle different types of problems, such as deduplication, geolocation enhancement and record linkage. We define the spatial set-similarity join problem in a general case and propose an algorithm for its efficient computation. Our solution utilizes parallel computing with MapReduce to handle scalability issues in large geospatial databases.

Second, applications that use geolocation data are seldom concerned with ensuring the privacy of participating users. To motivate participation and address privacy concerns, we propose *iSafe*, a privacy preserving algorithm for computing safety snapshots of co-located mobile devices as well as geosocial network users.

iSafe combines geolocation data extracted from crime datasets and geosocial networks such as Yelp. In order to enhance iSafe's ability to compute safety recommendations, even when crime information is incomplete or sparse, we need to identify relationships between Yelp venues and crime indices at their locations. To achieve this, we use SpsJoin on two datasets (Yelp venues and geolocated businesses) to find venues that have not been reviewed and to further compute the crime indices of their locations. Our results show a statistically significant dependence between location crime indices and Yelp features.

Third, review centered LBSs (e.g., Yelp) are increasingly becoming targets of malicious campaigns that aim to bias the public image of represented businesses. Although Yelp actively attempts to detect and filter fraudulent reviews, our experiments showed that Yelp is still vulnerable. Fraudulent LBS information also impacts the ability of iSafe to provide correct safety values. We take steps toward addressing this problem by proposing *SpiDeR*, an algorithm that takes advantage of the richness of information available in Yelp to detect abnormal review patterns. We propose a fake venue detection solution that applies SpsJoin on Yelp and U.S. housing datasets. We validate the proposed solutions using ground truth data extracted by our experiments and reviews filtered by Yelp.

TABLE OF CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1
1.1 Motivation 1
1.2 Research Approach 4
1.3 Main Contributions 5
1.4 Outline of the Dissertation 8

2. LITERATURE REVIEW 10
2.1 Spatial Set-Similarity Joins 10
2.1.1 Assumptions 10
2.1.2 Spatial Joins 11
2.1.3 Set Similarity Joins 14
2.1.4 Spatio-Textual Queries 15
2.2 Location Based Services Applications: Safe Cities Approach 16
2.2.1 Safe Cities 16
2.2.2 Participatory Sensing 17
2.2.3 Crime Prediction 18
2.3 Fake Review Detection and Opinion Spam in Location Based Services . . 19
2.3.1 Sybil detection 21
2.3.2 Web Crawling and Data Collection Process 22

3. SUPPORTING SPATIAL SET-SIMILARITY JOINS IN LOCATION BASED SERVICES 23
3.1 Introduction 23
3.2 System Architecture 25
3.2.1 Data Repository 26
3.3 Spatial Similarity Join 26
3.3.1 Tools 28
3.3.2 Processing Spatial Set Similarity Joins 29
3.3.3 Query Processing 33
3.3.4 Data Visualization 34
3.3.5 Framework Demonstration 35
3.4 Extending SpsJoin 36
3.4.1 Returning Relevant Pairs 36
3.4.2 Selecting the Best Match 37
3.4.3 Geographic Inverse Record Frequency 39
3.4.4 Incorporating *Girf* Values Into the Content Similarity 41
3.4.5 Definitions 42
3.4.6 Incorporating Uncertainty Regions 42
3.4.7 Putting it All Together 44
3.4.8 Incorporating Uncertainty: Entropy 45

3.5 Improving SpsJoins . . . 46
3.5.1 Spatial Partitioning Phase . . . 46
3.5.2 Local Join Implementation . . . 48
3.5.3 On Radius and Distance Computations . . . 51
3.6 Experimental Evaluation . . . 52
3.6.1 Performance . . . 53
3.6.2 Join Precision . . . 54
3.6.3 Experiment Definition . . . 55
3.7 Appendix . . . 58
3.7.1 SpsJoin Implementation . . . 58
3.7.2 Creating A Configuration File . . . 60
3.7.3 Running SpsJoin . . . 62
3.7.4 Appendix Conclusions . . . 66

4. TOWARDS PRIVACY PRESERVING LOCATION BASED SERVICE APPLICATIONS. THE SAFE CITIES CASE. . . . 67
4.1 Introduction . . . 67
4.2 Model and Background . . . 70
4.2.1 Geosocial Networks . . . 71
4.2.2 Crime Data . . . 73
4.2.3 Forecasting Tools . . . 75
4.2.4 Attacker Model . . . 77
4.3 Location Based Safety . . . 77
4.4 Predicting Safety . . . 81
4.5 Personalized, Context-Aware Safety . . . 82
4.5.1 Personalized User Safety . . . 83
4.5.2 iSafe . . . 86
4.5.3 Analysis . . . 89
4.5.4 Attacks and Defenses . . . 90
4.6 Geosocial Network Extensions . . . 91
4.6.1 Crime vs. Geosocial Activity Dependencies . . . 93
4.6.2 Geosocial iSafe . . . 96
4.7 iSafe Implementation . . . 96
4.7.1 Browser Plugin . . . 97
4.7.2 Mobile iSafe. . . . 99
4.8 Experimental Evaluation . . . 100
4.8.1 Browser Plugin Performance . . . 100
4.8.2 Forecasting Accuracy . . . 101
4.8.3 Yelp Safety Profiles . . . 103
4.8.4 Android iSafe Evaluation . . . 105

5. FILTERING FAKE INFORMATION IN LOCATION BASED SERVICES 108
5.1 Introduction . . . 108
5.2 System Model . . . 110
5.2.1 Yelp Data . . . 111
5.2.2 Yelp Events . . . 114
5.2.3 Yelp Event Collection . . . 114
5.2.4 Ground Truth Data Collection . . . 115
5.3 User and Venue Analysis . . . 116
5.4 Detecting Review Campaigns . . . 121
5.4.1 Review Spikes . . . 122
5.4.2 SpiDeR . . . 123
5.4.3 Yelp Events = Review Campaigns? . . . 126
5.5 Experimental Evaluation . . . 128
5.5.1 Spike Detection Evaluation . . . 130
5.5.2 An Analysis of Yelp Events . . . 132
5.5.3 SpiDeR Evaluation . . . 138
5.5.4 Conclusions and Limitations . . . 141

6. CONCLUSIONS . . . 142
6.1 Summary . . . 142
6.1.1 Future Directions . . . 143

BIBLIOGRAPHY . . . 147

VITA . . . 159

LIST OF FIGURES

FIGURE PAGE

2.1 Minimum Bounding Rectangle approximation. 12

2.2 R-Tree example. 13

3.1 Applications of an *spsjoin* operation 24

3.2 SpSJoin System Architecture 26

3.3 Example of a *Spatial Similarity Join*. Table PHY-YP contains the join result. 26

3.4 Data flow of a MapReduce job. 29

3.5 Dataset clustering. Clusters C_i are formed after Spatial Filtering phase. 31

3.6 Example workflow for SpSJoin. 32

3.7 Data visualization of joined records. 34

3.8 Candidates for record r_1 "John C". Best match is s_3 39

3.9 $Girf_t$ values for the top ten terms found in Miami 41

3.10 MapReduce: Spatial Partitioner algorithm data flow 47

3.11 MapReduce: Local Join algorithm data flow. 49

3.12 Running time and Relative speedup for SpsJBoxSort and SpsJRTree . . 54

3.13 ROC Curves: True positive - False positive rates 56

3.14 ROC Curves: True negative - False negative rates 56

4.1 Miami venue stats: Distribution of number of reviews per venue. 71

4.2 Miami venue stats: Distribution of venue ratings. 72

4.3 Distribution of number of crime events per type of crime. Outcome of DT classifier. 74

4.4 Miami-Dade county: geographical distribution of population. Polygons represent Census Block Groups. 75

4.5 Three day evolution of the number of crimes reported within one Miami-Dade block. 77

4.6 Safety index illustration for the Miami-Dade county: $SI(B, \Delta T)$ values are mapped into color-coded "safety levels". 80

4.7 Relation between venue ratings and the crime index (CI) levels of their location. . . . 92

4.8 Relation between the number of reviews received by a venue and the crime index (CI) level of its block. . . . 92

4.9 Number of rapes per number of venue's reviews. Locals and visitors. . . . 95

4.10 Number of larcenies/thefts per number of venue's reviews. . . . 95

4.11 Snapshot of iSafe's plugin functionality for a Yelp venue. . . . 97

4.12 Snapshots of iSafe on Android. . . . 98

4.13 iSafe browser plugin overhead: Collecting reviews from venues, as a function of the number of reviews. . . . 100

4.14 Crime Forecasting Experiments in Miami-Dade . . . 101

4.15 Crime Forecasting Experiments in Miami-Dade . . . 102

4.16 Distribution of block crime index values in the Miami-Dade county. . . . 103

4.17 Distribution of safety index values of Yelp users. . . . 104

4.18 SI value of a Miami-Dade block and the average of SP values of Yelp users that visited the block w.r.t time. . . . 105

4.19 Android iSafe overhead. . . . 106

5.1 Crawler architecture. . . . 111

5.2 Yelp user stats. . . . 113

5.3 Yelp filtered reviews stats. . . . 115

5.4 Statistics of User Reviews . . . 117

5.5 Geographic distribution of venues with more than 4 Yelp reviews, in Miami-Dade county, FL. . . . 118

5.6 Visualization of the timelines of a sample set of users plotted against the review rating they assigned, since they became yelpers. . . . 120

5.7 Venues timeline . . . 122

5.8 The timeline of "Pink Taco 2" (Los Angeles) and of the Yelp event for this venue. Note the correlation between the two. . . . 125

5.9 Snapshot of WatchYT 's plugin functionality for the venue "Ike's Place". 129

5.10 WatchYT overheads . . . 130

5.11 Performance of Outlier Detection Techiniques . . . 131

5.12 Yelp events: Spike count as a function of ΔT. . . . 132

5.13 Yelp events: Distribution of the immediate impact of Yelp events on the venues' ratings. . . . 133

5.14 Yelp events. Distribution of the improvement due to events. . . . 134

5.15 Yelp events. Distribution of the improvement with a random date. . . . 134

5.16 Dependency between the short term rating change of venues due to events and their number of reviews. Importance given by standardized residuals. . . . 135

5.17 Yelp events: Distribution of the improvement due to events . . . 136

5.18 Dependency between the long term rating change of venues due to events and their number of reviews. . . . 137

5.19 SpiDeR output. Zoom-in of Figure 5.11a. . . . 138

5.20 Distribution of the number of campaigns in which users participated, including Yelp events. . . . 139

5.21 Distribution of the number of campaigns in which the fake users participated. . . . 139

5.22 Distribution of the amplitude of spikes detected by SpiDeR when considering only Yelp non-filtered reviews and when considering also filtered reviews. . . . 140

CHAPTER 1

INTRODUCTION

1.1 Motivation

The popularity of *Location Based Services* (LBSs), in particular *Geosocial Networks* (GSNs), has grown at unprecendented levels as they embrace mobile technology for user interaction and geolocation capabilities. According to [Zic12], the overall proportion of American adults that use location based services has almost doubled from 2011 to 2012 and billions of geographic locations have been generated through GSNs [ASE13]. By hosting millions of users and generating overwhelming amounts of geographic data, LBSs have raised new challenges at different levels. In this dissertation, we identify three problems found in LBSs that deal with spatial and nonspatial data. First, we address the problem of ensuring accuracy in geolocation data. Second, we tackle privacy issues in LBSs in a safety-awareness application. Third, we address correctness of data in review based LBSs. Given that geolocation data contain spatial and nonspatial attributes, we argue that combining both types of attributes into mixed techniques provides robust and effective solutions for these three aforementioned problems.

In order to understand our motivation, we describe the problems below and we briefly explain why current solutions do not work.

- **Ensuring Data Accuracy**. In order to ensure effective service, an LBS provider needs to supply accurate location data. To this end, the LBS provider relies on spatial database systems that offer different types of queries (e.g. find the nearest hotel to a given location). However, since geolocation data come from different sources, the accuracy might be very low and it may affect trust and confidence of users e.g. an inaccurate query, when the a user tries to

find a home, might result in a high crime area. One way to improve the accuracy of the location data in a given geographic dataset is to leverage another, more accurate, geographic dataset, and find the corresponding real world entities in the more accurate one. This requires a database join operation that finds similar pairs from one or more geographic datasets. Given that geographic datasets in LBSs contain both location and textual data, one may use either similarity joins or spatial joins. In a similarity join [AGK, XWLY, BMS07, VCL10], records are matched based on their textual component only if they satisfy a similarity threshold. This join operation is commonly used for duplication detection, knowledge discovery and record linkage. In a spatial join [JS], records are matched based on their spatial component e.g. geographic coordinates, polygons, etc. The join constraint may be polygon overlap or closeness between points using a distance function.

However, direct application of any these joins are likely to fail. For instance, finding a pair of similar records in two geographic datasets requires the textual attributes to be similar and the location of records to be close: a match for "John Doe" in Miami cannot be "John Doe" in New York. In this case, a similarity join would match both "John Doe"s, ignoring completely the geographic information. On the other hand, a spatial join using a distance function constraint, would match "John Doe" in Miami with its nearest records in Miami, but not necessarily another "John Doe" since it ignores the textual attribute of the data.

Therefore, a join operation that leverages both types of attributes seems more appropiate for finding real world entities in geographic datasets. Furthermore, since this operation requires the combination of large geographic datasets, an efficient and scalable solution that leverages parallel computing is essential,

specially with the ever increasing costs of *Infrastructure As A Service* (IAAS) environments [AWS].

- **Ensuring Privacy In Safety Awareness Applications**. LBSs, in particular GSNs, have attractive features and support mobile capabilities that allow users to share locations and interact [New12]. Specifically, a system where users are seamlessly made aware of their safety in a personalized and private manner can be used to address an important issue that impacts people's lives: their safety.

  In order to achieve this, we need to properly understand and define safety. While safety is naturally location dependent, it is also inherently volatile. It not only exhibits temporal patterns (e.g., function of the season, day of week or time of day) but also depends on the current context (e.g., people present, their profile and behavior). Furthermore, since safety computations require the use of personal and location data, careless use of the information exposes users to significant risks, as they may be traced or spotted in places that they do not wish to reveal.

  Attempts to make people safety-aware include the use of social media as a means to distribute information about unreported crimes [FAdO+10], or web based applications for visualizing unsafe areas [Cri, Gua]. However, these solutions do not model safety and they are unable to integrate its use in the everyday life of people. Furthermore, these solutions do not handle important problems found in LBSs: dealing with inaccurate location data and fake information that greatly impact the ultimate goal of the applications.

- **Ensuring Data Correctness**. Review centered LBSs such as Yelp [Yel] and to some extent TripAdvisor [Tri], host tens of millions of reviews and attract

tens of millions of monthly visitors [Lyn11, Coc11]. Even though the review writing process is not rewarded financially, there exists a direct relationship between reviews and financial gain: Anderson and Magruber [AM12] show that in Yelp, an extra half-star rating causes restaurants to sell out 19 percentage points (49%) more frequently. Thus, the popularity and impact of these LBSs makes malicious behavior, in the form of fraudulent reviews, a threat not only to their credibility but also to the quality of life of their users.

Although previous work on TripAdvisor [YG09, OCCH11] and fake review detection in Amazon [JL08, JLL10, LNJ+10, MLG12] have shown promising results, they cannot be applied directly to Yelp as both, TripAdvisor and Amazon lack geographic location and social networking information. Furthermore, from the Online Social Networks (OSN) perspective, "sybil" and "spam" detection have been studied, but again these solutions cannot be directly used in GSNs since their initial conditions may not hold e.g. dealing with geolocation data. Therefore, ensuring data correctness in review centered LBSs implies detecting and filtering fake review information, as they have direct impact not only in the LBS business but also in other dependent applications that rely on the data, e.g. safety awareness applications.

1.2 Research Approach

With this motivation in mind, and given that geographic data is the central part of LBSs and GSNs, this dissertation addresses the aforementioned problems at three different levels.

First, to ensure data accuracy, we argue that the heterogeneity of the attributes (spatial and nonspatial) of the data used in LBSs (GSNs) has properties that can be leveraged to improve data preprocessing. We explore techniques that use the com-

bination of spatial and nonspatial data in order to solve a very important problem that has received much less attention: the spatial set-similarity join. We develop similarity measurements that model the degree of similarity between objects in a geospatial database. Our techniques involve hybrid textual and spatial data structures that allow our algorithms to prune unnecessary search paths. Furthermore, we use parallel algorithms to tackle the scalability problems when handling large databases of geospatial data.

Second, to address privacy concerns found in LBSs, We introduce iSafe, a platform that enables participating users to gauge their safety in real time. We believe that there exist relations between the crime level at a location and the quality and quantity of GSN information at that location. We explore this hypothesis using statistical tests, e.g., the χ^2 test. In order to address privacy concerns, iSafe employs a distributed algorithm to compute safety values in a private manner without compromising safety user data. Our techniques use secret sharing in a participatory sensing platform.

Finally, to ensure data correcness, we propose to detect fraudulent information in review based GSNs. We explore a combination of statistical tools (e.g., outlier detection) to identify abnormal review behaviors. We focus our work in fake review campaigns that seem to affect the immediate impact on venues registered in LBSs such as Yelp. By using Yelp data, we observed interesting behavior specific of Yelp (e.g. Yelp Events) and we studied the long and short term impact on venues as well.

1.3 Main Contributions

In this section, we state our main contributions for each of the problems that we tackle in this dissertation. We outline each problem and propose methods for validating our results.

1. **Supporting Spatial Set-Similarity Joins in Location Base Services [BCR11]**

   This contribution presents SpsJoin (Spatial Set-Similarity Join), a framework that allows users to perform spatial set-similarity joins efficiently on large geospatial datasets.

   - We propose a combined similarity-based approach to solve the Spatial Similarity Join problem.
   - We developed an algorithm that leverages the MapReduce parallel programming model to handle large amounts of geographical data, tackling the scalability problem.
   - We implemented the SpsJoin system, a platform for performing and analyzing results of spatial set-similarity joins on large geographical datasets.
   - We validate our results from two different perspectives: efficiency and precision of the matches found.

2. **Towards Privacy Preserving Location Based Service Applications, the *Safe Cities* Case [BRCR12, BCR$^+$13]**

   This contribution presents iSafe, a privacy preserving algorithm for computing safety snapshots of co-located mobile device as well as geosocial network users. We developed iSafe in the context of Safe Cities.

   - We propose novel approaches to defining location and user based safety metrics.
   - We propose metrics to make users of LBSs safety aware of their surroundings and provide a platform to motivate our approach.

- We investigate the relationships between crime indices at different locations and the different features in GSNs e.g. rating and number of reviews.
- We evaluate iSafe using crime and census data from the Miami-Dade (FL) county as well as data we collected from Yelp, a popular geosocial network.

3. **Filtering Fake Information in Location Based Services [BCRR13, BRC+13]**

   This contribution presents mechanisms that detect review campaigns in LBSs by identifying spikes generated by low rated reviewers. Our approach takes into account different features in the data, using outlier detection methods. We also study the impact of Yelp Events in the rating of the venues, either immediate or over long periods of time.

   - We introduce SpiDeR, an algorithm that detects review campaigns by identifying spikes generated by low rated reviewers. We implemented SpiDeR as a framework that can be extended to use other machine learning approaches.
   - We collected over a million reviews from Yelp using our own crawler mechanism. SpiDeR shows that spikes generated by low rated reviewers are frequent: we have identified hundreds of venues likely to have been the target of review campaigns.
   - We have discovered an unexpected type of review campaign: Yelp events, organized by Yelp. Yelp events are hosted by a venue and are attended only by Elite yelpers, whose reviews have a higher impact on the image of the venue.

- We implemented WatchYT, a system that extends Yelp with the SpiDeR functionality. WatchYT alerts users when browsing the Yelp pages of venues targeted by review campaigns. WatchYT consists of a browser plugin that collects reviews of venues browsed by users and reports them to a web server for further processing.

1.4 Outline of the Dissertation

Chapter 2 describes the background and related work in all of our contributions. Section 2.1 explores related work in spatial set-similarity joins. This section is divided into two subsections devoted specifically to show our investigation in each of the different types of joins: spatial joins and similarity joins. Section 2.2 shows related work in Safe Cities. We survey different topics in crime LBS applications, *Smart Cities* and participatory sensing techniques. We finish this section with crime forecasting methods. Section 2.3 shows related work in opinion detection methods and spam campaign detection. We conclude this section with sybil detection techniques applied in the context of Online Social Networks (OSN).

Chapter 3 describes our first set of contributions in spatial set-similarity joins. Section 3.2 shows the architecture of our solution. Section 3.3 defines the problem formally. Section 3.4 extends our initial definition. Section 3.5 shows our improvement over our initial version and present novel metrics to score pairs of relevant matches.Finally Section 3.6 shows the experimental evaluation of our approach in terms of performance and precision of our join.

Chapter 4 is organized as follows. Section 4.2 presents the model considered and motivates the problem of Safe Cities as an LBS application to compute safety on co-located users. It also describes the datasets and tools used in this work. Section 4.3 proposes a static, location centric safety labeling technique and Section 4.4 compares

the ability of existing forecasting tools to predict future crime and safety values. Section 4.5 introduces the concepts of personalized and context aware safety as well as the iSafe solution. Section 4.6 investigates relationships between social networks and crime levels. Section 4.7 describes the iSafe implementation and Section 4.8 presents the evaluation of our results.

Chapter 5 describes our last set of contributions by studying several features of Yelp. The section is organized as follows. Section 5.2 presents the system model as well as statistics of the data we collected from Yelp. Section 5.4.3 shows our analysis on Yelp Events. Section 5.4 describes our review campaign experiments. Section 5.3 introduces the notion of user timelines and proposes a user rating definition. Section 5.4 defines venue timelines and presents the SpiDeR algorithm. Finally, Section 5.5 evaluates the performance of the solutions that we propose.

Chapter 6 concludes the dissertation and summarizes the future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter we describe the most relevant literature. We first show existing approaches on similarity joins and spatial joins that will ultimately serve as our baseline for *spatial set similarity* joins. Then, we survey existing work in spatio-textual query problems and their applications in LBSs. We explore current and seminal work in LBS applications related to location aware safety which motivates privacy and participation using GSN providers. Finally, we describe current and related problems in filtering fake information in GSNs.

2.1 Spatial Set-Similarity Joins

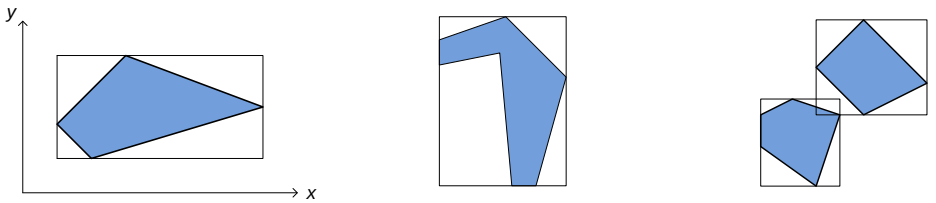
2.1.1 Assumptions

This sections present some assumptions and definitions that we make to better understand the related work that we present. A geographic database is a database of objects represented logically by records. Each object contains attributes that describe its characteristics and they can be numerical, textual and geographic. We assume that all objects have at least a geographic attribute. A geographic attribute represents a characteristic that describes the object in the space. For instance, latitude and longitude coordinates represent the position of the object on Earth. Another example of geographic characteristic is a polygon. It may represent the boundaries of the geographic object such as parcels, water bodies, landmarks, etc. To handle spatial queries efficiently, geographic databases index the data using the spatial component i.e. the geographic attribute.

2.1.2 Spatial Joins

Many algorithms have been proposed to tackle the problem of spatial join queries. In particular, Jacox et.al. [JS] provides an extensive survey on this topic, along with efficient methods to compute spatial joins in parallel. The main idea of a spatial join is to combine geographic objects from one dataset with another dataset so that a spatial constraint is satified. The most common spatial constraint used in spatial joins is the intersection. For instance, let R be a geographic database of water bodies and S a geographic database of bridges. The spatial join of R and S returns the bridges that pass through the water bodies. There are several methods for computing spatial joins. The most common approach to tackle a spatial join is to first preprocess the data by approximating the geographic extent of the objects using *Minimum Bounding Rectangles*(MBR) [JS]. Figure 2.1 shows examples of approximations to MBRs. Figure 2.1a shows how to approximate the object with an MBR in $\mathbb{R}^2$. Figure 2.1b shows how such an approximation may waste a lot of space since the object is significantly smaller than its MBR. Figure 2.1c shows that in an intersection even though the MBRs intersect, the actual objects might not. Once the objects are approximated, their MBRs are intersected in a filtering stage. Then, the set of candidate intersections undergo a refinement stage where the actual dimensions of the geographical extents of the objects are obtained, and the result set is built, removing false positives.

There are variations in the way these two stages work. When the objects are not indexed, algorithms use a nested loop approach [ME92]. These methods work well when the size of the datasets to be joined is small and they can work in memory. However, when the datasets are large, a nested loop approach is not recomended, as it implies a quadratic running time for processing. There are other methods that



(a) Approximation in $\mathbb{R}^2$ (b) Waste of space in approximation. (c) No intersection.

Figure 2.1: Minimum Bounding Rectangle approximation.

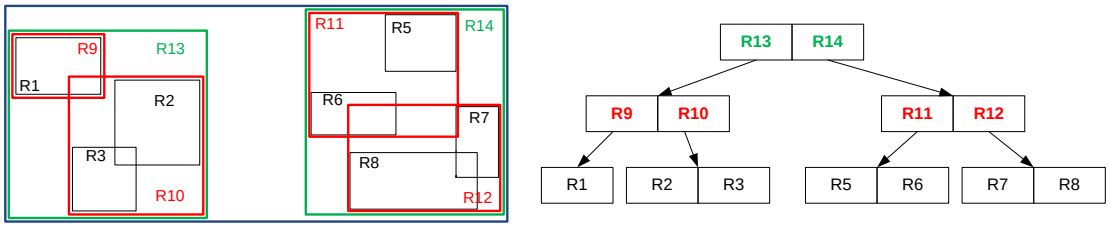
also work well in memory. The plane sweep approach [APR+98] works similar to a plane sweep technique in computational geometry in database objects.

As we noted above, objects can be indexed using their spatial component. Samet et.al. [Sam90] provides a survey of the most important data structures for processing spatial data using hierarchical structures (e.g. trees). The "flagship" spatial data structure used in databases is the *r-tree* [Gut84]. R-trees are balanced multiway trees that organize spatial objects using their MBRs. The hierarchy is stablished by MBR overlaping at each of the levels and r-trees are specially suitable to handle nearest neighbor queries and rectangle overlap. Figure 2.2 shows an example of this data structure. As we can see, in Figure 2.2a the space is partitioned based on the rectangles. Note that there may be overlap. Figure 2.2b shows how the data structure is organized in memory.

R-tree's maintenance and query processing work similar to *B-trees* [Com79] as they keep balance by using node splitting when node overflows occur. Also, r-trees are suitable for database management since they can keep actual spatial objects in disk while a node manager is in charge of the memory management using a hash table in memory to map memory-disk allocations.

R-trees have been used extensively for spatial joins. Huang et.al. [HJR97] propose a method for computing spatial joins using r-trees with global optimizations.

Their approach requires both datasets to be indexed in the database management system. However, there are other methods that do not need the datasets to be indexed. When one dataset is indexed, efficient techniques exist to bulk-load the unindexed dataset and perform a regular spatial join using r-trees. The main advantage is that the index of previously unindexed dataset is now a by-product of the process and can be used for further query processing. However, when there is no need for keeping an index, Lo et.al [LR94] propose spatial joins using *seed trees*.



(a) Spatial partitioning. (b) Actual structure.

Figure 2.2: R-Tree example.

We emphasize that these methods only take into account the spatial components of the data. Therefore, they are unlikely to be used for computing Spsjoin queries. However, we adapted spatial parallel processing techniques [PD96], to scale our solution in a MapReduce model. Processing large geographical datasets in parallel using MapReduce have been studied previously in [CSHR09] and [ZHLW09]. Cary et.al [CSHR09] use a z-order space filing curve approach for partitioning the dataset into groups based on the spatial attribute. Their solution achieves scalability by relying on random sampling of a set of points that are sorted based on z-order. These points define unidimensional intervals(partitions) where points in the dataset

are mapped. However, if the data is skewed, the claim that z-order generates almost uniformly-sized partitions is no longer true. Even though in practice, minimal variations in sizes of the partitions are acceptable, for certain datasets this may cause scalability issues.

2.1.3 Set Similarity Joins

Set similarity join queries have been widely studied in [AGK], [XWLY], [BMS07] and [KS98]. Arasu et. al. [AGK] propose efficient methods for computing similarity joins using filtering techniques based on threshold. This work derived interesting results that were later leveraged in [BMS07] and [XWLY] to describe new applications for near duplicate detection of web pages and documents in the web. A similarity join requires a similarity metric that measures the relatedness of the objects being considered. The general constraint that defines a similarity metric is given in Equation 2.1. This constraint stablishes that all pairs of objects (r, s) whose similarity measurement is greater than or equal to τ should be part of the join.

$$sim(r, s) \geq \tau \tag{2.1}$$

Tan et.al [TSK05b] present popular similarity metrics used in data mining, specially when working in clustering algorithms. Jaccard coefficient, *cosine* similarity and *overlap* similarity are used mostly in set-similarity joins. The idea is to take the attribute or set of attributes that the user deems as descriptors of the objects and create sets for further comparison. For instance, let r be an object that consists of a textual attribute such as its name "John Doe". This name may be transformed into a set in two different ways: by *tokenizing* the string using words or q-grams. Therefore, in the words case, "John Doe" is the set $\{$"$John''$, "$Doe''\}$, where as in the 3-grams case, the set is $\{$"Joh'', "ohn'', "hn'', "nD'', "Do'', "$Doe''\}$.

While current techniques on similarity joins are extremely powerful to prune candidate pairs with low similarity, they work well only for large values of thresholds, e.g. 0.9 or 0.8. For medium threshold values, the running time degenerates to quadratic computation and other techniques that rely on *Locality Sensitive Hashing* (LSH) are employed [AI08], under probabilistic guarantees. LSH techniques are also used when the dimensionality of the data is too large. This is true specially when objects are transformed into large sets of elements that need similarity computations very fast.

All of these methods work well in main memory, being *ppjoin* the fastest algorithm, using a suite of filtering strategies to prune candidates that may not satisfy the similarity threshold. However, with the explosion of the data, in-memory similarity joins are not suitable as they may experience scalability problems. Vernica et. al. [VCL10] propose scalable techniques for joining datasets using MapReduce. Their work rely on mapping of records using the prefix of the textual attributes under a global ordering. Then, in the reducers the algorithm perform local joins using either nested loop approach or *ppjoin*, proposed in *xiao:www*.

2.1.4 Spatio-Textual Queries

LBSs have adapted spatial and textual techniques to answer specialized queries that involve both, spatial and textual information. The main advantage here is that, when the spatial component has been already used to prune unnecessary data, the textual component helps further to prune data and a much more efficient hybrid algorithm results. However, this approach has a cost: it requeries storing of additional data by augmenting the data structures. It also poses additional scalability problems when the size of the data is extremely large.

The work of Hariharan et.al [HHLM07] shows how to build hybrid r-trees with augmented textual data at node level. However, their solution suffers from a phenomenon called the "keyword spread" problem. They propose handling the issue by collapsing pages of nodes with the same keyword at the same level, but it is not clear from the paper how this actually works. Alsubaiee et. al. [ABL10] provide a detailed description of a hybrid index for answering approximate spatio-textual queries. Their solution use inverted indexes at different levels of an r-tree, depending on cost functions that yield effective pruning depending on the spatial distribution of the data e.g. for sparse areas, spatial pruning is more effective than keyword prunning at high levels of the tree. However, even though this work claims that this type of index is space efficient, pathological cases may lead to the keyword spread problem as well. Also, they do not seem to apply space efficient filtering techniques as proposed by [XWLY].

2.2 Location Based Services Applications: Safe Cities Approach

LBS applications rely on geolocation data to provide services. This dissertation studies *Safe Cities* as a platform to motivate and to address privacy concerns found in LBSs. To this end, in this Section we studied related work in *Smart Cities* which are technological platforms in cities to reduce expenditures. We explore all related work that we use in our research, from machine learning techniques and crime prediction methods, to participatory sensing.

2.2.1 Safe Cities

Smart cities have been the focus of recent efforts at IBM [IBM] and several academic research groups at MIT [Lab] and UCLA [UCL]. Caragliu et. al. [CDBN09] present

a study on the factors that determine the performance of a "smart city". They focus specifically on European cities by analyzing urban environments, levels of education and different accessibility modalities that are positively correlated with urban wealth. Since one important aspect of smart cities is safety, Patton [Pat10] emphasizes the use of audio sensors and cameras that allow authorities to quickly respond in an emergency event without receiving a 911 call. We note that we consider a different angle: making users aware of their surroundings.

Furtado et. al. [FAdO+10] propose the use of social media in a collaborative effort to inform people about crime events that are not reported to police. Their wiki website spots areas on the map where participant users have reported crime events. Police departments also release tools to make citizens aware of their safety, e.g., the Miami-Dade police department, deployed an web application [Dep] that identifies crime areas based on current crime reports. We note however that our solution seamlessly integrates context and time sensitive safety metrics into the everyday user experience.

2.2.2 Participatory Sensing

Participatory sensing is receiving increasing attention due to the popularity of mobile devices. The multimodal sensing capabilities of devices enable a broad range of applications that leverage collected data from participants, sensed from their surroundings. Estrin [Est10] discuss advantages of participatory sensing in health and transportation and provide insights on the architecture of participatory sensing applications. Thiagarajan et. al. [TBGE10] propose cooperative transit tracking using mobile phones. Privacy becomes a serious concern when the user personal information may be compromised. Christin et. al. [CRKH11] present a survey on the efforts made to preserve privacy in participatory sensing systems. In contrast,

our work does not collect user information, but instead allows devices to aggregate information collected from co-located users without learning personal information.

Dynamic safety practices leveraging social networks and GPS mobile phones have been introduced in [YBL+08] to create a system for personalized safety awareness. The system exploits sensors available in mobile phones to enhance the personal safety of users by aggregating community. Our work is different in that we predict future crime levels, define a safety index that includes the impact of crimes on locations and on the profiles of users and propose a distributed algorithm that privately aggregates safety indexes of co-located users.

2.2.3 Crime Prediction

The problem of crime prediction has been explored in several contexts. Hotspot mapping [CTU08] is a popular analytical technique used by law enforcement agencies to identify future patterns in concentrated crime areas. Different methods and techniques have been analyzed to review the utility of hotspot mapping in [ECC+05], [CR05], [Jef99], [CRS02]. Hot spot analysis however, often lacks a systematic approach, as it depends on human intuition and visual inspection.

A variety of univariate and multivariate methods have been used to predict crime. Univariate methods range from simple random walk [BSV98] to more sophisticated models like exponential smoothing. While exponential smoothing offers greater accuracy to forecast "small to medium-level" changes in crime [GO01], we have shown that ARIMA and ANN models outperformed it on our data. In [EA07], Ediger et al. show the effectiveness and reliability of ARIMA and SARIMA models in predicting the total primary energy demand of Turkey from 2005 to 2020. Olligschlaeger [Oll97] showed that ANNs were able to predict drug markets. We note that the goal of our work is not intrinsically crime forecasting. Instead, we incorporate crime forecasting

techniques into our safety metrics, in an attempt to provide to participating users a dynamic framework for safety awareness.

2.3 Fake Review Detection and Opinion Spam in Location Based Services

Data in LBSs should be correct and consistent since it forms the central part of the operations in this type of systems. In review-centered LBSs the data come in the form of reviews with different characteristics, venue or product information and user profiles if the LBS has GSN capabilities. The correctness of the data implies that fake information may have a significant impact in LBSs as this may hurt the credibility and confidence in the use of the system.

In this section, we explore current literature available that tackles opinion and fake review detection systems. These works rely on different techniques, such as machine learning and statistical analysis, but also in natural language processing mechanisms to process text data. We also review some of the web crawling techniques and data collection processes that we use heavily in our research.

Jindal and Liu [JL08] introduced the problem of detecting opinion spam in the context of product reviews. The techniques proposed in the context of Amazon reviews, include detecting spam, duplicate or plagiarized reviews and outlier reviews. Jindal et al. [JLL10] extend this work to identify unusual review patterns which can represent suspicious behaviors of reviewers. They formulate the problem as finding unexpected domain independent rules and also test their solution on Amazon reviews. In the context of review spam, Lim et al. [LNJ+10] propose techniques that determine a user's deviation from the behavior of other users reviewing similar products. Our work complements this research. We focus on reviews written in geosocial networks, where we further rely on the location of reviewers and reviewed

venues as well as social dimensions. This allows us to discover new relations and exploit them to detect not only fake reviews and reviewers but also venues that are frequent targets of such attacks.

Of notable importance is the work of Ott et al. [OCCH11] who created a database of fake hotel reviews in TripAdvisor, then integrated work from psychology and computational linguistics to develop and compare three approaches to detecting deceptive opinion spam. Unlike this work, which focuses on the text of reviews, our research relies on social and geographic dimensions to address the same issue in Yelp: Unlike TripAdvisor, Yelp provides us with access to the location and friends of reviewers.

Li et al. [LHYZ11] and Ntoulas et al. [NNMF06] rely on the review content to detect review spam. Li et al. [LHYZ11] exploit machine learning methods in their product review mining system. Ntoulas et al. [NNMF06] propose several heuristic methods for detecting content based spam and combine the most effective ones to improve results. Our work differs through its emphasis on relationship among reviewers, friends and ratings in the context of Yelp's spatial and temporal dimensions.

Mukherjee et al. [MLG12] focus on fake reviewer groups, reviewers who work collaboratively to write fake reviews. They propose the use of a frequent itemset mining method to find a set of candidate groups, then used several behavioral models derived from the collusion phenomenon among fake reviewers and relation models based on the relationships among groups, individual reviewers, and products they reviewed to detect fake reviewer groups. We consider a different adversarial model, where the membership of reviewer groups inherently changes due to the nature of the recruitment process (i.e., Amazon Mechanical Turks).

Gao et al. [GHW+10] target asynchronous wall messages to detect and characterize spam campaigns. They model each wall post as a pair of text description and

URL and apply semantic similarity metrics to identify large subgraphs representing potential social spam campaigns and later incorporate threshold based techniques for spam detection. Instead, we focus on temporal and geosocial review context, the where reviewer activity and behavioral pattern are of significant importance. Feng et al. [FXGC12] relies on the J-shaped distributions of review ratings received by most venues to identify venues that receive too many 5 star reviews from single-time users.

Wang et al. [WXLY11] introduce the concept of heterogeneous review graphs and iterative methods exploring relationship among reviewers, reviews and stores to detect spammers. While we also consider social relations among reviewers we differ on our focus on temporal and spatial dimensions.

2.3.1 Sybil detection

Detecting review campaigns can benefit from existing sybil detection techniques. Of particular relevance is DSybil, the work of Yu et al. [YSK+09] that applies in the context of the news voting social network Digg. The semantics of reviews for venues of Yelp however differ fundamentally from news. Venues change in time, and reviews are always welcomed in expressing the time fluctuations of a venue's quality. In a sense, the performance of a venue each day can be viewed as a "news item". Tran et al. [TMLS09] proposed SumUp, a trust based sybil defense mechanism that uses "adaptive vote flow aggregation" to limit the number of fake feedback provided by an adversary to the number of attack edges in the trust network - that is, the number of bi-directional trust edges the attacker is able to establish to other users. WatchYT can benefit from the techniques of SumUp, or even be used in conjunction with it. However, we note that previous work [BSBK09, BMBR11] has shown that Facebook users frequently accept friend invitations from complete strangers.

2.3.2 Web Crawling and Data Collection Process

In this section we survey the most important research in web crawling techniques.

A web crawler is a system whose main purpose is to bulk-downloading large amounts of web pages. An interesting survey can be found in [ON10]. This survey handles efficient techniques, either static or dynamic, to download web pages with different objectives. The main goal of a web crawler is to produce the collection of web pages that are indexed by a search engine and therefore, it relies heavily on link discovery within the web pages crawled. The work of Aggarwal et.al. [AAGY01] proposes the concept of *intelligent* crawling which learns characteristics of the link structure found in web pages. The idea is to use the inlink attributes to determine the probability that a candidate is useful for crawling. This has tremendous impact in the performance of the crawler, since repeated web pages cause the crawler to slow down its execution.

From the theoretical point of view, Baeza-Yates et.al.[BYC07] studied several probabilistic models that predict how deep users of the internet go while exploring Web sites. They propose the *back one level at a time*, *back to the first* level and *back to any previous* level models and compare their results. We note that our crawling techniques use the *one level at a time* type of model, since we do not explore explore more than two hops deep in the Yelp data. In Section 5 we show our web crawler architecture used to get crawl the data that we leverage in this dissertation.

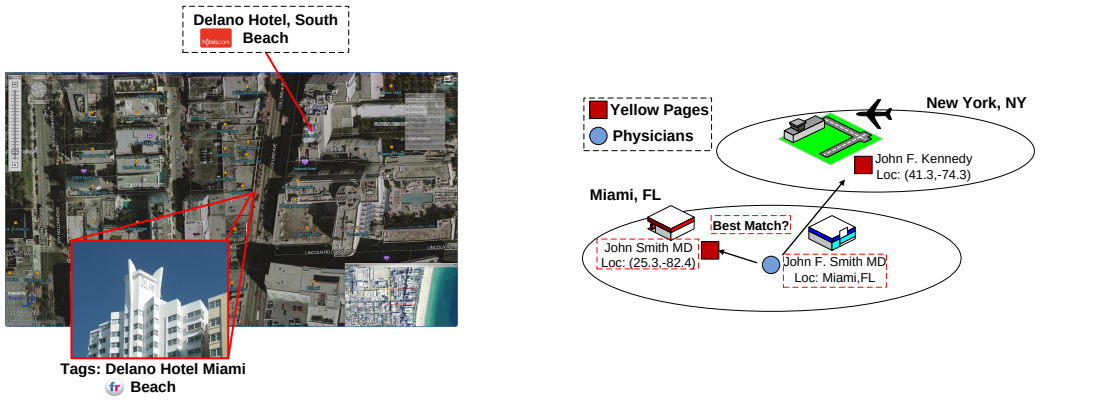
CHAPTER 3

SUPPORTING SPATIAL SET-SIMILARITY JOINS IN LOCATION BASED SERVICES

In this chapter, we present *SpsJoin*, a framework for computing spatial set-similarity joins. We first show our motivation through specific applications in LBSs. This framework is used primarily in our flagship platform Terrafly [FIU] for geolocation enhancement and knowledge discovery by leveraging the richness of our data repository. Then, we define the problem formally as general case where the user is agnostic of the similarity parameters. Finally, we extend our solution by using a novel content similarity function that ranks pairs of best matches. We conclude with a set of experiments that validate our approach using real world datasets.

3.1 Introduction

In modern geographical databases, records contain textual and spatial attributes to describe characteristics and location of real-world entities. When the location of the records has low accuracy, e.g. geolocated at the center of the city, their location may be enhanced by finding their *most similar* records in another database, known to have high location precision. For instance, Figure 3.1b shows sample records of *Physicians* database, geolocated at city center level precision and *Yellow Pages* database with high geolocation precision. Intuitively, the most similar object of physician "*John F. Smith MD*" is "*John Smith MD*" in Yellow Pages, since both names are very similar and geographically closer. The same is true for a Flickr [FLI] dataset that might be enhanced by finding similar records in a dataset of Hotels, as shown in Figure 3.1a. Therefore, finding the most similar pairs between two



(a) Flickr *spsjoin* Hotels (b) Best match for physician "*John F. Smith*"

Figure 3.1: Applications of an *spsjoin* operation

geographical databases requires a composite join operation that considers both types of attributes, textual and spatial.

Such type of join, namely *Spatial Set Similarity join*, has received much less attention in the research community than individual joins on either textual or spatial attributes. In the textual case, the degree of resemblance in a *similarity join* [AGK, XWLY] is measured by a similarity function, e.g. Jaccard coefficient or Levenshtein distance, and pairs that satisfy a user-defined similarity threshold are included in the output. Recently, parallel processing with MapReduce, a parallel programming model proposed by Google [DG], has been explored to tackle the scalability problem of this type of joins [VCL10]. In the spatial case, a *spatial join* [JS] between two geographical datasets matches records based on their spatial attributes. The spatial relation may be expressed in several ways, e.g. distance threshold or polygon overlap.

Direct application of either spatial join or similarity join techniques to solve the spatial similarity join problem has the disadvantage of potentially generating lots of pairs that do not satisfy the composite constraint; for example, in Figure 3.1b several similar physician names and yellow page contact persons may be located far away from each other, e.g. "*John F. Kennedy*" in New York, but we are interested only

in the geographically nearest pair. Also, when a threshold is predefined for either similarity or spatial joins, some records may not find their most similar pair when they do not satisfy the threshold. It is then up to the user to define an appropriate distance or similarity threshold even when there is no knowledge of the precision and quality of the data. In addition, as geolocation data is rapidly increasing in databases, scalability in processing spatial similarity joins is a top concern.

Spatial set-similarity joins have generally the same applications as similarity joins, including data cleansing and record linkage. In addition to geolocation enhancement, this join might be used in disaster management applications, e.g. joining 911 call records with Nationwide cadastre and White Pages databases to pinpoint massive emergency events. Geolocation enhancement has several implications in LBS. As we will show in subsequent chapters, the accuracy of the data is of vital importance for LBS that provide safety of co-located users, as crime index and safety metrics depend on geolocation data.

3.2 System Architecture

The SpSJoin system is divided into four components. Figu-re 3.2 shows the proposed architecture for our system. The *Data Repository* stores the geographical databases used by the system and supports data persistency required by the interacting modules. The *Spatial Similarity Join* module performs the join and returns the result set that is indexed by the *Query Processing* module. Finally, the *Data Visualization* module presents an interface to the user for displaying and analyzing the join results.

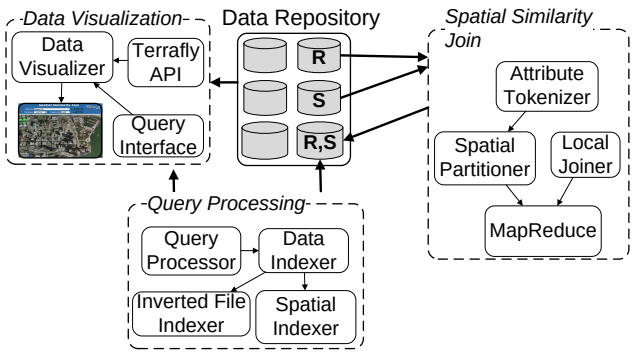


Figure 3.2: SpSJoin System Architecture

3.2.1 Data Repository

The data repository contains several geographic datasets used in different GIS applications. Data comes from di-fferent sources, including the Internet and public and private sources, that may or may not need additional geographic location processing. Examples of datasets found in the repository include *Hotels*, *Crime Data*,*Places and Landmarks*, etc., all of them containing different attributes and geographic location. Figure 3.3 shows the *Physicians* and *Yellow Pages* datasets with some of their attributes and spatial location.

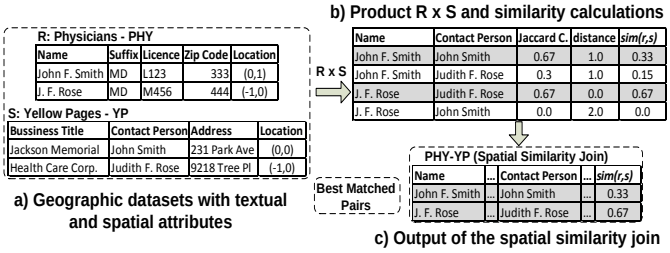


Figure 3.3: Example of a *Spatial Similarity Join*. Table PHY-YP contains the join result.

3.3 Spatial Similarity Join

Intuitively, a Spatial Similarity Join finds pairs of objects from two spatial datasets, a *target* and a *source*, in which every pair represents a match of an object in the

target with the *most related* object in the source. Relatedness between objects is modeled with a composite similarity function that combines spatial and textual attributes. For instance, in Figure 3.3b, the similarity of a pair is calculated by combining the distance of the objects with their textual similarity on *Name* attribute in Physicians and *Contact Person* in Yellow Pages. The most related pairs from the Cartesian product are the ones with the highest value given by the similarity, $sim(r, s)$, function, e.g. $\langle$"*John F. Smith*", "*John Smith*"$\rangle$ and $\langle$"*J.F. Rose*", "*Judith F. Rose*"$\rangle$. Next, we present the problem statement and describe our approach for processing spatial similarity joins efficiently.

Notation. We denote our input datasets as R (*target*) and S (*source*). Without loss of generality, records in these datasets are tuples of the form $o = \langle a, p \rangle$, where a denotes a textual attribute and p is a point in the space that denotes the location of the object o. In practice, objects may contain additional textual attributes, which we omit to simplify the explanation. MBR refers to the Minimum Bounding Rectangle that encloses a set of objects. Given two objects r and s, we refer to the function $sim_t(a_r, a_s)$ as the textual similarity between attributes a_r and a_s, and $dist(p_r, p_s)$ as the distance between points p_r and p_s. We denote $sim(r, s)$ as the composite similarity function in the problem statement.

Problem Statement

Given two datasets R and S and a composite similarity function $sim(r, s) \in [0, 1]$, that combines spatial and textual similarity, the problem of *Spatial Similarity Join* finds the set of pairs $(r, s) \in R \times S$, such that $sim(r, s) = \max_{s' \in S}\{sim(r, s')\}$. We say that s is the *most related* object of r found in S and the pair (r, s) is a *best matched pair*.

Choosing an adequate $sim(r, s)$ function is challenging since each type of attribute has its own semantics and independent similarity values. Therefore, careful analysis of the datasets is required. For example, if R and S are known to have very precise spatial attributes, then $sim(r, s)$ may give less importance to the textual attributes.

3.3.1 Tools

MapReduce

In this work, we use MapReduce [DG] to describe our algorithms. MapReduce is a popular programming paradigm that leverage distributed computing in commodity clusters. It is used primarily for data-intensive parallel applications that share no communication between nodes in the cluster. The data is partitioned and stored in a distributed file system (DFS). Each partition, called split, is processed in parallel through *map* tasks. Figure 3.4 shows how the data flows in a MapReduce job. The map tasks process and generate lists of key-value pairs (<K,V>) that are later sorted, merged and grouped by key. These groups of lists become the input of *reduce* tasks, i.e. each reducer process a single list of key-value pairs that share the key, and generate a list of new key-value pairs. This is the final stage of a MapReduce job.

A popular implementation of the MapReduce model is Hadoop [Apa12], an open source framework that allows for the distributed processing of large datasets. The model also defines special functions called *combiners*. Combiners may help in reducing the amount of data sent through the network by processing the output of the map tasks in memory before merging and shuffling. Similar to reduce tasks, combiners receive lists of key-value pairs grouped by key and return an aggregation

of the key-value lists. These aggregations are then merged and grouped by key, so that reduce tasks can process them. Combiners are useful only when the data can be aggregated and have the same signatures as reduce tasks and there is no always a guarantee that they will be executed.

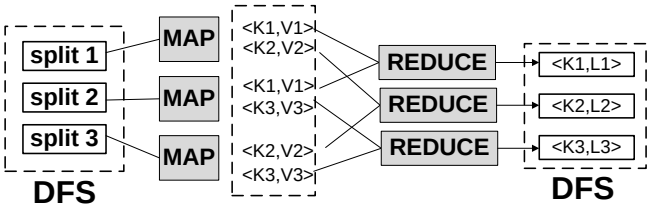


Figure 3.4: Data flow of a MapReduce job.

3.3.2 Processing Spatial Set Similarity Joins

In our approach, $sim(r, s)$ meets the criterion that similarity of pairs of proximal objects must be higher than objects located far away from each other. We defined the following similarity function

$$sim(r, s) = \frac{sim_t(a_r, a_s)}{1 + dist(p_r, p_s)} \tag{3.1}$$

Where $sim_t(a_r, a_s)$ is a textual similarity function (we used the *Jaccard* coefficient in our experiments, $sim_t(a_r, a_s) = \frac{|a_r \cap a_s|}{|a_r \cup a_s|}$) and $dist(p_r, p_s)$ is a distance function (we used *Great Circle* distance since geographical objects are located with latitude and longitude). In general, if an object r has two possible matching objects s and s' with equal similarity value (i.e. $sim(r, s) = sim(r, s')$), the pair with minimum distance is considered the better pair. In Section 3.4.2 we will define other types of similarity functions tailored for specific problems in LBS.

When processing spatial similarity joins in large datasets, scalability is a key challenge. Our algorithm leverages pa-rallel computing with MapReduce, which has proven its effec-tiveness in large-scale data intensive problems [DG].

The join process is divided into two main phases: a *Spatial Filtering* phase and an *Expansion* phase. In the Spatial Filtering phase, the entire set of records is partitioned w.r.t. their spatial attribute. The rationale is that geographically proximal object pairs are more likely to generate higher si-milarity values, using Equation 3.1. In this way, potential best matches are co-located in the same partition, filtering out pairs with low similarity value whose evaluation is not nece-ssary, e.g. far away objects do not represent the same real world entity.

Theorem 3.3.1 *Let s be a match candidate of record r. The best match s_b of record r can be found within the region limited by $e_r(r,s) = \frac{1}{sim(r,s)} - 1$*

Proof. From Eq. 3.1, the similarity of r and s is $sim(r,s)$. Let s_b the best match of r. Then, the textual similarity of r and s_b is given by $sim_t(r,s)$, which is maximum when $sim_t(a_r, a_{s_b}) = 1$, hence $sim(r,s) \leq \frac{1}{1+dist(p_r,p_{s_b})}$. It follows that $dist(p_r, p_{s_b}) \leq \frac{1}{sim(r,s)} - 1 = e_r(r,s)$. □

Since each partition may contain some local best pairs that may have globally best matches, i.e. with increased similarity value, the Expansion phase gradually expands the search space of each partition using an upper bound *Expansion Region*. Object pairs are reprocessed itera-tively on adjacent geographical regions until their similarity value cannot be improved anymore, i.e. the best pairs are found, or the expansion region covers all universe of objects. We illustrate the join execution with an example, shown in Figure 3.6, that describes the workflow of the process. We denote clusters of records as C_i, $i = \{1,2,3\}$, and sets L_i^j as local output in cluster C_i at iteration j. Final join output is denoted as L.

Spatial Filtering phase. Figure 3.6 part **(I)**. The *Spatial Partitioner* component is used for partitioning the entire set of records. It is expressed as a MapReduce job that clusters $R \cup S$ in parallel using a clustering algorithm; in our experiments,

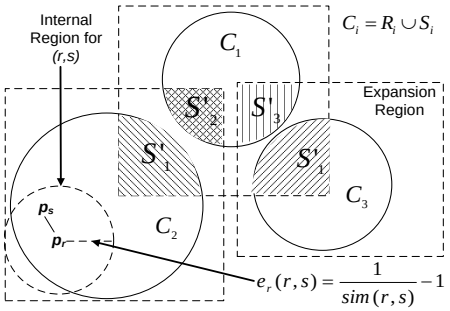


Figure 3.5: Dataset clustering. Clusters C_i are formed after Spatial Filtering phase.

we used the X-means clustering technique [PM]. Figure 3.5 shows the spatial layout of the three clusters in this example: C_1, C_2 and C_3. Note that each C_i is expressed in Figure 3.6 as $R_i \cup S_i$.

Expansion phase. Figure 3.6 part **(II)**. Each cluster C_i is processed locally in parallel using several expansion itera-tions. Each iteration of our example is described next.

Iteration 1. For each cluster C_i, the *Local Joiner* component joins R_i and S_i using a nested-loop approach; we implemented the Local Joiner using a modified version of the fuzzy join proposed in [VCL10], leveraging the MapReduce framework. Mappers tokenize textual attributes from records in $R_i \cup S_i$ and generate record projections for each token, tagged with the relation name. Reducers receive records that share the same token, sorted by relation (S_i first), and records in R_i are combined with records in S_i. To accelerate the process, records in S_i are indexed using their spatial attibute. For every record in R_i, the spatial index filters records in S_i that will not improve in the combined similarity. The combined similarity is computed for candidate pairs and the pair with the highest $sim(r, s)$ is kept. The output of the Local Joiner L_i^1 is the set of local best matched pairs found in cluster C_i.

In order to prepare for the next iteration, the input $P_i \cup S'_i$ needs to be calculated. We observed that each pair (r, s) in L_i^1 defines an *internal region*, as shown in Figure 3.5, with center p_r of r and

The union of all internal regions defines the upper bound *Expansion Region* for the cluster, in which objects from R_i may find better matches. Since the Expansion Region may overlap adjacent clusters, objects in pairs with internal regions that lie within the cluster's MBR will not find a better match and the corresponding pairs are stored in the B_i database as part of the final output. This reduces the size of the input in the next iteration. With the remaining pairs, objects in R_i are extracted and stored in P_i, which need further iterations. Finally, the system identifies the nearest cluster C_k, that overlaps the Expansion region, and stores the overlapping objects from S_k in S'_i. In Figure 3.5 for example, the nearest cluster of C_1 is C_3, so S'_1 is the set of records from S_3 in the shaded region of C_3.

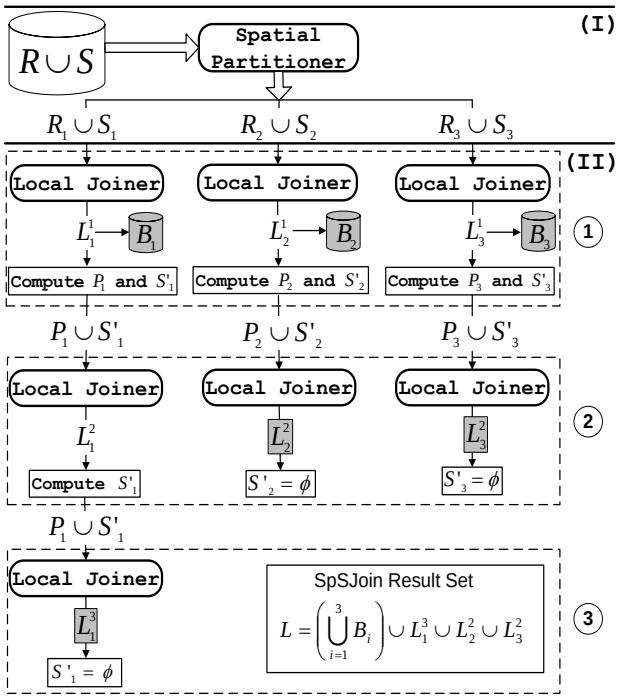


Figure 3.6: Example workflow for SpSJoin.

Iteration 2. Each Local Joiner receive $C_i = P_i \cup S'_i$ as input and joins P_i and S'_i as in the previous iteration. Ouput pairs in L_i^2 that improved their similarity are updated as the new best pair matches. If further clusters need to be explored, the next nearest cluster that overlaps the Expansion region is identified and S'_i is computed as above. Else, the local process finishes its execution. In Figure 3.5, Expansion regions for clusters C_2 and C_3 do not expand anymore so L_2^2 and L_3^2 are part of the final output. On the other hand, Expansion region of C_1 overlaps C_2 so P_1 requires further processing. S'_1 is now the set of records from S_2 in the shaded region of C_2

Iteration 3. Local Joiner is called again with the new input $C_i = P_i \cup S'_i$ and the output L_i^3 is generated. In our example, the Expansion region for cluster C_1 has no more overlapping clusters to cover, hence set L_1^3 is part of the final output. Since no clusters need further expansion, the process terminates and the join result set L is complete. Shaded blocks in Figure 3.6 form the final output of the join.

3.3.3 Query Processing

The *Query Processing* module , Figure 3.2, is used primarily by the *Data Visualizer* component to retrieve records of joined databases (generated by the *Spatial Similarity Join* module). This module executes spatial queries with non-spatial constraints posted by users for join quality inspection. Attributes in the join result are first indexed using a hybrid data structure that leverages R-trees and inverted files [CWR] by the *Data Indexer*. Second, the *Query Processor* parses a user query to identify the query window (geographi-cal region) and (optionally) non-spatial constraints, and it uses the hybrid index structure to efficiently retrieve records. For instance, in the join example of Figure 3.3, joined records of physicians with last name "*Smith*" and located in "*Miami, FL*" are displayed in Figure 3.7.

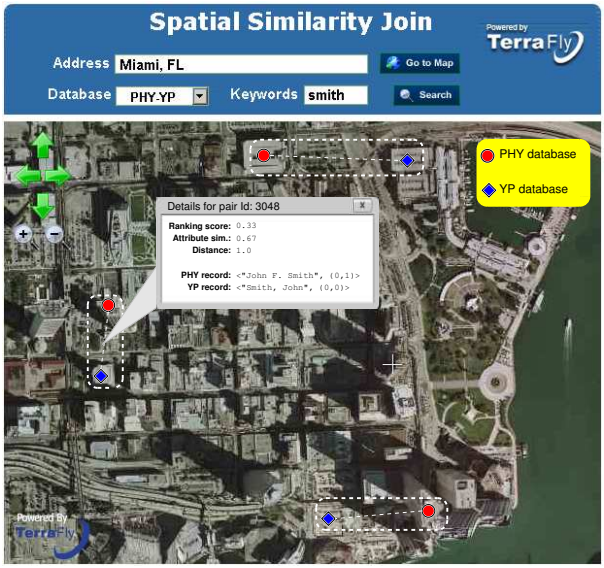


Figure 3.7: Data visualization of joined records.

3.3.4 Data Visualization

The *Data Visualization* module displays the results of spatial similarity joins on a map. Figure 3.7 shows the general user interface of the system. Aerial and satellite imagery as well as user interface widgets are provided by the TerraFly system[1] via its public API.

When the user selects a location to visualize, the currently displayed map portion determines the query window that will be submitted to the *Query Processing* module. Then, users pick a previously joined database from a database drop-down list to visualize its records. Optionally, users can include keywords in the query to locate specific objects for inspection. Records that match the query criteria are displayed as pairs, visually distinguished by circles and diamonds, united by lines and enclosed in rectangles. Users can click on individual object icons to display detailed information about the pair the object participates on.

[1]http://terrafly.fiu.edu

Table 3.1: Geographical databases of *Physicians* (PHY) and *Yellow Pages* (YP) used in experiments.

Database	Records	Joining Attributes	
		Textual	Spatial
PHY	2 millions	name	zip
YP	20 millions	contact person	location

3.3.5 Framework Demonstration

We demonstrate our system as follows. First, two real geographical databases, *Physicians* (target) and *Ye-llow Pages* (source), were joined with the SpSJoin operator. The database sizes and joining attributes are shown in Table 3.1. Objects in the databases represent real-world entities located in the United States. For example, YP entries include medical professionals of various specialities, which are expected to match with records in PHY. Jaccard coefficient and Great Circle distance were used to compute the similarity of textual and spatial attributes, res-pectively. The data was provided by the HPDRC laboratory[2]. Second, joined records were stored in a third database PHY-YP inside our *Data Repository*, and its attributes were indexed by the *Data Indexer* component. The use case demonstrates the improvement of geolocation precision in the *Physicians* database with matching objects in the *Yellow Pages* database; initially, records in PHY were geolocated to the center of their ZIP codes.

During the demonstration, users will have the opportunity to interact with the system by visualizing the joined data in the PHY-YP database as shown in Figure 3.7.

[2]http://hpdrc.fiu.edu

3.4 Extending SpsJoin

In the previous sections we have shown our framework implementation of SpsJoin and the interaction of its components in the architecture. Our definition relies on a similarity function that combines textual and spatial attributes and measures the degree of relatedness. Although the algorithm guarantees that the whole universe of records is not searched when looking for a good match, a large number of pairs may be of no use.

Following the approach of a set-similarity join, we define a set of parameters that allow our new SpsJoin to be fine-tune for tailored LBS applications. We take advantage of the frequency of tokens in geographic locations and rank pairs based on a novel score function. We also involve accuracy radius in the similarity computation, since it will help in finding correct matches. We evaluate our solution using a ground truth built with a pre-computed join that allow us to experiment with different parameters. We also experimented with large real world datasets to show the viability and accuracy of our solution.

3.4.1 Returning Relevant Pairs

By definition, the result set of a spatial set-similarity join is a set of pairs. However, pairs returned by a spatial similarity join might not be appropiate matches. How certain are we that the pair of objects we are matching are correct? State of the art suggests the use of set similarity functions that match records naively, as long as a similarity threshold is satisfied. Our work tries to improve this definition by merging textual simlarities with geographic distances. However, we have not fully exploited even more characteristics of the geographical data that may help in identifying truly

matches, or at least to give a more precise definition of the quality of the pairs being matched.

In this sense, if one tries to find the match of record a r in dataset S, there are some interesting observations. We measure the uncertainty of location of any object by defining a accuracy radius. This radius may be large when the measurement of location of the object was done with imprecise tools, or when the location of the object was recorded with noise. For instance, in the Physicians (PHY) database, the location of the objects contains information only of the zip code where they belong. In this case, the accuracy radii of the physicians are enclosed in their respective zip codes. On the other hand, photos from Flickr may have a small accuracy radius, if a GPS device was used to geo-tag them. The following sections elaborate more on these ideas.

3.4.2 Selecting the Best Match

In Section 3.3.2 we define a combined similarity function that satisfies our initial requirement: objects that are closer in distance and whose textual similarity is larger, will, overall, have a larger composite similarity value. However, the problem statement in Section 3.3 specifies that matched records will be sorted based on the value of the similarity function and only the top one will be part of the join result set. Although from the user's point of view this approach is very convinient, since user does not have to specify a similarity threshold, it is very likely to find a large set of pairs that are false positives. This is true especially for datasets whose relatedness is very low. For instance, let r be a record in a white pages dataset (WP) and let s be a record in hotels dataset (H). Clearly, a SpsJoin result set between WP and H will likely have a lot of false positives since r and s may share one single token and yet be close enough that the similarity function will be maximum.

A similarity threshold may be used in this case, but now it is up to the user to have clear understanding on how to define it. A fine tune process is required. However, other issues are raised when a similarity threshold is used. Let us consider selecting an independent threshold per type of attribute and let *John Smith Lex* in R dataset and *John Smith* and *John Lex* in S dataset, all in Miami. First of all, a set-similarity threshold of 0.9 will miss both of them in S. A much smaller threshold will get too many false positives, many of them with a closer similarity value. Let us consider now the case of the spatial attributes. A distance threshold indeed works but one might be missing important matches if the distance threshold is not large enough. This is the case of a physician geolocated at a certain zip code. If the user specifies a short search radius, it might not find the true match.

For certain problems, such as deduplication or recommendation systems, finding the best match is not of paramount importance. As noted in [BGM12], finding pairs that satisfy a threshold may be enough for these tasks. However, in this work we are interested in extracting knowledge from the best match. Geolocation enhancement is one of those applications, as we are interested in extracting the geographic coordinates of the best match. Our overaching goal is to be able to rank matches so that our search returns useful candidates. Our approach, on the other hand, returns matches that we are *almost* certain. This notion of certainty is measured using a similarity function.

Figure 3.8 shows an example of a geolocation enhancement of records in R. Each record has an uncertainty radius, which is the circular region surrounding each point displayed in the figure. We are interested in finding the best match for record r_1, "John C" and we set our textual similarity threshold to 0.3. The set of candidates is composed of the records s_1, s_2 and s_3.

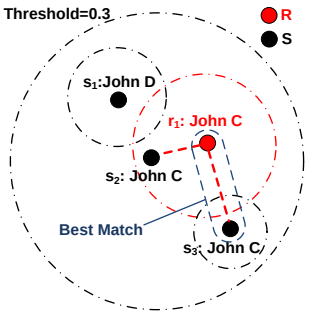


Figure 3.8: Candidates for record r_1 "John C". Best match is s_3

Notice that all three records satisfy the textual threshold of 0.3. If we further set another threshold for the distance, meaning that records that are outside a certain radius will be pruned, we have a clear winner: record s_2. However, s_3 is actually a bad choice for the best match. The reason is that its uncertainty radius is too large. This means that it can be located at any place within that range. Record s_3, on the other hand has a smaller uncertanty radius, it is more accurate and it is more likely that this record is the best match. Therefore, the uncertainty radius has to be taken into account in the similarity computation.

3.4.3 Geographic Inverse Record Frequency

When matched pairs have been computed, a similarity function measures the degree of relatedness between the matched objects. Similarity functions (e.g. Jaccard coefficient, cosine similarity) do not exploit the richness in content that low frequency terms provide to the overall similarity measurement. In geographical datasets, the frequency of the terms is dominated by the geography where they belong. For instance, demographic areas provide a clear distinction when it comes to names found in the population. In hispanic/latino areas in the U.S., "Maria" is a very popular name. However, it is not the case in places where population is mostly english speaking. Therefore, in mostly english speaking areas, "Maria" provides

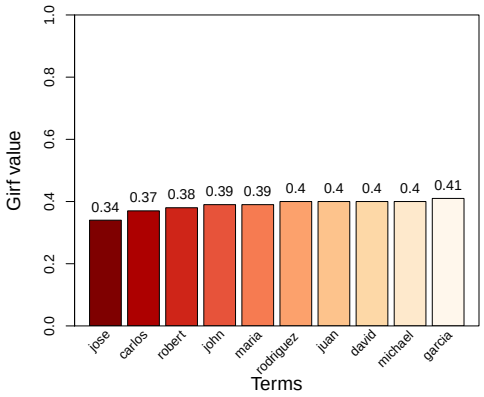
more information about the certainty of relatedness of the match than in mostly spanish speaking areas.

To leverage this intuition, we incorporate a term content weight into the similarity funtion. Inspired in the *inverse document frequency* (idf) measurement from information retrieval theory [MRS08], we define the *Geographic inverse record frequency*, $Girf_t$, of a term t within geography G as shown in Equation 3.2. We denote n the number of records within G and m the number of records that contain the term t in their textual attributes. If t is not within G, $Girf_t$ is undefined.

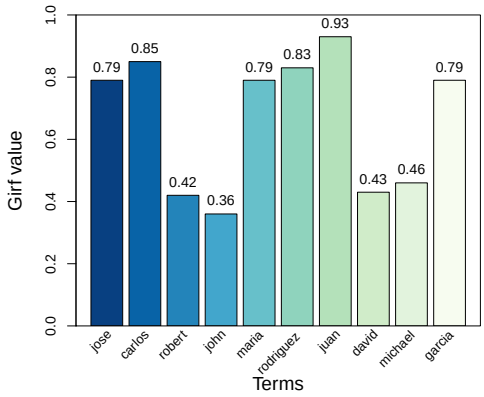
$$Girf_t(G) = log_n(\frac{n}{m}) \tag{3.2}$$

Thus, the $Girf$ of a rare term is high, whereas the $Girf$ of a frequent term is likely to be low, depending on the geography where they have been measured. To see this effect, we extracted and tokenized the names in the *contact person* field in the Yellow Pages (YP) dataset from two different geographies: Miami, Florida and Anchorage, Alaska. The number of extracted records in Miami is 92071 and the number of extracted records in Anchorage is 10854. Figure 3.9a shows $Girf$ of the top 10 more frequent terms in Miami whereas Figure 3.9b shows the $Girf$ of the same terms in Anchorage. For instance, term *jose* in Miami is very popular, so a set of candidates that contains the term *jose* may be large and therefore, it does not carry a distinguishable feature for any of the candidates in order to stand out as a good match. However, in Anchorage, jose is quite infrequent and therefore a candidate match that contains the term is likely to be more relevant than any other candidate with a similar set of matching terms.

Another important observation is that the $Girf_t$ value of t in a given geography G is maximum w.r.t geographies contained in G. Thus, for a given geography $G_i \subset G$, $Girf_t(G) \geq Girf_t(G_i)$.



(a) Miami, 92071 records.



(b) Anchorage, 10854 records.

Figure 3.9: $Girf_t$ values for the top ten terms found in Miami

3.4.4 Incorporating *Girf* Values Into the Content Similarity

As shown in the Section 3.4.3, *Girf* values can be used to score a term in a textual attribute. The question of how to incorporate this into the textual similarity function (from now on called content similarity function) has the folowing rationale. We need to keep the content similarity bounded. Therefore, we quantify the contribution of each term in a weighting function that decreases the content similarity if the average of the *Girf* values is small. When this average is large, the true similarity remains intact so attaining maximum value of 1 is desirable. To achieve this, we use the following weighting function.

$$CW(a_r, a_s) = \frac{1}{(2 - Gf_t(G))} \tag{3.3}$$

Where $Gf_t(G)$ is the average of all $Girf_t(G)$ within geography G provided that $t \in a_r \cup a_s$.

3.4.5 Definitions

This section presents important definitions and parameters that SpsJoin receives to perform fuzzy joins.

- **Candidate.** A record s is a matching candidate for record r if the accuracy radius of s overlaps the accuracy radius of r and they share at least one token(term) in the textual attribute.
- **Radius.** It is constant that defines a circle around the point coordinates of the record such that there is a high confidence that the object lies entirely within this circle. It is also called the accuracy radius.
- **Default Radius.** It is a dataset dependent radius that expresses the default accuracy radius of the objects in the dataset. In other words, it is the smallest accuracy radius that an object can achieve in the current dataset. For instance, if we know that Yellow pages has many objects with rooftop level accuracy, we may assign a 30 meters default radius for matching.
- **Geography.** It is a geographical extent that encompases an area that may or may not contain records of a predefined dataset.
- **Padded Radius of record** r It is the radius of the record plus a constant δ specific to a particular join run.

3.4.6 Incorporating Uncertainty Regions

When working with geographic datasets, it is important to understand the difference between the concepts of accuracy and precision of geographic attributes, e.g. geographic coordinates. Indeed, the location of the objects in the dataset have been

measured using either GPS devices, geocoder engines, or any other device that produces geographic coordinates e.g. cellphone triangulation, Wi-Fi based positioning systems, etc. In particular, GPS devices rely on satellites to compute the location of users by comparing distances from the subject and the time the signal takes to reach the GPS receiver. This computation is affected by several factors, e.g. atmospheric, noise, etc. that affects the accuracy of the measurement. Therefore, they can only guarantee an accuracy within a certain radius on average. Similarly, *geocoder engines* translate strings (home addresses) into geographic coordinates. The geocoder strives to *geolocate* the given string using a database of streets. When the address matches exactly, the geocoder simply returns the corresponding coordinates. Since perfect match is not always the case, as the string may contain typos, or the streets database does not have all information, the geocoder interpolates the given address using approximate locations that are partially extracted from the address, e.g. city, street, state, etc. and returns the geographic coordinates along with an accuracy level (accuracy radius). We define the accuracy as the tendency of the measurements to agree with the true values. It is measured using an *uncertainty region*, that is a closed region where there is a guarantee that the true location of the object may lie. On the other hand, precision is the degree to which the measurement pin down an actual value. In general, there is no need for much more precision in the measurement than there is accuracy built into it. In fact, using too much precision may mislead users of a system into believing that the accuracy is greater than it really is. In our model, we encode the uncertainty region using the radius of the circunference that encloses the uncertainty region. Given an object r in a geographic dataset, we denote r_{rad} as the lenght of the radius that defines uncertainty region. We will refer r_{rad} as the *accuracy radius*. Equation 3.4 defines a piecewise function that penalizes

the degree of the quality of a pair, based on the uncertainty region of each object and their dataset dependent default radius.

$$RW(r,s) = \begin{cases} \frac{dist(r,s)+rad(r)+rad(s)}{def(R)+def(S)} & \text{if not fully contained} \\ \\ \frac{rad(r)+rad(s)}{def(R)+def(S)} & \text{otherwise.} \end{cases} \tag{3.4}$$

Hence, when one of the objects is fully contained within the uncertainty region of the other, the distance between both is zero. Therefore, the first part of RW is used as region weight. When uncertainty regions overlap, the distance between objects is considered important, since the best pair is likely to be the closest one. In order to know when two regions overlap, the inequality $dist(r,s) \leq rad(r) + rad(s)$ has to be satisfied.

3.4.7 Putting it All Together

We incorporate CW from Equation 3.3 and RW from Equation 3.4 weights into our combined similarity measurement. Equation 3.5 shows the transformed similarity function sim_w after we plug in the weights.

$$sim_w(r,s) = \frac{tsim(a_r,a_s) * CW(a_r,a_s)}{1 + RW(r,s)} \tag{3.5}$$

Note that sim_w is still bounded but it does not reach 1 when both, r and s, are the same. The reason is that there are no precise objects in the real world. There is always an uncertainty region and the weights of the Equation are intended for reflecting this.

3.4.8 Incorporating Uncertainty: Entropy

It is a measure of unpredictability in a random variable [CT91, TSK05b]. The best example used to explain it is to consider the toss of a coin. When one toss a fair coin, the probability of obtaining heads or tails is equal to 0.5. Therefore, it is hard to predict what would be the result of the next coin toss. However, if the coin has two heads and no tails, we are "certain" on what the next outcome will be. In this case, the entropy tells us that the unfair coin gave us more information than the fair coin, in order to predict the outcome.

Let X be a random variable. The entropy H is defined as shown in Equation 3.6:

$$H = \sum_{i} p(x_i) \log p(x_i) \tag{3.6}$$

We leverage this concept to measure the uncertainty of selecting a best matching pair. Given a set of possible candidates, we group them into buckets of similarities. For our purposes, we use only 5 buckets (clusters). We cluster the data using a one-dimensional k-means algorithm which partitions the data optimally. The mean of each bucket is the representative similarity value of the bucket. Then, we measure the entropy of the cluster. If the entropy is large (close to one when normalized) we know that we cannot make a decision, since we are uncertain on which candidate pair to pick. On the other hand, if the entropy is low (close to zero), then we are more certain that we can pick a candidate pair. We will see later how to pick the uncertainty threshold to reject or to accept a candidate pair by combining this measure with the similarity value.

3.5 Improving SpsJoins

Maintaining scalability is an issue of paramount importance when working with large datasets in parallel. We express our algorithms using the MapReduce programming model for two main reasons. First, the framework provides all the functionality of a distributed system such as fault tolerance and job scheduling, all of this transparent to the programmer. Second, the possibility to scale out when more nodes are added to the computational cluster without changing the main program guarantees the portability of the proposed solution to different cluster configurations. We modified the our implementation shown in Section 3.3.2 by incorporating all the features that we have found so far.

3.5.1 Spatial Partitioning Phase

Scalability In this work, we propose a small modification in the k-means algorithm to handle data skewness. Although far from perfect, our heuristic tests for large variations in the sizes of the partitions, using *Interquartile Ranges* (IQR), and recomputes k-means on those outstanding partitions. Our experiments show that this heuristic works well when centroids are biased toward outliers or small sized partitions. However, there is still a change of large partitions after the heuristic, that is a consequence of poor selection of centroids.

Our k-means implementation is efficient in the sense that we do not emit all the records through the network. We rely on a *Combiner* function that aggregates points in local centroids and emits weighted aggregated points to reducers. Reducers obtain streams of weighted points and compute a weighted average, emiting the resulting centroid.

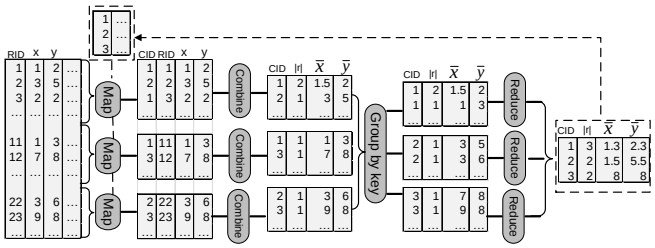


Figure 3.10: MapReduce: Spatial Partitioner algorithm data flow

We partition the datasets based on geographical component. Data skewness may harm the load balancing of the overall join algorithm. The reason is that a clustering technique may favor closeness to centroids rather than number of records in each cluster. We note that our previous Xmeans implementation tries to guess the best number of clusters. A partitioning technique can be derived with this method. However, Xmeans relies on Bayesian Information Criterion or Akaike Information Criterion [PM]. Although the technique is effective in determining a good number of clusters, it does not solve the balance problem i.e. if a cluster of a low number of records seem to be far away, the number of records are unlikely to merge to satisfy the minimum number of records require for the each cluster. The work of Ng et.al [NH94] seems to be promising but it has a major drawback for very large datasets: it requires high level of computations for solving the nonlinear optimization problem of k-means. We proposed a much simpler but powerful heuristic that seem to work very good in large datasets.

We note that we are not interested in finding the best possible solution for load balancing and the same time satisfying the constraint of the best partition. Solving that kind of problem seems to require a lot of computation and those two constraints are unlikely to be satisfied efficiently with a simple heuristic. We believe that the bulk of operations are centered in the local computation of the join. Therefore, a load balancing strategy that performs reasonable well is likely to produce good

results. The main problem with any type of load balancing technique in this join is that the accuracy radii of the considered objects may overlap several clusters. If the partition forces the object to be repeated in several clusters, then we cannot just simply reassign the object to another cluster. The object is to be kept within the cluster. Also, even if the cluster does not have the required minimum number of objects, it would be useless to assign objects that would not match between them, just because we need to satisfy a quota of minimum objects.

Our solution works as follows: Let k be the number of clusters used. Each cluster is supossed to have the maximum number of records allowed to compute a join. We then compute the IQR (interquantile range) of the number of records per cluster [TD00]. For all clusters that have more than the upper fence value (UFV) e.i. more than $Q3 + 1.5 * IQR$, we split them based on the median value mv of the clusters. Then, we compute $kx = N/mv$ that we use to further compute a kx-means algorithm.

Here, we briefly explain the modification for the spatial partitioning procedure, in order to load-balance the clusters.

- Each mapper simply maps the records that need further parititioning.
- On each reducer: first, we set $M = n/k$. For each centroid c (in the reducer), if centroid c has more than $2M$ counts, let it be $|c|$. If set $m = |c|/M$ then run an m-means with the centroids in the reducer. Then we eliminate previous centroid and we add the new m centroids.

3.5.2 Local Join Implementation

We briefly explain our approach for computing local joins using two different data structures. As we shown before, local joins run on each reducer. In order to decrease

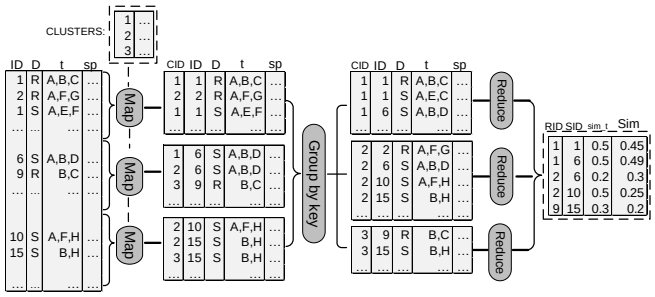


Figure 3.11: MapReduce: Local Join algorithm data flow.

the running time, we implemented two spatial data structures that we describe below.

SpsJRTree. This algorithm uses a lightweight R-tree [Gut84] implementation. We augment the original algorithm with a textual data structure similar to inverted files for text processing, that we call *token file* (TF). TF is simply an extension of the inverted file. It maps tokens to lists of nodes that contain the token. It was implemented as a hash table with double indirection for fast retrieval of nodes. To fill out the TF, once the R-tree has been built, the algorithm extracts the tokens from the records at the leaves. For each token at the leaf, it checks if the current leaf has not been added in the list of nodes of the token in TF. If not, then the algorithm inserts it and continues recursively using its parent. The base case occurs when the current node has been inserted in the TF list of the probed token. For this algorithm to be memory efficient, several optimizations can be done. First, the nodes of the TF data structure may leverage the frequency of the token. For frequent tokens, the size of the buckets is bounded by the number of nodes. However, for infrequent tokens e.g. tokens that appear just one time, the size is bounded by the height of the R-tree. Another optimization can be done by observing the distribution of the tokens in the nodes. Infrequent tokens may map the same nodes if they belong to the same leaf, so we can collapse the inverted lists of both tokens. However, this requires additional work in the algorithm and , for efficiency, we need to keep it

simple. **SpsJBoxSort.** This implementation uses a space efficient data structure called *Box Sort* tree [Hou87]. The Box Sort tree is similar to a $k-d$tree but for polygons. One advantage it has w.r.t the R-tree is the small amount of data used to maintain it. The space complexity is $O(n)$, where n is the size of the input data and search queries are guaranteed $\log(n)$. This data structure is suitable for this problem not only because of the space complexity, but also because we do not need to perform deletions or insertions in the local joins.

We also included text information in the nodes that allow us to prune unnecessary branches when we do not find tokens that are deeper in the leaves. In Section 3.6.1 we compare SpsJBoxSort and SpsJRtree for performance.

As a caveat, we noted that the radius picked for our algorithms can have a significant impact in the performance. The reason is that objects have accuracy radii. This accuracy radius may be very large, so in that case, our R-tree or Box Sort will have too much overlaping which implies that spatial search is not as effective as token pruning. On the other hand, if the precision radius is small, spatial prunning is quite effective and token prunning becomes important only at high levels of the R-tree. That is the main reason that we use text augmentation in both of our algorithms for spatial set-similarity joins.

Algorithm 1 Spatial Set-Similarity Join algorithm

1: **procedure** SpsJoin(R,S, k, $type$)
   Spatial Partitioner Phase:
2: **if** $type$ = LEFT **then**
3: $C \leftarrow$ spatialPartitioner(R) ▷ This works for Left outer join only
4: **end if**
   Local Join Phase:
5: $Candidates \leftarrow \emptyset$ ▷ List of candidate pairs $< r, list[s] >$, $s \in S$
6: *Map* each $o \in R \cup S$ to pairs $< c_i, o >$ such that $o.MBR$ overlaps $c_i \in C$
7: **for all** $c_i \in C$ **do** ▷ In parallel
8: *Reduce* localJoiner(c_i,$list[o_i]$)
9: Append $< r, list[s] >$ pairs to $Candidates$
10: **end for**
   Filtering Phase:
11: **for all** $cand_i \in Candidates$ **do** ▷ In parallel
12: $< r, l > \leftarrow cand_i$
13: $G_i \leftarrow \bigcup_{s \in l} s.MBR$
14: Create vocabulary V from terms in l and r
15: Compute $Gir f_t(G_i)$ for each $t \in V$
16: $fl \leftarrow \emptyset$ ▷ Priority queue of size k
17: **for all** $s \in l$ **do**
18: Push $< sim(r, s), (r, s) >$ to lf
19: **end for**
20: **while** $fl \neq \emptyset$ **do**
21: Pop fl ▷ Returns pairs $< sim(r, s), (r, s) >$
22: **end while**
23: **end for**
24: **end procedure**

3.5.3 On Radius and Distance Computations

The precision radius is inherent in the object and depends on the resolution that the location of the object was measured. Every SpsJoin operation may have different purposes. For instance, it may be used to improve location of one dataset using another dataset. In this case, the imprecise location is rather large and intersection of radii play an important role to find the right candidate. On the other hand, if the join is composed with objects with precise location, the precision radii are

small. In this case, radius overlap may not match any object. An example of this is a join between Flickr dataset and Yellow Pages. Precision radii are small on both. To handle this issue we define what we use the default radius we defined in Section 3.6.1. For instance, if we know that Flickr has geolocated objects at certain radii, we increase the default radius to 200 meters to account for people who have taken pictures far away from the target.

3.6 Experimental Evaluation

In this section we describe the performance evaluation of our techniques and proposed algorithms. We evaluate the precision of our join using a ground truth dataset that we built for this purpose. In order to understand the performance of the parallel algorithms we measure absolute running time, relative speedup and scaleup [DG92].

Our testbed is a two processor AMD machine with 24 cores 16MB cache machine with 256GB RAM operating at 1066MHz with 12 x 500GB disk drives at 7200 RPM SATA. For our experiments we used Hadoop v1.0. in a pseudo-distributed configuration. The machine was configured with a SELinux Carbon Release 6.2 operating system, running Java 1.6 JVMs. In order to maximize performance, we set up the replication factor to 1 and disable the speculative task execution feature.

In order to exercise our join algorithm, and to test precision of the join, we used the following real world datasets:

- **WPMIA.** It has over 1.3M records of white pages in the city of Miami. Each record contains first name, last name, phone, title, address, city, state and geographic coordinates. The attributes that we used for joining are the address and the geographic locations. Note that the accuracy of this dataset

may vary, as it underwent a geocoding process. The total size of this dataset is 145Mb.

- **PRMIA** It has over 360K records of parcels in the city of Miami. Each record contains a parcel ID, Folio number, registered owner, address and geographic coordinates. It is worth noting that this dataset has very accurate geographic coordinates. The accuracy radius is approximately 15 meters, as these are GPS coordinates. The attributes used for computing the join are

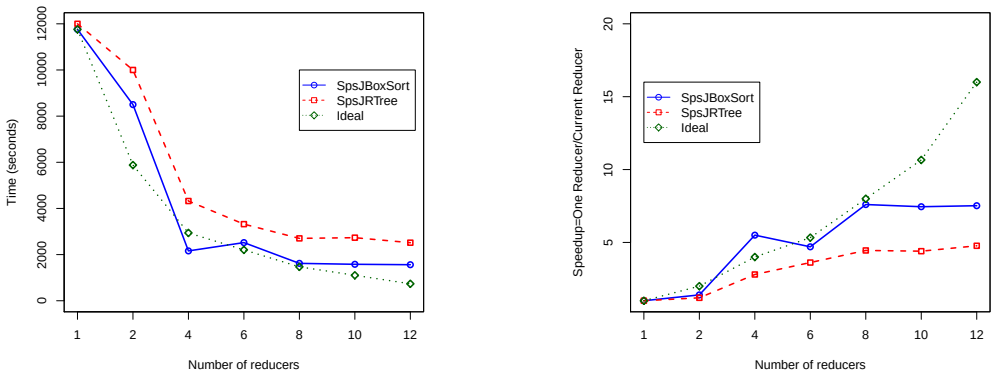
We evaluate and discuss our techniques in two different layers: performance and precision of the join. We tested SpsJRTree and SpsJBoxSort approach. We used WPMIA and PRMIA to test performance and Section 3.6.2 describes a ground truth data set to test precision.

3.6.1 Performance

We observed that a nested loop approach of an SpsJoin to handle such large amounts of data is simply a waste of computation and time, specially if users rent cloud computing solutions. Therefore, we do not compare our algorithms with the naive case. Instead, we assume that our baseline partitions the data using the spatial attributes. We test our two approaches, SpsJBoxSort and SpsJRTree and evaluate their performance.

For each approach, we tested its running time but we also show its ideal speedup curve. Figure 3.12a shown the running time of both approaches and also its speedup. For instance, if the cluster has twice as many reducers and the data size does not change, the approach should be twice as fast. Figure 3.12b we show the same numbers, but plotted on a relative scale. That is, for each number of reducers, we plot the ratio between the running time for the smallest cluster size and the running

time of the current cluster size. For example, for the 10-reducers run, we plot the ratio between the running time on the 2-reducer cluster and the running time on the 10-reducer cluster. We can see that all two combinations have similar speedup curves, but none of them speed up linearly.



(a) Running time.

(b) Relative Speedup.

Figure 3.12: Running time and Relative speedup for SpsJBoxSort and SpsJRTree

3.6.2 Join Precision

In this section we test the precision of our join. To that end, we created a ground truth that allow us to measure and fine-tune the parameters needed to get as much true positives as possible.

Ground Truth

SpsJoin combines records from two datasets and returns the most similar pairs based on the similarity function we described in 3.3. However, the best match $s \in S$ of an object $r \in R$ may not be necessarily an actual match. For instance, it may be the case that *John S. Smith* has a best match *Alice Smith* in a given dataset S. Defining a threshold is good way to address this problem but the question of how large it should be remains unanswered.

In this section, we define a ground truth or a golden dataset, so that we can have an idea of the value of the thresholds and parameters that we should set our system.

It is important to note that this threshold depends on the type of the set-similarity and distance functions. For example, when computing a SpsJoin on Flickr dataset and Yellow Pages, we may use overlap similarity to match pairs and we might be interested in high overlap of infrequent keywords. This overlap may be small but the keywords involved can be so rare they may boost the overall similarity measure.

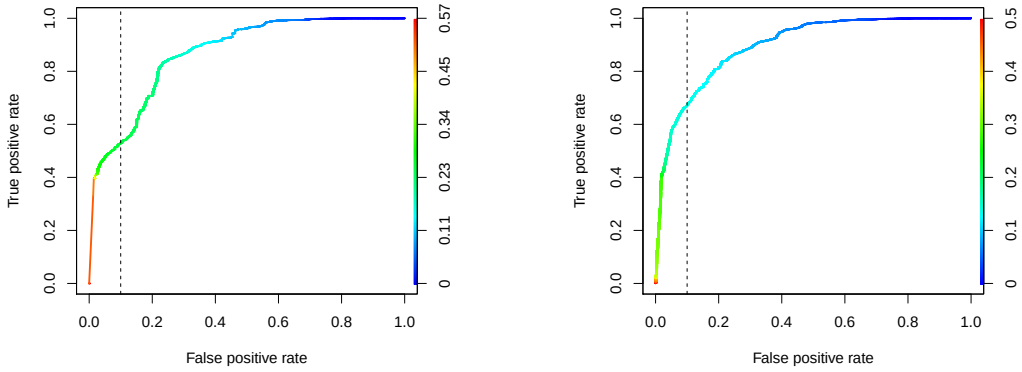
If the datasets has nothing in common textually and geographically, then a spsjoin is not the correct approach. There are efficient techniques to determine a similarity join selectivity. In particular [LNS11] use stratified sampling methods to sample both datasets, defining mutually exclusive strata that will be joined using *LSH* techniques. Unfortunately, this approach does not seem to work for spsjoins, since LSH techniques require metrics in a metric space that satisfy the triangle inequality.

Measuring the precision performance of an spsjoin is difficult without a ground truth. Therefore, for our ground truth we used an already known join of two datasets on a given, basically an already computed relational join.

3.6.3 Experiment Definition

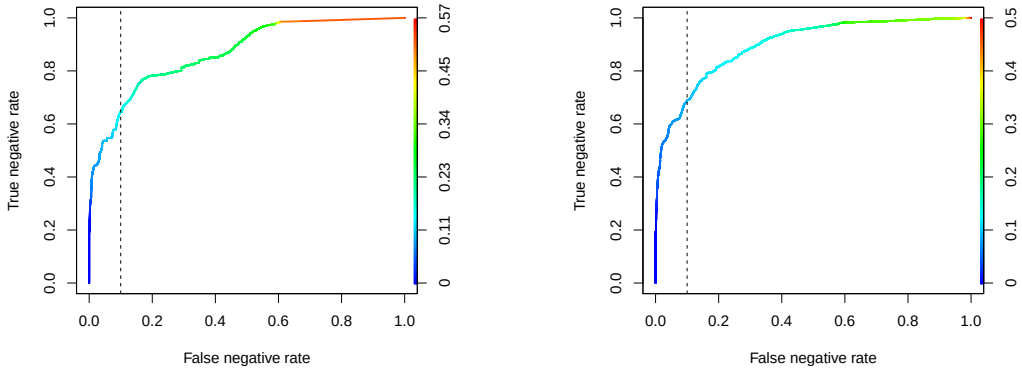
In order to measure the precision of our join, we used an already computed join called `CALLREAL`. This file contains MLS Real Estate listings in Miami-Dade county that are updated daily. It consists of several characteristics of the parcels such as number of beds, number of bathrooms, number of rooms and other geographic and non geographic information. This dataset is the result set of a join between the `ALLREAL` and `FLPROPERTIES` on the folio number, which is a unique identifier of the parcels.

We can test the precision of an SpsJoin by splitting this join into its original datasets and rejoin them again, but this time using an SpsJoin.



(a) Only similarity values. (b) Combined score.

Figure 3.13: ROC Curves: True positive - False positive rates



(a) Only similarity values. (b) Combined score.

Figure 3.14: ROC Curves: True negative - False negative rates

Then, in the experimention we will compute a SpsJoin on `ALLREAL` and the dataset of florida parcels `FLPROPERTIES` on their address and it geographic coordinates. Note that the addresses of a matching pair may not be exact. This also implies that the location is approximate, since each dataset might have been treated with different geocoders. Therefore, the matching is *fuzzy*.

We run our experiments following an incremental approach until we obtained good performance. This performance can be shown through *ROC* curves [SSBL05]. ROC curves are graphical plots that illustrate the performance of binary classifiers as their discrimination thresholds are varied. We harness their power to show our

incremental approach. They can show the percentage of false positives versus the percentage of true positives as the threshold is varied during the experiment.

For instance, Figure 3.13a shows an ROC curve using the similarity value. In the x axis, the ROC curve is showing the percentage of false positives. The y axis shows the percentage of true positives. Parallel to the y axis there is an odometer that shows the different similarity values. This odometer is useful to pick a threshold value that maximizes the number of true positives.

We can see from the figure that selecting a threshold ≥ 0.45, we will get about 40% of true positives. However, with a threshold ≥ 0.34 we get 45% of true positives, but we need to tolerate 10% of false positives.

In order to improve the performance of our metrics, we propose the similarity/entropy score. The rationale for defining such a score is that, as the similarity increases, our uncertainty in selecting a best match decreases. Therefore, large values of this score are likely to capture more true positives. Figure 3.13b shows this improvement. From the plot, we can see that tolerating 10% of false positives, we get up to 70% of true positives. This means a 25% improvement over the similarity approach.

We also experimented with the rate of false negatives. Figure 3.14 shows a similar analysis using both, the similarity metric and the entropy score. In this case we do not see a huge improvement when we tolerate a 10% false negatives. The rate of true negatives is very similar (65%) in both, Figure 3.14a and Figure 3.14b.

In conclusion, this ground truth allowed us to measure the precision of our join. It was shown that a threshold of 1.8 in the entropy score gives the best performance, capturing a good number of true positives. The main drawback is that we will not be able to get 100% of true positives. The main reason is that matching two records may depend on other attributes that are not being considered in the similarity

function. However, this system gives an initial step in matching pairs of records from two different datasets using similarity functions.

3.7 Appendix

In this section, we describe how to setup the experiments using the system implemententation described in this chapter. We also document each module of the system for further reference.

3.7.1 SpsJoin Implementation

As described in Section 3.6, SpsJoin was implemented in Java 1.6. We used Apache Hadoop V.1.0 and it is available open source in [Apa12].

The source code was created using NetBeans IDE 7.2, available in [Net].. NetBeans is an Integrated Development Environment (IDE) that provides the user with comprehensive facilities for software development. The default configuration of NetBeans includes a Java compiler and other plugins e.g. Tomcat, but other languages can be used.

The different modules of the system were developed in separate projects each. This means that each module is part of a java package that was shipped in jar files. However, when it is deployed to the cluster, the source code was recompiled and shipped in a single jar for execution. More details are described below.

Cluster Configuration

In Section 3.6, we mentioned that the cluster was configured using the *pseudo*-distributed mode. The status of the job tracker of the cluster can be queried using the URL `http://131.94.128.150:50030/jobtracker.jsp`.

In order to configure the cluster, a user was created. The nickname of the user is `hadoopuser`. This user is the one that owns the source code and can perform the executions. To get access to the code, log in with `hadoopuser`. From the home folder, go to the `/hadoop` folder. This folder contains the instalation of the cluster. The user can also modify the configuration by modifying the hadoop files `core-site.xml, mapred-site.xml` and `hdfs-site.xml`. For instance, to modify `core-site.xml`, run the following commands

```
% cd  /hadoop/conf
% nano core-site.xml
```

The above commands will open a text editor so that the XML file can be modified for a new configuration. Check [Apa12] for documentation about Hadoop cluster configuration.

We may check the status of the HDFS's Namenodes by querying the URL `http://131.94.128.150:50070/dfshealth.jsp`. As we can see, the cluster is configured with a single Namenode with no replication. The reason for this is that the machine has been set with a RAID 6 controller.

SpsJoin Source Code

SpsJoin source code is available in the folder `/spsjoinSoftware`. It contains all unpacked modules of the system. However, when the program is executed, all source is compiled and packed in a jar file called `alllibs.jar`. The breakdown of the modules, with their respective folders and descriptions is shown in Table 3.2.

Note that folder `parallelspsjoin` contains only `.class` files that forms the main program. The actual source code can be found in the `allsrc` folder, under file

Table 3.2: List of modules in SpsJoin

Folder	Description
boxsort	Implements of the Box Sort data structure
com	Implements the Bloom Filter data structure
genericextractor	Contains the XML parser and other utilities
geometry	Implements Rectangle and other geometry classes
onedimensionalpartition	A unidimensional k-means implementation
parallelkmeans	Implements spatial partitioner in Java
parallelsampler	Implements a random sampler for the input data
parallelspsjoin	Contains the .class files of the main SpsJoin program
rtreeimpl	Implements a lightweight R-Tree (experiments)
mrUtil	Utilities for `parallelsampler`
utils	Package with utilities e.g. Similarity functions

names `SpsJoinRTree.java` and `ClusterPartitioner.java`. The process to compile and generate a full jar package for execution is as follows

```
% cd /home/hadoopuser/spsjoinSoftware/
% javac -classpath  /hadoop/hadoop-core-1.0.1.jar:allsrc
                                    -d allsrc allsrc/*.java
% jar -cvf alllibs.jar -C allsrc/ .
```

Once the file `alllibs.jar` is created, we can run it in Hadoop.

3.7.2 Creating A Configuration File

SpsJoin requires an XML file that provides the main parameters of the system execution. This XML file can be created using any text file editor. This file has to be stored in the `/spsjoin` folder. In this section we provide the description of the parameters along with an example XML. The folder `/spsjoinSoftware` contains

a template that can be used to modify the join execution. Here we provide some definitions of concepts used throughtout this Section.

Primary textual field. It is the main textual attribute that the join operation uses for content similarity matching.

Secondary textual field. It is the optional textual attribute that the join operation uses in case the primary textual attribute has no match with any of the terms in the content similarity. In case where match is on primary and secondary, the join operation keeps the content similarity that achieves the largest one.

Radius. it is constant that defines a circle around the point coordinates of the record such that there is a high confidence that the object lies entirely within this circle.

Default Radius. It is a dataset dependent radius that expresses the default precision radius of the objects in the dataset. In other words, it is the smallest precision radius that an object can achieve in the current dataset. For instance, if we know that Yellow pages has many objects with rooftop level precision, we may assign a 30 meters default radius for matching.

Padded Radius. It is the sum of the radius of an object and a positive constant. Note that the confidence of the object lying within its padded radius increases.

Distance Threshold. It is the maximum distance radius that the system will use to search for objects. By default it is set to 3 miles.

Similarity Threshold. It is the threshold used to filter best candidates. It depends on the experiments that we defined in the previous Section.

Now we show an example of the XML file that we used to configure the system. The file starts with a main tag called "parameter" and its subfields are defined within.

```
<parameters>
        <joinType>LEFT</joinType>
        <simtype>0</simtype>
        <relation>
                <id>R.asc</id>
                <primary>name</primary>
                <secondary></secondary>
                <latitude>latitude</latitude>
                <longitude>longitude</longitude>
                <defaultRadius>200</defaultRadius>
        </relation>
        <relation>
                <id>S.asc</id>
                <primary>title</primary>
                <secondary></secondary>
                <latitude>latitude</latitude>
                <longitude>longitude</longitude>
                <defaultRadius>200</defaultRadius>
        </relation>
</parameters>
```

3.7.3 Running SpsJoin

SpsJoin can be run using the `alllibs.jar`. This section provides an step-by-step process along with an example that shows the execution process. We begin by selecting the datasets that we are going to use for the operation. We assume that these datasets are stored in a local disk. In our case, they are stored in `TFoverlays`

folder. Also, we have to create the XML configuration file that SpsJoin uses for execution, as we showed in Section 3.7.2.

SpsJoin Directory Structure

Spsjoin operates in the Hadoop Distributed File System (HDFS). The root of the directory structure is the folder `/spsjoin`. Table 3.3 shows a breakdown of the folders in `/spsjoin` folder along with their description. The `/input` folder feeds SpsJoin with input datasets along with their header files. The user has to make sure that both, the name of the file and the header have the same name. The only difference is that the header file contains the extension ".header" appended in the name. The `/kmeans` folder records the execution of the spatial partitioner. This is a k-means algorithm that runs several times depending on the number of iterations set by the user and the activation of the load balancer. The output (the list of centroids) is kept in a subdirectory called `/finalcentroids` in the `mergedFile` file. The `/modinput` folder contains the modified input of the SpsJoin program. It was created to add an additional surogate key. This is done because we have identified that several datasets do not include a unique key in their fields. This is necessary to join the original records after the SpsJoin execution. Note also that the new files are renamed as "R.asc" and "S.asc". This way, SpsJoin identifies the provenance of the records when processing the files in Hadoop. Recall that Hadoop does not differenciate between files in its input so we take advantage of the split names to identify them. The `/output` folder stores the SpsJoin output after the execution of the main algorithm. It only contains a set of files in SpsJoin format with the ids of the matches and other information. Note that this is not the final output. The folder `/recordJoin` contains the final result set. It has two subdirectories that records its execution. The result set is stored in the folder `/output2`. Finally, the `/sampler`

Table 3.3: List of folders in `spsJoin` folder

Folder	Description
input	Stores input datasets
kmeans	Records the execution of K-means
modinput	Stores the input with surogate keys
output	Stores the output of SpsJoin (no original records)
recordJoin	Stores the result set with original records
sampler	Records the execution of the sampler algorithms

folder is used basically to store the execution of the sampler algorithm. It is the input of the kmeans algorithm so it is created along the way in the SpsJoin execution.

Running Example

Now that we know how the directory structure is built, we can show the step-by-step process to run an SpsJoin. First, we copy the input files from a local directory to the HDFS. Recall to always log in using the `hadoopuser` username to run the program. Also, the user has to run `ssh localhost` before running the system. The following commands show how the execution works.

```
% hadoop dfs -copyFromLocal /home/hadoopuser/data/* /spsjoin/input/
```

Then, copy the XML configuration file:

```
% hadoop dfs -copyFromLocal
    /home/hadoopuser/spsjoinSoftware/spsjoin.config.xml /spsjoin/
```

These commands copy the input files and the configuration file from local disk to the HDFS. After this, we can run the sampler. This module basically samples the dataset in order to obtain a good initial seed of centroids needed in the spatial partitioner. We proceed by running the following command

```
% hadoop jar  /spsjoinSoftware/alllibs.jar parallelsampler.Sampler
                                100 4 groundTruthR.asc groundTruthS.asc
```

The parallel sampler receives three parameters, the sampling rate (100), the number of initial seed centroids (4) and two datasets. As mentioned before, the output of the parallel sampler is the input of the spatial partitioner.

At this point, we have what we need to run the spatial partitioner. Our spatial partitioner is a k-means algorithm that also runs a load balancer when the size of the clusters are not evenly distributed. In order to run it, we have to use the following command

```
% hadoop jar  /spsjoinSoftware/alllibs.jar parallelkmeans.Kmeans 3 2 0
```

The spatial partitioner receives three parameters: the number of iterations (3), the number of reducers used in the execution (2) and the mode (0). This mode is used primarily to determine if this is a first time executing the partitioner or a subsequent time, 0 for first time or 1 for the rest. It is useful only in experimentation.

Once we run the spatial partitioner, we are ready to execute the main join program. The following command shows how.

```
% hadoop jar  /spsjoinSoftware/alllibs.jar parallelspsjoin.SpsJoinRTree
```

The main program receives only one parameter, which is the number of reducers that are going to be used in the execution. As mentioned before, this program leaves the output in the `/output` folder. The resulting file is not a final file. This process has to undergo another procedure to join the original records. To do this, we run

the following command

```
% hadoop jar  /spsjoinSoftware/alllibs.jar recordsjoin.JoinRecords 1
```

The record join receives one parameter: the mode. Mode is an integer that can be one or two. By default, it should be one. Two is only for debugging purposes.

3.7.4 Appendix Conclusions

We have shown how to run SpsJoin using simple Unix commands. The system has been implemented using Java 1.6 and Hadoop V 1.0. It is important to note that we used the last version of Hadoop by the time this document was written. This system can also be extended to handle more interesting variants of joins, such as multi-way joins. SpsJoin showed great precision and scalability and future improvements will definitively benefit from this initial version.

CHAPTER 4

TOWARDS PRIVACY PRESERVING LOCATION BASED SERVICE APPLICATIONS. THE SAFE CITIES CASE.

In this chapter, we address privacy concerns and motivate LBS applications in the context of *Safe Cities*, using a mixed spatial and nonspatial approach. Overall, this chapter aims to enable the vision of smart and safe cities, by exploiting mobile and social networking technologies to securely and privately extract, model and embed real-time public safety information into quotidian user experiences. We provide novel approaches to define location and user based safety metrics. We devise iSafe, a privacy preserving algorithm for computing safety snapshots of co-located mobile devices as well as geosocial network users. In our analysis, we leverage SpsJoin to find venues that have not been reviewed and to further compute the crime indices of their locations.

Acknowledgements. I would like to acknowledge Mr. Mahmudur Rahman. He contributed to specific parts shown in this chapter and gave valuable insights. In Sections 4.7 and 4.8.4, he wrote Android source code for the iSafe algorithm and for the web server-plugin. He performed crime forecasting experiments, as shown in Section 4.8.2, and he currently mantains the website of the iSafe project.

4.1 Introduction

Modern technological advances and, in particular mobile devices and online social networks, have paved the way toward a smarter management of resources in today's cities. As population density grows and natural disasters and man-made incidents (e.g., hurricanes, earthquakes, riots [LAR12, Fra12, Eng12]) impact increasing num-

ber of people, maintaining the safety of citizens, an essential smart city component, becomes a problem of paramount significance and difficulty.

We envision an LBS system where users are seamlessly made aware of their safety in a personalized manner, through quotidian experiences such as navigation, mobile authentication, choosing a restaurant or finding a place to live. This is a clear definition of a LBS where location data is of paramount importance. The main objective of this system is to motivate the use of a location service that addresses privacy concerns but at the same time, tackles an interesting problem in smart, safe cities. We propose to achieve this vision by introducing a framework for defining public safety. Intuitively, public safety aims to answer the question "Will location L present any danger for user A when she visits L at a future time T"?

An important challenge to achieving this vision is the need to properly understand and define safety. While safety is naturally location dependent, it is also inherently volatile. It not only exhibits temporal patterns (e.g., function of the season, day of week or time of day) but also depends on the current *context* (e.g., people present, their profile and behavior). Furthermore, as suggested by the above question, public safety has a personal dimension: users of different backgrounds are likely to be impacted differently by the same location/time context.

Previous attempts of making people safety-aware include the use of social media as a means to distribute information about unreported crimes [FAdO+10], or web based applications for visualizing unsafe areas [Cri, Gua]. The main drawbacks of these solutions stem from the difficulty of modeling safety and of integrating its use in the everyday life of people. Instead, here we investigate the combination of space and time indexed crime location datasets, with mobile technologies and online social networks to provide personalized and context aware safety recommendations for mobile and social network users.

Specifically, we first define location centric, static crime and safety labels, based on recorded crime events. We take advantage of observed crime behavior periodicities, to conjecture that location safety values are predictable. To verify this hypothesis, we investigate the ability of timeseries forecasting tools to predict future location crime and safety index values based on recorded crime events.

Moreover, we use mobile device and geosocial network technologies to record the *trajectory trace* of a user: the set of (location, time) pairs where the user has been present. When sufficient crime information exists to enable an accurate prediction of location based crime levels, we introduce the concept of *personalized safety* recommendations: A user U is safe at a location L, if the average crime index of the locations in U's trajectory trace equals or exceeds the crime index predicted for the near future at L.

When insufficient crime information exists at a given location, we propose to augment the "context" of the location with data collected from co-located mobile devices and geosocial networks. We define the *vicinity crime* metric, to be the chance of crime events being reported around a user or a group of users, based on their past location trajectories. We introduce then the concept of *context aware safety* recommendations: a user U is safe around users $U_1, .., U_k$ if U's vicinity crime metric equals or exceeds the aggregate vicinity crime metric of users $U_1, .., U_k$.

Furthermore, through the statistical χ^2 test we show that dependencies exist between the quantity and quality of reviews venues receive in Yelp (a popular geosocial network) and the crime indexes of the venues's locations. We then propose to similarly augment spatiotemporal context with trajectory traces collected from geosocial network users. We leverage SpsJoin from Chapter 3 on Yelp venues and geolocated businesses to find venues that have not been reviewed and to further compute the crime indices of their locations.

The approach outlined above relies on the ability to aggregate user location trajectories. Access to the trajectory traces of users, along with associated crime and safety index values, either by other users or a centralized service provider, raises significant privacy concerns: even social network providers have been shown to leak [KW10] and sell [SF] user data to third parties.

To address this issue, we devise iSafe, a distributed algorithm that takes advantage of the wireless capabilities of mobile devices to compute real-time snapshots of the safety profiles of close-by users in a privacy preserving manner. iSafe uses secret splitting and secure multi-party computation mechanisms to aggregate the trajectories of co-located users without learning the private information of participants.

We have implemented iSafe as a browser plugin component and an Android application. We provide extensive evaluations of our contributions using crime and census data from the Miami-Dade county (FL) as well as data we have collected from the accounts of users and businesses in Yelp [Yel], a popular geosocial network centered on user feedback. Our experiments performed on a testbed consisting of several smartphones show that the Android iSafe app is efficient: the computation overhead is a few milliseconds while the communication overhead is a few hundred milliseconds. The iSafe project can be found online [iSa], providing downloadable Chrome plugin and Android app executables.

4.2 Model and Background

The framework consists of three participants, (i) a service provider, (ii) mobile device users and (iii) geosocial networks. The service provider, denoted by S in the following, centralizes crime and census information and provides it upon request to clients.

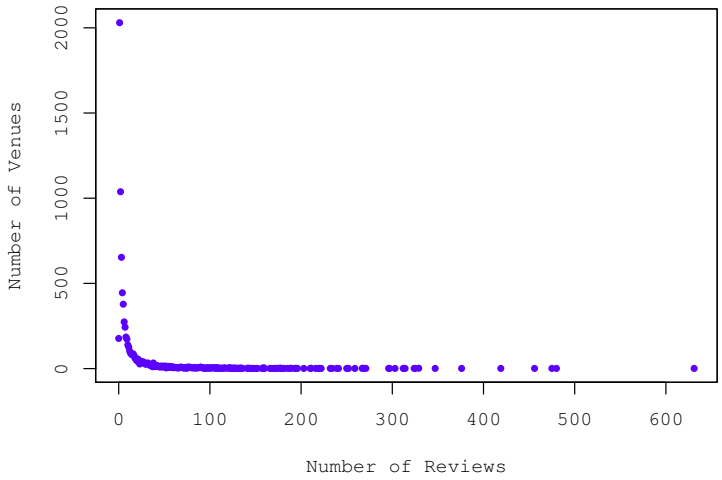


Figure 4.1: Miami venue stats: Distribution of number of reviews per venue.

We assume the mobile devices are equipped with wireless interfaces, enabling the formation of transient, ad hoc connections with neighboring devices. Devices are also equipped with GPS interfaces, allowing them to retrieve their geographic location. Devices have Internet connectivity, which, for the purpose of this work may be intermittent. Users take advantage of Internet connectivity not only to report to geosocial networks but also to retrieve crime information (both described in the following). Each user needs to install an application on her mobile device, which we henceforth denote as the *client*.

In the remainder of this section, we describe the geosocial network concept, the crime and census datasets that we use in our work, detail several forecasting tools we use and describe the attacker model we consider in this work.

4.2.1 Geosocial Networks

Geosocial networks (GSNs) such as Yelp and Foursquare extend classic social networks with the notions of (i) venues, or businesses and (ii) *check-ins*. Besides user

accounts, GSNs provide accounts also for businesses (e.g., restaurants, yoga classes, towing companies, etc). Users "check-in" to report their location, in terms of their presence at one of the venues supported by the GSN. Users can share check-in information with friends and also use it to achieve special status (badges, mayorships) and receive frequent customer discounts from participating venues. In addition, geosocial networks encourage and reward user feedback, in the form of ratings and reviews, left for visited venues. Users rating range from 1 to 5 stars and are aggregated to produce an overall venue rating.

Yelp Data. We have collected Yelp information from all the venues in the Miami-Dade county, Florida, for a total of 7699 venues. For each venue, we have collected the name, type and address, along with the list of reviews received. For each review, we collected the home city and state of the reviewer.

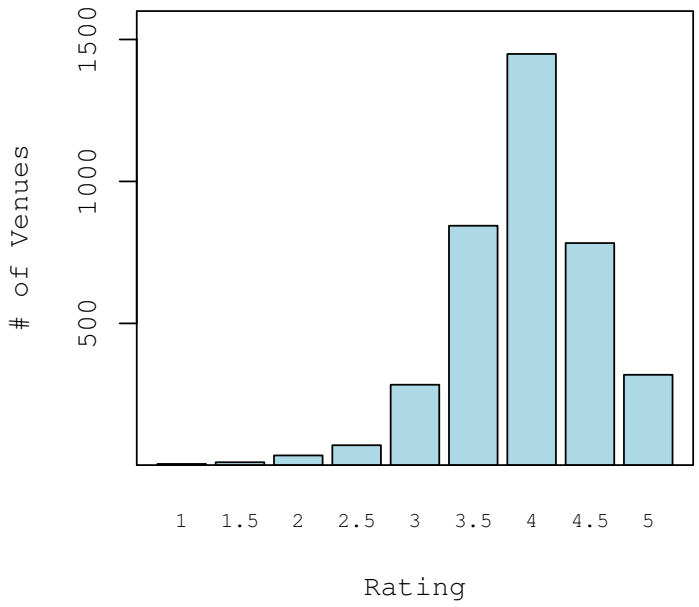


Figure 4.2: Miami venue stats: Distribution of venue ratings.

Figure 4.1 shows the distribution of the per-venue number of reviews of Miami-Dade venues, with a logarithmic y scale. It shows a long tail distribution, with around 2000 venues having 1 review but only 1000 venues having 2 reviews. We emphasize the low number of venues without reviews - only 177. Figure 4.2 shows the distribution of the number of venues with an aggregated rating ranging between 1 and 5: Yelp reviews are mostly positive as most aggregate ratings are at or above 4 stars.

4.2.2 Crime Data

We use a historical database of more than 2.3 million crime incidents reported in the Miami Dade county area since 2007 [Ter]. Each record is labeled with a crime type (e.g., homicide, larceny, robbery, etc), the time and the geographic location where it has occurred. We briefly document two problems we encountered when pre-processing this data. First, since records come from different Police departments, the crime type labels are non-uniform, (e.g., *murder* in Miami Beach vs. *homicide* in North Miami). Second, crime reports include many minor incidents (e.g., fire alarms issues), resulting in over 140 different crime types.

In order to standardize and eliminate ambiguities, we mapped crimes into 7 categories: Murder, Forcible Rape, Aggravated Assault, Robbery, Larceny/Theft, Burglary/Arson, Motor Vehicle Theft. We removed minor crime reports that did not fall into these categories. Due to the large number of records in the database, manual mapping was infeasible. Instead, we have experimented with two machine learning techniques for classifying each record: the Naive-Bayes (NB) classifier and the Decision Trees (DT) classifier [TSK05a]. In order to build our training and test sets, we manually annotated a random sample of 2000 records from different police departments. Then, we split this subset of records into training and test datasets,

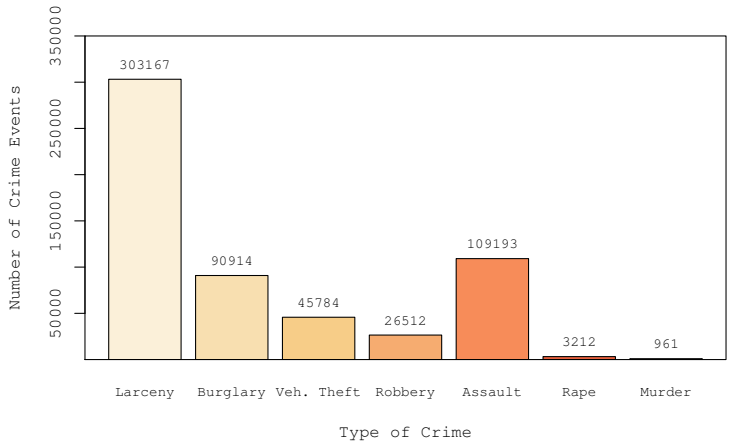


Figure 4.3: Distribution of number of crime events per type of crime. Outcome of DT classifier.

each containing 1000 records. We built our classifiers using the NLTK library [NLT]. The accuracy was measured using a simple metric that measures the percentage of inputs in the test set that the classifier correctly labeled. For instance, a crime type classifier that predicts the correct crime type 60 times in a test dataset containing 100 crime types, would have an accuracy of 60%. On our crime dataset, the NB classifier achieved an accuracy of 91% and the DT classifier an accuracy of 98%. Thus, we have used the outcome of the DT classifier. Figure 4.3 shows the crime set's distribution of the crime categories following the DT classification.

Let c denote the number of crime types. In our case, $c = 7$. Let $\overline{CT} = \{CT_1, .., CT_c\}$ denote the set of crime types.

We use Census data sets [Cen10], reporting population counts and demographic information. The data is divided into geographical extents e.g. polygons, called *census block groups*. Each block contains information about the population within (e.g., population count, various statistics). According to the data, Miami Dade

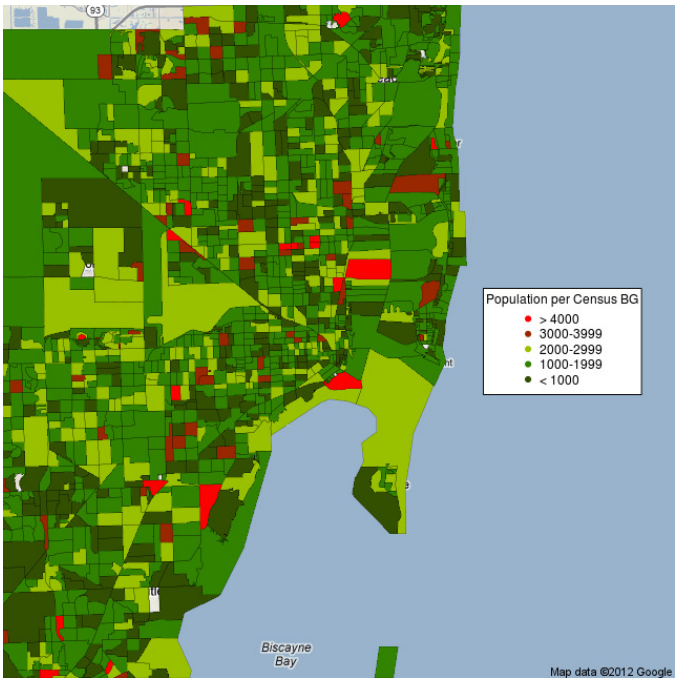


Figure 4.4: Miami-Dade county: geographical distribution of population. Polygons represent Census Block Groups.

county has a population of 2, 496, 435. Figure 4.4 shows the geographical distribution of the population in the Miami Dade county.

4.2.3 Forecasting Tools

We describe here several time series forecasting tools that we use in our work.

ARIMA Model. ARIMA models incorporate autoregressive (p),integration(d) and moving average terms(q) to provide higher fitting and forecasting accuracy. ARIMA uses the input data to determine the appropriate model form. The ARIMA forecasting procedure consists of four steps [TT02], (1) identifying the ARIMA(p, d, q) structure, (2) estimating the unknown parameters, (3) fitting tests on the estimated residuals and (4) forecasting future outcomes based on historical data.

Linear (Double) Exponential Smoothing (LES) Model. Brown's linear (double) exponential smoothing [Nau] includes trend variations of the time series without

a significant seasonal component. The process is controlled by a smoothing parameter α whose value ranges between 0 and 1. α decides the weight placed on the most recent observations during the forecast process. We determine the value of α by minimizing the root mean squared error (RMSE) [HA01] from one step-ahead forecasts and repeating the process for all forecast values.

Artificial Neural Network (ANN). ANNs are data-driven self-adaptive methods that learn and generalize from experience and capture subtle functional relationships among the empirical data even if the inherent relationships are unknown or difficult to describe. In this chapter, we focus on the multi-layer perceptrons (MLP) ANN model, which is particularly suitable for forecasting, due to its ability for input-output mapping. The ANN we consider consists of an input layer (of the same size as the input vector), two layers of hidden nodes and an output layer providing the forecast value. Before the training phase, we normalize the input data to a $(-1, 1)$ range; following the prediction step we map the output back to the initial range. For the training phase we use a multilayer feedforward network trained using back propagation and the Levenberg-Marquardt algorithm to perform function fitting (nonlinear regression).

Error Measurement. We use the root mean squared error (RMSE) and mean absolute percent error (MAPE) [HA01] as error measurements to evaluate the accuracy of different models. MAPE can be easily affected by the magnitude of the series but it does provide information about the relative magnitude of the forecast error. On the other hand, RMSE is a more objective measure in absolute magnitude. Thus, in our evaluation, the RMSE is used as the primary and MAPE as the secondary accuracy measure.

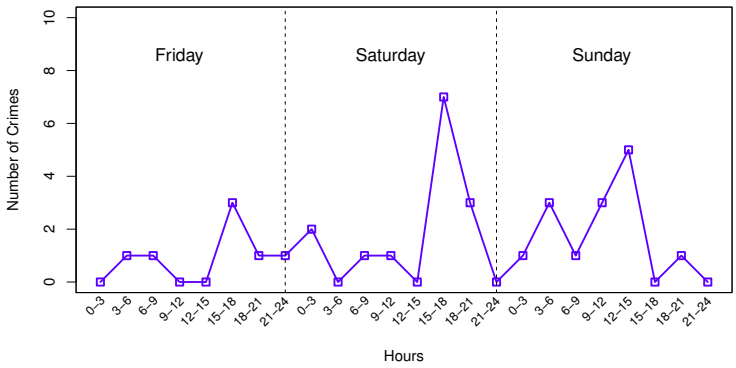


Figure 4.5: Three day evolution of the number of crimes reported within one Miami-Dade block.

4.2.4 Attacker Model

We assume a semi-honest, or honest-but-curious service provider. That is, the service provider is assumed to follow the protocol correctly, but attempts to learn as much user information as possible. We assume users can be malicious. However, each participating user needs to install a provider-signed client application.

4.3 Location Based Safety

We exploit the crime dataset to define an initial, location-centric safety metric. We divide space into census blocks. We divide time into fixed-length epochs, e.g., 1 hour long, 24 epochs per day. To understand the need for a time dependent safety metric, we have studied the evolution in time of crimes reported within blocks of the Miami-Dade county. Figure 4.5 shows the evolution over three consecutive days (Friday-Sunday, July 15-17, 2011) of the number of crimes reported within one such block, with a 3 hour time granularity. Most of the events are larcenies. The plot shows a significant variance in the number of crimes reported throughout a day,

Crime Type	Weight
Assault	0.176
Robbery	0.180
Rape	0.307
Homicide	0.336

Table 4.1: Crime weight assignment using the FCPC.

with a spike between noon and 6pm. Thus, a fixed aggregate of past crime events is unlikely to accurately define the present.

Block crime and safety indexes. For a census block B and an epoch e denoted by the time interval ΔT, let $C(B, \Delta T)$ represent a c-dimensional vector, where the i-th entry denotes the number of crimes of type $CT[i]$ recorded in block B during interval ΔT. Let $\overline{W}$ denote a c-dimensional vector of weights; each crime type of $\overline{CT}$ (defined in Section 4.2.2) has a weight proportional to its seriousness (defined shortly). Let $BC(\Delta T)$ denote the population count recorded for block B. We then define the *crime index* of block B during interval ΔT as

$$CI(B, \Delta T) = min\{\frac{C(B, \Delta T)\overline{W}}{BC(\Delta T)}, 1\} \tag{4.1}$$

where $C(B, \Delta T)\overline{W}$ denotes the vectorial product between the number of crimes per type and the weights of the crime types. That is, B's crime index is the per-capita weighted average of crimes recorded during interval ΔT. The safety index SI of block B during interval ΔT is then defined as

$$SI(B, \Delta T) = 1 - CI(B, \Delta T) \tag{4.2}$$

Both the CI and SI metrics take values in the [0, 1] interval. Higher $SI(B, \Delta T)$ values denote safer blocks.

Crime weight assignment. We need to assign meaningful weights to the crime types $\overline{CT}$. An inappropriate assignment may make a large number of "lighter" of-

fenses overshadow more serious but less frequent crime events, (e.g., consider larcenies vs. homicides). Assigning weights to crime types is also a subjective matter: certain people are more likely to be vulnerable to certain crime categories. In the following, we restrict our definition of safety to crimes against persons e.g., assault, robbery, homicide and rape and ignore crimes against property. Although our model can be applied to both categories, the focus of this work is on personal safety.

We propose to assign each crime type a weight proportional to its seriousness, defined according to the criminal punishment code, i.e., the Florida Criminal Punishment Code (FCPC) [oC]. The FCPC is divided into *levels* ranging 1-10, and each level L_k contains different types of felonies. The higher the level, the more serious is the felony. Each felony has a *degree*, (i.e., capital, life, first, second and third degree, sorted in decreasing order of seriousness), with an associated punishment (years of imprisonment) [Hor].

Let L_k denote the set of felonies within level k and let P_k denote the set of corresponding punishments. Let $l_k = |L_k|$ denote the number of felonies within level k. Then, we define the weight of crime type $CT[i]$, $\overline{w}_i$, as

$$\overline{w}_i = \sum_{k=1}^{10} \rho_k \frac{P_k[i]}{\sum_{j=1}^{l_k} P_k[j]},$$

where $\rho_k = k / \sum_{i=1}^{10} i$ is the weight assigned to level k (normalized to the sum of the number of levels). The weight of crime type $CT[i]$ is the weighted sum of the per-level punishment value ($P_k[i]$) associated with the occurrence of $CT[i]$ within the felonies of level k, normalized to the total punishment of level k. Table 4.1 shows the resulting weights.

Example. We exemplify the impact of level L_8 on the weight of the "Robbery" crime. Out of the felonies represented on level 8, two are related to "Robbery": "Robbery with a weapon" and "Home-invasion robbery". Both are first degree

felonies, therefore punishable with up to 30 years of imprisonment. The other represented felonies are "Homicide", with 6 different counts, for a total of 135 years penalty and "Rape", with 1 count of up to 15 years penalty. Thus, the contribution of level 8 to the weight of "Robbery" is $\frac{8}{55} \times \frac{60}{60+135+15} = 0.0415$.

Illustration. We use the Miami-Dade crime set to illustrate the geographic distribution of block-level safety index information, where the epoch, denoted by the interval ΔT, is the year 2010. We use the census dataset to extract the population count $BC(\Delta T)$. Figure 4.6 shows the color-coded safety index for each block group in the Miami-Dade county (FL) where crimes have been reported during 2010. The safety index considers only crimes against persons. Blocks without color have a very low reported crime level. Green blocks denote safer locations while darker yellow and red blocks denote areas with more reported crimes.

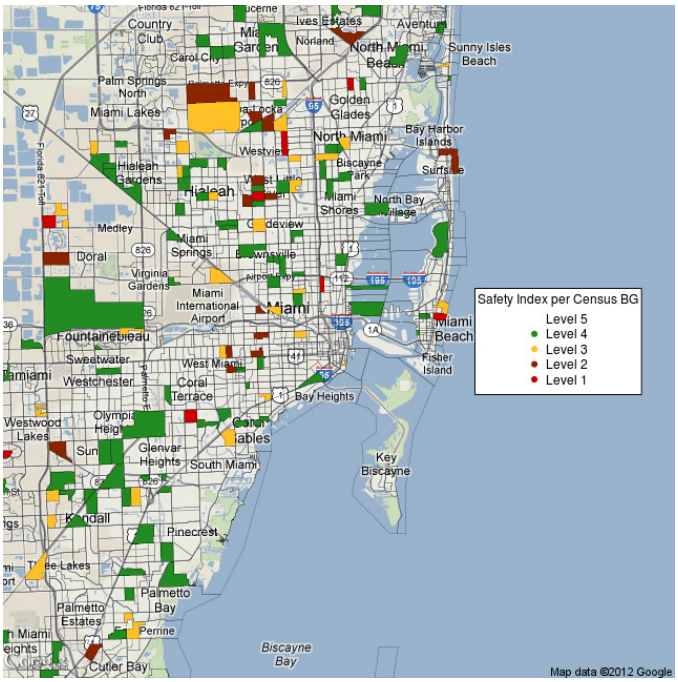


Figure 4.6: Safety index illustration for the Miami-Dade county: $SI(B, \Delta T)$ values are mapped into color-coded "safety levels".

4.4 Predicting Safety

The crime index computation of Equation 4.1 can only be performed for past epochs, when all crime events have been reported. Safety information however is most useful when provided for the present or near future. One way to compute the predicted crime index of a block B for the next epoch denoted by the interval ΔT, $PCI(B, \Delta T)$, is the average crime index of the block during the same epoch in the day for the past d days, where d is a system parameter (e.g., d=7 for 1 week of recorded per-block history). This solution however is unable to detect and factor in all crime periodicities, including seasonal, weekly and daily fluctuations. As such, it may include unnecessary errors – e.g., higher number of crimes in a past August may introduce inaccuracies in the crime index considered in the current month of April.

We propose to address this issue through the use of the time series forecasting techniques discussed in Section 4.2.3. Specifically, we use time series forecasting tools to compute long and short term predictions of the number of crimes to be committed within an area (e.g., census block, zipcode, city, etc), based on the area's recorded history.

Predicting crime and safety indexes. At the beginning of each epoch (denoted by the time interval ΔT), we compute predictions for the number of crimes of each crime type to be committed at each census block B during the epoch. Let $PC(B, \Delta T)[i]$ denote the predicted number of crimes of type $CT[i]$. Using a formula similar to Equation 4.1 we compute the predicted crime index for B during interval ΔT as $PCI(B, \Delta T) = min\{PC(B, \Delta T)\overline{W}/BC(\Delta T), 1\}$. The predicted safety index is then $PSI(B, \Delta T) = 1 - PCI(B, \Delta T)$.

4.5 Personalized, Context-Aware Safety

The ultimate goal of defining crime and safety indexes is to provide users with safety advisory information. People are however not equally exposed and vulnerable to all crime types. Age, gender and an array of personal features, preferences and choices play a central role in the perception of an individual's safety. Since such information may not be readily accessible, we use instead the localization capabilities of a user's mobile device to periodically record and locally store her trajectory trace. This enables us to define the crime index level with which a user is comfortable: the average crime index of the locations in her trajectory. When enough crime information exists to enable the prediction of the near-future crime index of a location, we introduce the concept of *personalized safety*: the user is safe if her comfortable crime index level equals or exceeds the predicted crime index of her current location.

However, crime information is not always available or detailed enough to allow a confident prediction of location crime index values. For instance, as shown in Figure 4.5, the number of recorded events can quickly switch between 0 and 1 in successive intervals. Accurately predicting event counts within a short time interval is difficult, as the difference between 0 and 1 crimes is significant.

We propose to address this issue, by exploiting the intuition that the safety of a place depends not only on its history but also on its current context. One way to define the context of a place at a given time is through the people located there at that time (in Section 4.6 we show how geosocial network data can be used to construct context). We use the trajectory trace of the user to define the chance of a crime to occur around the user and generalize this approach to compute the chance of a crime to occur around groups of users. We then introduce the concept of *context aware safety*: a user is safe if the chance of a crime to occur around her

equals or exceeds the chance of a crime to occur around the other users currently co-located with her.

We take advantage of the wireless communication capabilities of user mobile devices to form short lived, ad hoc communities with co-located devices and use them to aggregate the trajectory information of their users. Since user trajectories are sensitive information, we introduce iSafe, a distributed algorithm that allows the aggregation of trajectory traces of co-located users while preserving the privacy of involved participants.

4.5.1 Personalized User Safety

We extend the crime and safety index definitions from locations to users. We assume the user's device can capture the user's location, e.g., using GPS or a combination of celltower and Wi-Fi access point localization techniques. We assume a block level localization precision. Let $TJ_U = \{[B_i, T_i, CI(B_i, \Delta T_i)] | i = 1..h\}$ denote the trajectory trace of user U, consisting of recorded [block, epoch, crime index] tuples. ΔT_i denotes the epoch encompassing time T_i when U was present at block B_i, $T_i \in \Delta T_i$. For privacy reasons, we require each user to store her trajectory trace on her device.

We define the *vicinity crime* metric for a user U, V_U to be the percentage of the user's trajectory places where crimes have been reported around the time of her visit:

$$V_U = \frac{\sum_{i=1}^{h} sgn(CI(B_i, \Delta T_i))}{h} \tag{4.3}$$

$sgn(x)$ denotes the sign function, that is 0 when x is 0 , and 1 when x is larger than 0. For instance, if a user has 100 locations in her trajectory and crimes have been reported at 60 of those locations during the epoch of the user's presence, the user's

vicinity crime metric is 60%. We then define the crime index of a user U to be the average crime index of locations in her trajectory:

$$CI_U = \frac{\sum_{i=1}^{h} CI(B_i, \Delta T_i)}{h} \tag{4.4}$$

Safety Decision With Accurate Crime Data

When user U is located at time T_c in a block B, where accurate past crime data exists, allowing the proper prediction of the crime index, we compute the predicted crime index $PCI(B, \Delta T)$, as specified in Section 4.4, where ΔT denotes the current epoch, $T_c \in \Delta T$. We then introduce the notion of personalized safety recommendation:

Definition 4.5.1 *(Personalized safety). A user U is safe at a block B within time interval ΔT, if $CI_U \geq PCI(B, \Delta T)$.*

Intuition. A user is safe if the user's crime index equals or exceeds the block's crime index predicted for the duration of the user's presence. If the crime index of the user's current block, predicted for the epoch of the user's presence, does not exceed the user's level of comfort, it means the user has spent at least half of her time in locations with more crime than the current location. Thus, the user is likely to be comfortable with the crime level of her current location.

Safety Decision Without Accurate Crime Data

Certain locations may have insufficient crime data to ensure an accurate prediction of the location's crime index. This is the case also during unexpected events (natural and man made disasters) when the future does not reflect the past. To address

this issue, we propose to use existing context information, collected from co-located users. To achieve this, we exploit ad hoc networks established by devices of co-located users.

Our approach is the following. We define the safety index of a user U to be the chance of no event being reported in her vicinity: $SI_U = 1 - V_U$. Let $U_1, .., U_k$ be the users co-located with user U. We define a *super user* $SUP_{1..k}$, as a fictitious user whose trajectory trace encompasses the trajectories of users $U_1, .., U_k$. That is, $TJ_{U_{1..k}} = TJ_{U_1} \cup .. \cup TJ_{U_k}$. We note that both users and super users can be located in multiple blocks during the same epoch. We then use Equation 4.3 to compute the vicinity crime metric of $SUP_{1..k}$, $V_{SUP_{1..k}}$. We define the safety index, $SI_{SUP_{1..k}} = 1 - V_{SUP_{1..k}}$. These definitions enable us to introduce the notion of personalized safety recommendation:

Definition 4.5.2 *(Context-aware safety). A user U is safe in a context consisting of neighboring users $U_1, .., U_k$, if $SI_U \leq SI_{SUP_{1..k}}$, i.e., $V_U \geq V_{SUP_{1..k}}$.*

That is, the user is safe if it is surrounded by users whose aggregated safety index is higher or equal to the user's safety index.

Intuition. The safety index of a user encodes the chance of no event occurring around the user. The safety index of a group of users (e.g., $SUP_{1..k}$) is defined as the chance of no event occurring around the group. Definition 4.5.2 states that a user is safe if it is surrounded by a group of users whose aggregated chance of no event occurring is higher or equal to the user's chance of no event occurring. A low safety index value does not imply the user is unsafe, but merely the fact that the user spends time in places where events do occur. If the location sampling process is done periodically, the formula naturally ensures that blocks where the user spends

more time have more impact on the user's safety index. Being around a group of users whose aggregated safety index is low suggests that the place is likely to have a low safety level.

4.5.2 iSafe

One question that remains to be answered is how can the above decisions be made without requiring participating users to provide sensitive location traces and safety index values. To answer this question, we introduce iSafe, a protocol that implements the above solution, in a privacy preserving aware fashion. iSafe consists of a main procedure, $C.safetyDecision(B, \Delta T)$, executed periodically by C, at the C's user current block B.

Definition 4.5.3 *(Location Privacy)*
Let an adversary $\mathcal{A}$ control the service provider S and any number of clients, such that the number of clients controlled by $\mathcal{A}$ at any location is at most $NThr - c$, where $NThr$ and $c > 1$ are integers. The challenger $\mathcal{C}$ controls a client C. $\mathcal{A}$ contacts $\mathcal{C}$ at any time T. $\mathcal{C}$ invokes $C.safetyDecision(B, \Delta T)$, where B denotes C's current block and $T \in \Delta T$. $\mathcal{A}$ outputs B', its guess of the block B where C is located. We say a solution provides location privacy if the advantage of $\mathcal{A}$ in this game, $Adv_{\mathcal{A}} = |Pr[B' = B] - 1/n|$ is negligible.

Algorithm 2 shows the pseudocode of iSafe. In a first step, the client C installed on the wireless-enabled mobile device of a user contacts the service provider S, storing the crime and Census datasets. C retrieves the predicted crime index of the block B where the user is located (line 12). This operation is performed privately,

Algorithm 2 iSafe pseudocode.

```
1.Object implementation iSafe;
2.  neighbor[] N;            #set of neighbors
3.  double CI, SI;           #crime, safety indexes
4.  double V;              #vicinity crime prob
5.  BigInteger R;            #random value
6.  BigInteger[] shares;       #set of shares
7.  BigInteger[] NShares;      #shares of neighbors

8.  int BWC;                 #blocks with crime
9.  int TBlk;                #total blocks visited

10. Operation int safetyDecision(Epoch ΔT)
11.     B := getCurrentBlock();
12      PCI_B := S.getPCI(B, ΔT);
13.     if (PCI_B! = −1) then return (CI ≥ PCI_B);
14.     else return cas(); fi
15. end

16. Operation int cas()
17.     N := discoverNeighbors();
18.     if (N.size < NThr) then return − 1;
19.     BWC_SUP := multiPartySum(0) − BWC;
20.     TBlk_SUP := multiPartySum(1) − TBlk;
21.     return(V ≥ BWC_SUP/TBlk_SUP);
22. end

23. Operation BigInteger multiPartySum(int type)
24.     R := getRandom();
25.     shares := split(R, N.size);
26.     for i := 1 to N.size do
27.         send(N[i], shares[i]);
28.         NShares[i] := recv(N[i]); od
29.     int order := electLeaderOrder();
30.     BigDecimal S := 0; int count := 0;
31.     while (count < N.size) do
32.         count := count + 1;
33.         if (count = order) then
34.             if (type = 0) then S := S + BWC + R;
35.             else S := S + TBlk + R; fi
36.             for i := 1 to |N| do S := S − NShares[i]; od
37.             mcast(S);
38.         else S := recv(); fi
39.     od
40      return S;
41. end
```

without the client leaking its location trace, by using a private information retrieval technique [Gas04].

If the crime index of the block can be accurately predicted (line 13), the operation returns the decision of Definition 4.5.1. Otherwise, it invokes the *cas* operation (line 14). *cas* first discovers all the ad hoc neighbors of the user (line 17). If the number of neighbors is below a system-wide threshold value, $NThr$, it returns -1: not enough information exists to perform an accurate decision. Otherwise, it invokes the $multiPartySum$ operation twice, with different input arguments (lines 19-20). When invoked with argument 0, $multiPartySum$ calculates BWC_{SUP}, the sum of the blocks with crimes visited by all the user's neighbors. When invoked with argument 1, $multiPartySum$ calculates $TBlk_{SUP}$, the sum of the total blocks visited by all the user's neighbors. Thus, the ratio of BWC_{SUP} and $TBlk_{SUP}$ generates the vicinity crime metric of the super user representing the user's neighbors. In line 21, *cas* returns the safety decision of Definition 4.5.2.

The $multiPartySum$ operation is a secure multi-party sum evaluation. It achieves privacy through the use of (i) frequently changing, random MAC addresses for user devices and (ii) secret splitting. Each client generates a random value (line 24) and splits it into shares – one for each neighbor. That is, if the random value is R, the shares $sh_1, .., sh_k$ are generated randomly such that $\sum_{i=1}^{k} sh_i = R$. The client sends each share to one neighbor (lines 26-27) and receives a share from each neighbor (line 28). The clients engage in a leader election and order selection distributed algorithm (line 29), where each client is assigned a unique identifier, between 1 and k.

When a client's turn comes, according to the order established, it adds either the user's BWC value (number of census blocks with events visited by the user) or the user's TBlk value (total number of blocks visited), according to the input variable

type, and adds its random value R to the overall sum (S), (lines 34-35). It then subtracts all the shares of secrets of its neighbors (line 36) and sends a multicast of the result (line 37), reaching all its neighbors. If it's not the user's transmission turn, the client blocks to receive the multicast values of its neighbors (line 38).

4.5.3 Analysis

We now prove the following results.

Theorem 4.5.4 *An adversary $\mathcal{A}$ controlling $k-c$ out of k participants in the iSafe algorithm, can only find the sum of the input values (BWC or Tblk) of the remaining c honest participants.*

Proof. Secret splitting is information theoretical secure: Without knowing *all* the shares of a secret, no information can be inferred about the secret. The adversary $\mathcal{A}$ has access to all intermediate values multicast in Algorithm 2, as well as $k-c$ shares of the secret of each of the remaining c honest participants. Let R_i denotes the random value of the i-th (honest) participant and let $s_{1i}, s_{2i}, .., s_{ki}$ be the shares received by that participant from all the other participants. Then, the sum $R_i + s_{1i} + s_{2i} + .. + s_{ki}$ is random and cannot be predicted by $\mathcal{A}$: $\mathcal{A}$ only controls $k-c$ shares of R_i (out of $k-1$ shares), but not R_i, thus the other c values in the sum are random and not under the control of $\mathcal{A}$. Thus, $\mathcal{A}$ cannot infer the value (BWC or TBlk) of user i by comparing the value of S before and after user i's multicast.
□

Theorem 4.5.5 *iSafe provides location privacy.*

Proof. (Summary) The adversary $\mathcal{A}$ can only access user location information from (i) user trajectory traces, (ii) queries made by iSafe (Algorithm 2 line 12) and (iii) during computations of the aggregate super user crime and safety indexes (the $multiPartySum$ operation).

For the first point, we observe that user trajectories are only stored on the the user's mobile devices and are never shared with other participants. For the second point, the queries made by users in iSafe to $\mathcal{A}$ are private, e.g., use PIR (see Section 4.5.2). Thus, $\mathcal{A}$ cannot learn the location of the user with a probability non-negligible higher than $1/n$, where n is the number of census blocks, without breaking the security of the PIR solution employed. The third point's implicit requirement is that the provider colludes with users in order to learn information about their neighbors. The use of random, frequently changing MAC (or physical device) addresses by participating devices prevents however even such a powerful adversary from linking a device identifier to a user, thus linking a user to a location. Moreover, Theorem 4.5.4 shows that if $\mathcal{A}$ controls at most $NThr - c$ clients at any location where at least $NThr + 1$ clients are located, $\mathcal{A}$ can only learn the sum of the secret values of the remaining (at least c+1, $c > 1$) honest clients. □

4.5.4 Attacks and Defenses

Safety profiles of co-located users are aggregated to obtain a safety image of locations. Since that image impacts user decisions, it can become the target of malicious attacks. For instance, malicious users may attempt to incorrectly (i) improve the safety of desired locations, for instance to attract unsuspecting users to unsafe locations or to (ii) decrease the safety image of target locations. We now describe several mechanisms that could be exploited to perform these attacks, and suggest defenses.

Reporting incorrect locations. Malicious users may report incorrect locations, corresponding to safe areas. Even with GPS verification mechanisms in place, committing location fraud has been largely simplified by the recent emergence of specialized applications for the most popular mobile eco-systems (LocationSpoofer [Bos11] for iPhone and GPSCheat [GPS] for Android). To prevent this attack, location verification mechanisms can be used [CP12, SW09, ZC11]. For instance, Carbunar et.al. [CP12] has developed venue-centric location verification techniques, that rely on devices installed by venue owners within their venues. In the scenario considered in this chapter, the owners' incentive for participation is to prevent the tampering of the safety image of their neighborhood.

Turning off devices in unsafe areas. Users could turn off their iSafe application when entering bad areas. While we cannot prevent this behavior, we propose to use rewards and game mechanics to encourage people to report their location. For instance, users gain points for each reported location, perhaps more for the occasional unsafe location. Points are used to acquire badges, similar in principle to those used by geosocial networks like Foursquare [fou] or Yelp [Yel].

4.6 Geosocial Network Extensions

Geosocial networks, with their emphasis on the location of both users and venues, seem ideal candidates for augmenting spatiotemporal context. We first investigate relations between crimes and geosocial networking activities. We then propose to use geosocial network user location trajectories to improve the accuracy of iSafe.

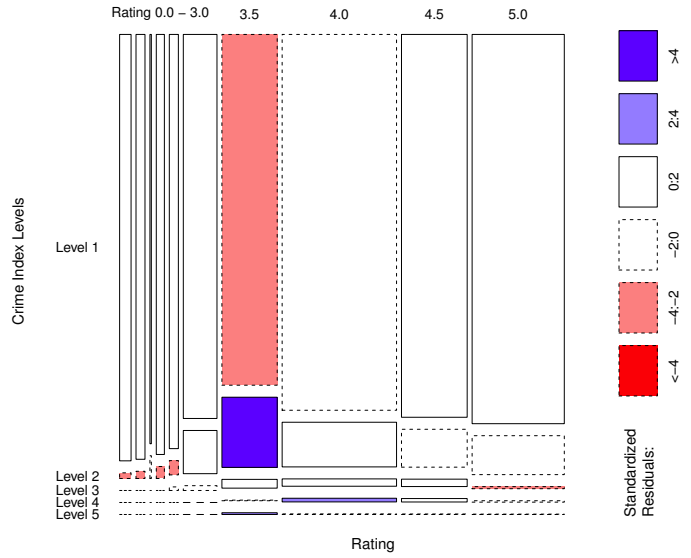


Figure 4.7: Relation between venue ratings and the crime index (CI) levels of their location.

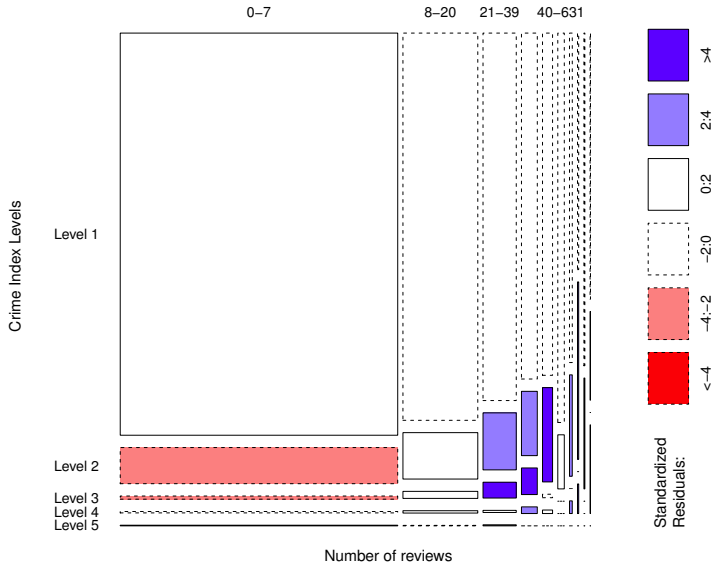


Figure 4.8: Relation between the number of reviews received by a venue and the crime index (CI) level of its block.

4.6.1 Crime vs. Geosocial Activity Dependencies

We conjecture that the crime activity recorded at a location has a bearing on the quality and quantity of reviews recorded at nearby venues. We investigate this hypothesis through the combination of review data we collected from Yelp and the Miami-Dade crime dataset.

One question we need to answer is whether there exists a relation between the rating of a venue and the safety of its location. For this, we first mapped each venue in the Miami-Dade county to its corresponding census block, then computed Crime Index (CI) values for each block using the crime events of 2011. We need to test for dependencies between two different mixed variables, (i) categorical user ratings and (ii) continuous CI values. Since, linear regression or any other method for continuous variables are not ideal, we discretized the CI variable into 5 levels, using 1-dimensional k-means (k set to 5), that guarantees optimal partitioning for one-dimensional data.

We have then built a contingency matrix, by grouping the venues according to their ratings and assigning them to their corresponding CI level: each cell in the contingency matrix contains the number of venues that have the corresponding user rating and belong to a block having the corresponding CI level. We have used the χ^2 test to test the dependency between the two categorical variables [TD00]. We used the R [R D11] package to compute the χ^2 test and we obtained the p-value, or the observed level of significance, and corresponding standard residuals. In short, the standard residuals indicate the importance of the cell to the ultimate χ^2 value; by comparing standard residuals, one can easily identify the cells that contribute the most to the χ^2 test. Since the observed level of significance is extremely low

(very close to zero) we reject the null hypothesis and therefore we conclude that there exists a dependence between CI values and user ratings.

Figure 4.7 shows the corresponding mosaic plot, displaying the relationship between ratings and CI values: the areas of the rectangles are proportional to the probabilities of the user ratings and to the conditional probabilities of the CI levels. It shows that the bulk of the Yelp venues (even low rated ones) are in places where crime levels are low. This can be due to the fact that the distribution of the venues per CI values is long tail, which may be further explained by the fact that (i) in the Miami-Dade county there are few areas with high crime levels and (ii) Yelp is not popular in those areas - people may not even report venues located there in Yelp. Moreover, as shown in Figure 4.2, Yelp ratings are biased toward higher values.

A second question is whether there exists a relation between the number of reviews a venue receives and the safety of the venue's location. Once again, even though the number of reviews is not a categorical variable, it is discrete. Therefore, we tested their association with CI values using the χ^2 test. We created review count interval buckets and we assigned each venue to one bucket according to its number of reviews. We computed the range of the intervals using the 1-dimensional k-means algorithm with k set to 10. The χ^2 test produced a corresponding p-value very close to zero, thus answering our question in the affirmative. Figure 4.8 shows the corresponding mosaic plot of this experiment. It confirms that most Yelp venues are located in safe areas as well as the long tail distribution of the number of reviews per venue in Yelp, shown in Figure 4.1.

In order to identify the sources of the dependencies, we studied a specialized view of this data - the relationship between review counts and crime types (see Section 4.2.2). One finding is depicted in Figure 4.9, showing the relationship between reported rapes and review counts: rapes occur more frequently in places with low

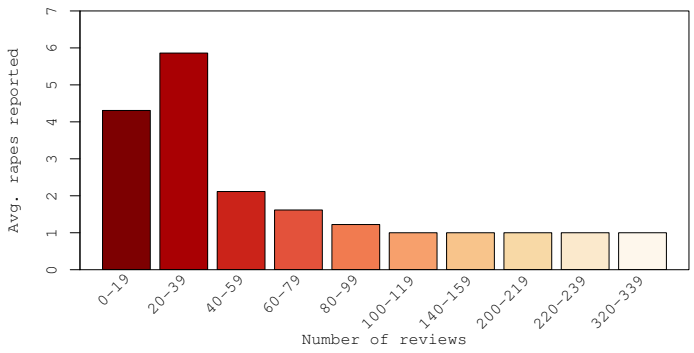


Figure 4.9: Number of rapes per number of venue's reviews. Locals and visitors.

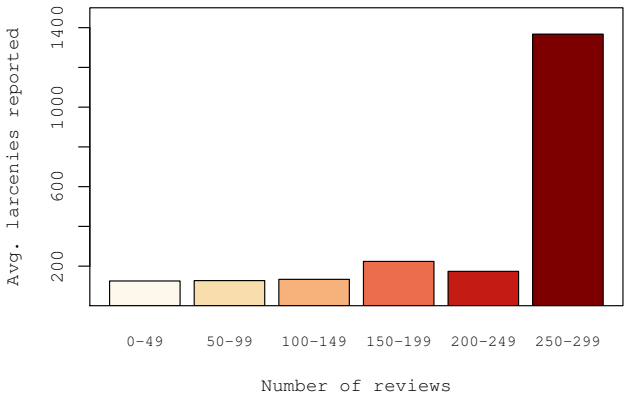


Figure 4.10: Number of larcenies/thefts per number of venue's reviews.

number of reviews. Furthermore, we study the relation between crime types and the number of reviews received from visitors vs. locals. This information is publicly available, as Yelp users need to specify a home city/state. Figure 4.10 shows that the number of larcenies is high around venues with many local reviews. A potential explanation is that local yelpers (Yelp users) are more likely to choose venues in good neighborhoods, and good neighborhoods are more likely to attract thieves.

4.6.2 Geosocial iSafe

We propose to extend iSafe with geosocial network information. For each geosocial network user U, we define the trajectory trace $TJ_U = \{[B_i, \Delta T, CI(B_i, \Delta T_i)] | i = 1..h\}$. Each TJ_U record consists of (i) the block containing a venue where U has written a review, (ii) the time epoch ΔT when the user wrote the review and (iii) the crime index of the block during that epoch. In Yelp, the timestamps associated with reviews have a 1-day granularity, thus, ΔT is 1-day long.

While geosocial network user trajectories are likely to be more sparse than those collected from mobile devices, their similar definition enables us to use Equations 4.3 and 4.4 to compute the user's vicinity crime metric and crime index values. Furthermore, we use the vicinity crime metric and crime index values of users who wrote reviews for a Yelp venue to compute aggregate venue crime index and vicinity crime values, using the mobile version of iSafe (see Algorithm 2). These definitions allow us to extend the personalized context aware safety decisions of Section 4.5.1.

4.7 iSafe Implementation

We implemented iSafe as a (i) web server, (ii) a browser plugin running in the user's browser and (iii) a mobile application. We use Apache Tomcat 6.0.35 to

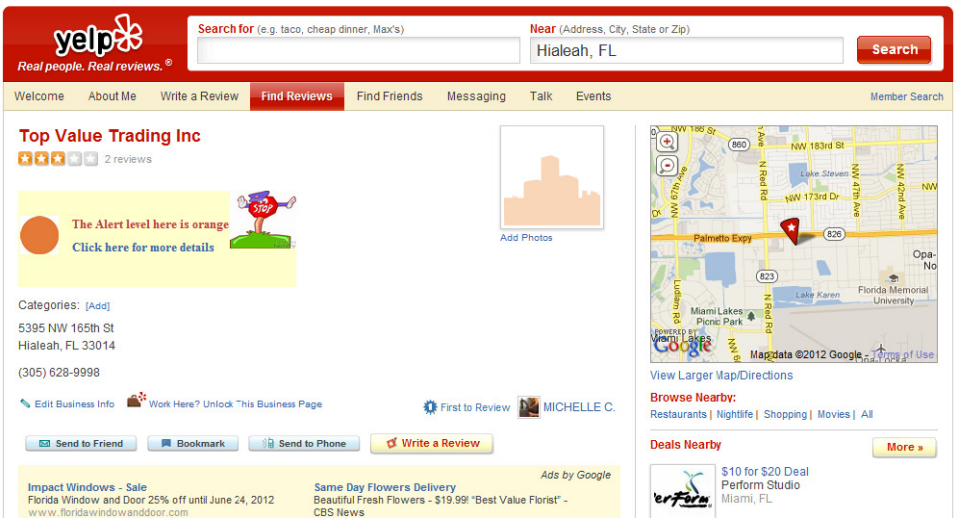


Figure 4.11: Snapshot of iSafe's plugin functionality for a Yelp venue.

route requests (exposed to the client through a REST API interface) to our server-side component. The server-side component relies on the latest servlet v3.0 which offers additional features including asynchronous support, making the server-side processing much more efficient.

4.7.1 Browser Plugin

We implemented the browser plugin for the Chrome browser using HTML, CSS and Javascript. The plugin interacts with Yelp pages and the web server, using content scripts (Chrome specific components that let us access the browser's native API) and cross-origin XMLHttpRequests. If our content script receives content from another web site, it inspects it for cross-site scripting attacks before injecting the content into the current page (e.g., to protect the user from a hijack attack). To store and process review and user data for each venue, we use the SQLite 3.7.12.1 as the DB server.

The idea behind the browser plugin is to extend the experience of geosocial networks like Yelp [Yel] with safety information. Specifically, the browser plugin

(a) (b)

Figure 4.12: Snapshots of iSafe on Android.

becomes active when the user navigates to a Yelp page. For user and venue pages, the plugin parses their HTML files and retrieves their reviews. We employ a stateful approach, where the server's DB stores all reviews of pages previously accessed by users. This enables significant time savings, as the plugin needs to send to the web server only reviews written after the date of the last user's access to the page. The initial access is likely to be slower, requiring the plugin to access multiple pages of reviews.

Given the venue's set of reviews, the server determines the corresponding reviewers. Since we do not have access to the location trajectories of users, to compute a user's security label we rely on the venues reviewed by the user: The user safety is computed as an average over the safety labels of the blocks containing the venues reviewed by the user. Given the safety labels of reviewers, we determine the safety level of the venue. The server sends back the safety level of the venue, which the plugin displays in the browser. Figure 4.11 shows iSafe's extension to the Yelp page of the venue "Top Value Trading Inc." in Hialeah, FL (central left yellow rectangle containing iSafe's safety recommendations).

4.7.2 Mobile iSafe.

We have implemented the location centric static safety labeling component of iSafe for a mobile application using Android. We used the Android Maps API to facilitate the location based service employed by our approach. We represent safety using five color labels ranging from green (safe) to red (unsafe).

We used the SQLite version 3.4.0 database to store the trajectory trace of the user, along with timestamps, on her smartphone. The database also caches the Census block structure and associated safety indexes for the city where the user is located. This ensures both (i) privacy – the user trajectory and her requests for block safety indexes never leave her phone and (ii) performance – frequent block safety index requests are performed locally, while infrequent census block safety index updates are performed periodically to ensure an accurate copy of the device's cache.

Whenever a user starts the iSafe app, iSafe retrieves the user's current geolocation, derives the current census block and also the corresponding crime index. iSafe stores the user's trajectory as one record $[block, time, crime\_index]$ in the SQLite database. The initial threshold values for creating a new record are 60 seconds. iSafe uses an exponential backoff algorithm [KSM05] coupled with accelerometer data to ensure that a static device does not consume battery power on GPS queries. iSafe updates then the user's current crime index and vicinity crime values.

iSafe uses Bluetooth [SIG01] to compute the vicinity crime metrics for the user's neighbors. We implemented a client-server Bluetooth communication protocol where each device acts as a server and other connected devices act as clients per P2P communication. Bluetooth is a packet-based protocol with a master-slave structure in which one master may communicate with up to 7 slaves in a piconet [SIG01].

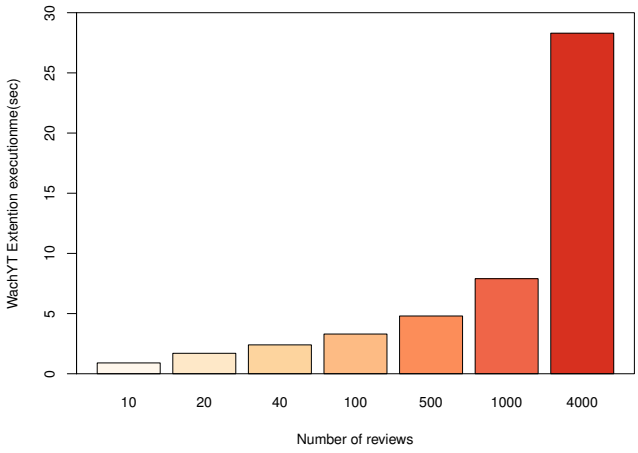


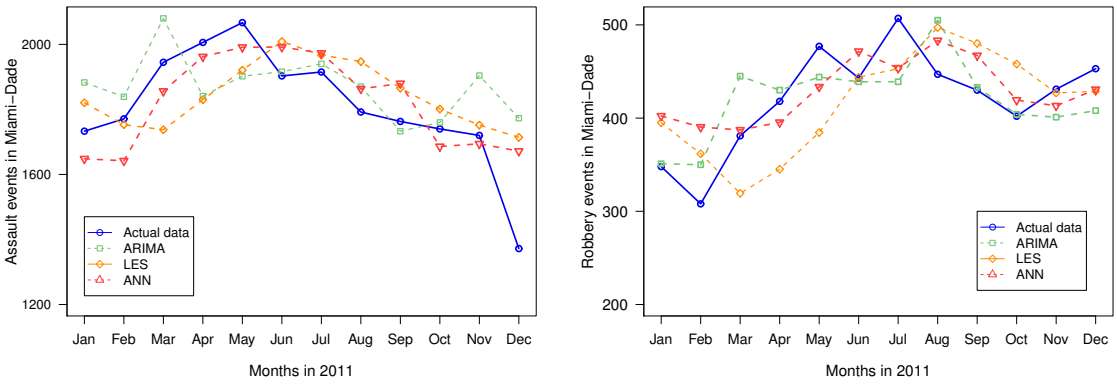
Figure 4.13: iSafe browser plugin overhead: Collecting reviews from venues, as a function of the number of reviews.

iSafe has a separate background service that displays the status bar of the Android device, the safety color label of the user's current location. Figures 4.12a and 4.12b show snapshots of iSafe's functionality.

4.8 Experimental Evaluation

4.8.1 Browser Plugin Performance

Figure 4.13 shows the overhead of the iSafe plugin when collecting the reviews of a venue browsed by the user, as a function of the number of reviews the venue has. It includes the cost to request each review page, parse and process the data for transfer. The experiments were performed on the Dell laptop. It exhibits a sub-linear dependence on the number of reviews of the venue (under 1s for 10 reviews but under 30s for 4000 reviews), showing that Yelp's delay for successive requests decreases. While even for 500 reviews the overhead is less than 5s, we note that



(a) Prediction of assaults, 2011 monthly basis. (b) Prediction of robberies, 2011 monthly basis.

Figure 4.14: Crime Forecasting Experiments in Miami-Dade

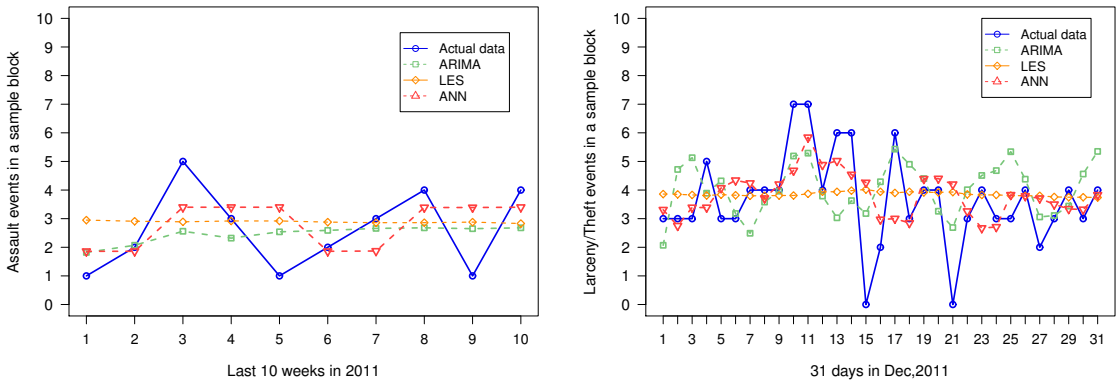
this cost is incurred only once per venue. Subsequent accesses to the same venue, by any other user will no longer incur this overhead.

4.8.2 Forecasting Accuracy

We explore here the performance of the time series forecasting techniques discussed in Section 4.2.3 in predicting the number of crimes to occur at a location during the near future, based on the recorded history.

We used the R statistical software package [R D11] to generate the ARIMA model and MATLAB toolboxes [MAT10] for LES and ANN models. In the following, we analyze separately three crime types, aggravated assault, robbery and larceny/theft that make up for more than 75% of the total amount of crimes. As we show later in this section, predicting categorized event counts enables the prediction of future safety values.

In the first experiment we used crime data recorded between 2007 and 2010 to predict per-month categorized event counts for the year 2011, for the entire Miami-Dade county.



(a) Prediction of assaults in a given block for the last 10 weeks of 2011.

(b) Prediction of larcenies in a given block for the last 31 days of 2011.

Figure 4.15: Crime Forecasting Experiments in Miami-Dade

Figure 4.14a compares the predictions for the number of assaults made by ARIMA, LES and ANN against the recorded values. Table 4.2 shows the RMSE and MAPE values for the three methods. All three models correctly predict the downward trend from May until December, with ANN achieving a slightly better accuracy than LES and ARIMA.

Figure 4.14b compares the predictions for the number of robberies made by ARIMA, LES and ANN against the recorded values. All models accurately predict the initial increase followed by a slight decrease in the number of robberies. ARIMA and ANN outperform the LES model, as confirmed by the RSME and MAPE values (see Table 4.2). ARIMA slightly outperforms ANN.

We further focus on finer grained spatial and temporal predictions: per-block, weekly events. For ANN, we partition the input data into 95 training vectors and 10 test vectors. Figure 4.15a compares the recorded data against the ARIMA, LES and ANN predictions of assault events in the last ten weeks of 2011, for one block in the Miami-Dade county. We emphasize the accuracy of the prediction (see Table 4.2), which is similar for ANN and ARIMA. Finally, we focus on daily

Model	Figure 4.14a		Figure 4.14b		Figure 4.15a		Figure 4.15b	
	RMSE	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
ARIMA	158.80	6.42	38.77	7.08	1.27	43	1.57	34.52
LES	151.03	6.79	53.57	11.89	1.41	42.08	1.61	30.07
ANN	116.48	5.32	40.44	8.23	1.3	35.72	1.49	27.02

Table 4.2: Error measurement data for ARIMA, LES and ANN.

crime predictions. For the same block used in the previous experiment, using a time window of events recorded between Jan 1, 2010 and Nov 30, 2011, we predict the 31 days of December 2011. Figure 4.15b shows the comparison between the recorded data and the ARIMA, LES and ANN forecast, for the daily number of larceny/theft events.

Experiment conclusions. ANN slightly outperforms ARIMA and LES, but all models exhibit good accuracy - except for the unexpected zero crime incidents observed during a couple of days. Intuitively, using predicted, future values for the number of crimes to define the safety of a block leads to more accurate values than using a static approach.

4.8.3 Yelp Safety Profiles

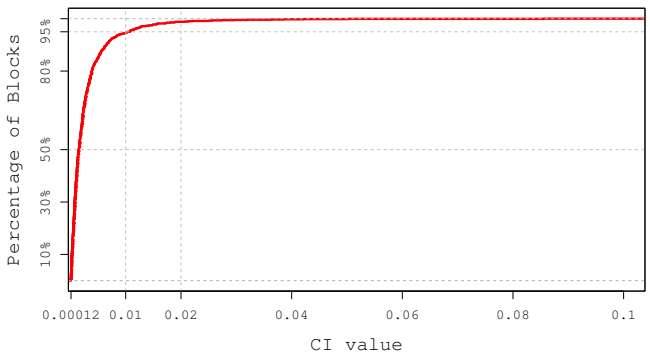


Figure 4.16: Distribution of block crime index values in the Miami-Dade county.

We have collected public information from the accounts of 2025 Yelp users, all residents of the Miami-Dade county. The information collected for each user includes the number of reviews, the venues reviewed, existing check-ins at any venues, and the date when each review and check-in was recorded. We build the crime index, CI, value for each Census block from the Miami-Dade county in 2010. Figure 4.16 shows the cumulative distribution function of the CI values (Figure 4.6 shows their spatial distribution). It shows that for the Miami-Dade county, most blocks experience relatively low levels of crime per-capita: 50% of blocks have a CI value smaller than 0.0015 and only 5% of blocks have CI values exceeding 0.01.

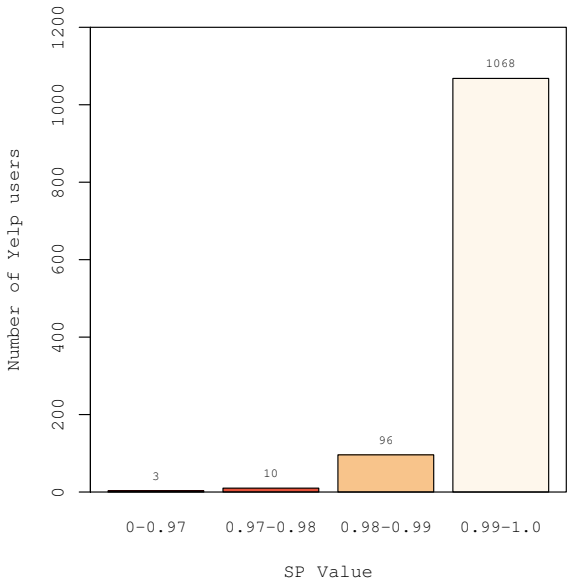


Figure 4.17: Distribution of safety index values of Yelp users.

Given the CI values of the blocks containing the venues visited (reviewed or subject of a check-in) by a yelper (Yelp user), we compute the user's crime index value, as defined by Equation 4.4, then the user's safety index: $SI_U = 1 - CI_U$. Out of the 2025 collected yelpers, 1194 had written reviews in 2010. Figure 4.17 shows the distribution of the safety index values of these 1194 yelpers. It shows that

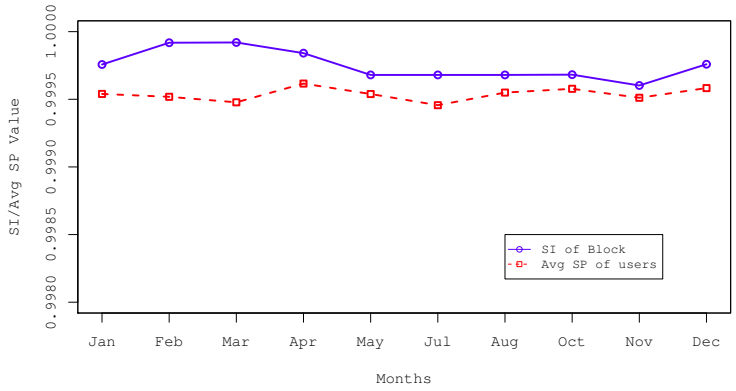


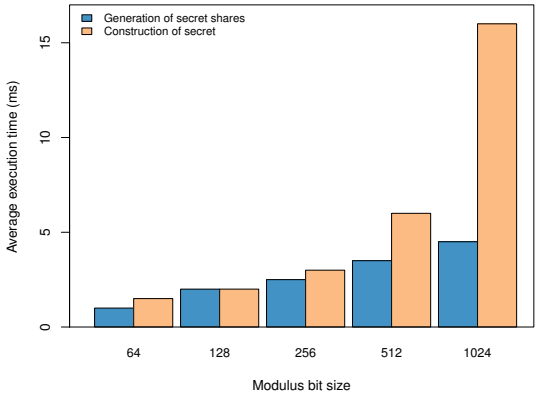
Figure 4.18: SI value of a Miami-Dade block and the average of SP values of Yelp users that visited the block w.r.t time.

most Miami-Dade county yelpers are safe: all have a safety index value larger than 0.96 (1 is the maximum value), with 90% of them exceeding 0.99.

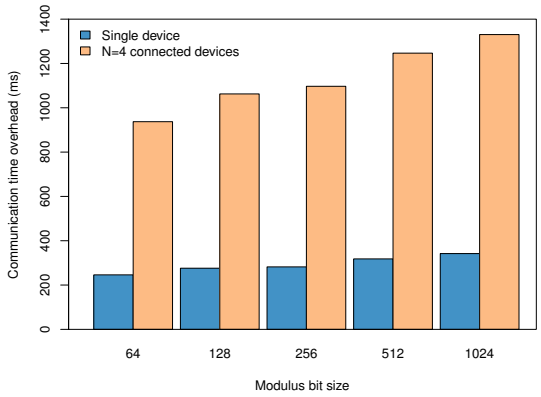
We further compare the evolution in time of the safety index SI_B of a block B with the average safety index values over the Yelp users that visited B (and left feedback). To this end, based on the crime database, for each month we calculate the SI value of each block in the Miami-Dade county. We then compute the monthly average of safety index values of yelpers that reviewed venues within B (during the month). Figure 4.18 shows the monthly evolution of the SI_B value of a Miami-Dade block and the average safety index value of the Yelp users that visited the block during 2010. For this block, the two metrics have similar values. This shows that an average of the safety indexes of the block's visitors can be used to replace a crime-based safety index for the block.

4.8.4 Android iSafe Evaluation

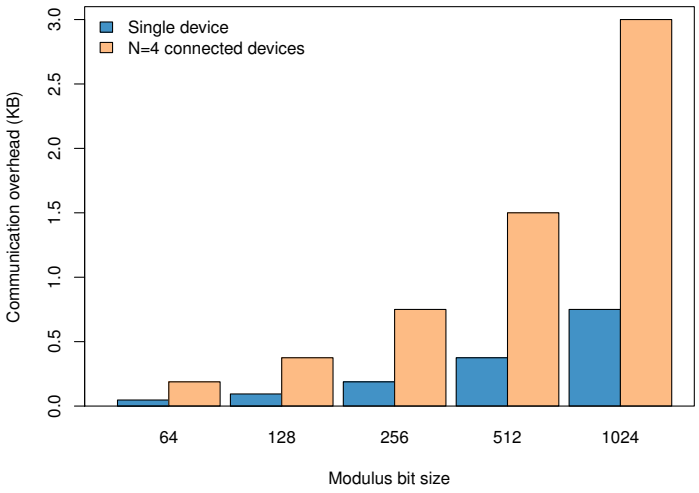
We have created a testbed consisting of 4 Android smartphones: Samsung Admire (OS: Gingerbread 2.3.4), HTC Aria (OS: Eclair 2.1), Sony E10i (OS: Eclair 2.1) and



(a) Secret share generation and secret reconstruction time overhead.



(b) iSafe communication overhead for single device and for all 4 devices.



(c) iSafe total communication size for single device and for 4 connected devices.

Figure 4.19: Android iSafe overhead.

Samsung GALAXY S II (OS: Gingerbread 2.3.4). We used Shamir's secret sharing solution. For single device testing, we used the Samsung Admire smartphone with a 800MHz CPU. In the following, all reported values are averages taken over at least 10 independent protocol runs.

We have first measured the overhead of the secret share generation and reconstruction operation. Figure 4.19a shows the overhead on the smartphone, when the modulus size ranges from 64 to 1024 bits. Note that even a resource constrained smartphone takes only 4.5 ms and 16 ms for secret splitting and reconstruction even for 1024 bit long moduli.

Furthermore, we focus on the time and space communication overhead for a single device as well as for the 4 connected devices in our testbed. Figure 4.19b shows the dependence of the communication time on the modulus bit size. Even for modulus size of 1024 bits, the average end-to-end communication overhead of a single device is 342ms and 1.3s of our whole system. Figure 4.19c shows the dependency of the communication overhead (in KB) on the modulus size ranging from 64 to 1024 bits, for a single device and for the whole system of 4 connected devices. Even for 1024 bit moduli, the total communication overhead is around 3KB.

CHAPTER 5

FILTERING FAKE INFORMATION IN LOCATION BASED SERVICES

In this chapter, we focus on detecting fraudulent information in review centered GSNs, such as Yelp and its ability to provide correct data. Specifically, we focus on malicious campaigns that aim to bias the public image of represented businesses in LBSs through the use of fake reviews. This chapter presents SpiDeR, an algorithm that takes advantage of the richness of information available in Yelp to detect review campaigns as venues exhibiting abnormal review patterns. We also leverage geolocation data and SpsJoin to detect fake venues that allow us to define a much richer ground truth dataset and to evaluate our results.

Acknowledgements. I would like to acknowledge Mr. Mahmudur Rahman. He contributed to specific parts shown in this chapter and gave valuable insights. In Sections 5.2.3 and 5.2.4, he helped in collecting Yelp Event and ground truth data respectively. In Section 5.4.1, he evaluated the Hampel identifier and wrote source code for the browser plugin in the WatchYT application (Section 5.5). He currently maintains the website of the WatchYT project.

5.1 Introduction

Geosocial networks (GSNs) such as Yelp [Yel] and Foursquare [fou] extend review-centered sites (e.g., Amazon [Ama], TripAdvisor [Tri]) with social and geographic dimensions. Subscribers own accounts where they store public profiles, use them to befriend and maintain contact with other users and provide feedback, in the form of reviews, for visited venues.

Since the impact of the occasional fraudulent review is likely to be minimized by many honest reviews, we focus here on *review campaigns*: entities that hire groups of people to write fake reviews and bias public opinion.

Previous work on TripAdvisor [YG09, OCCH11], focuses on identifying patterns in the text of fake reviews. Instead, our main goal is to increase the difficulty and thus the financial cost required to launch successful review campaigns. To achieve our goal we rely on Yelp's unique combination of geolocation and social components.

This chapter investigates techniques for detecting review campaigns. We collected publicly available data from Yelp using our own data crawler mechanism. Our experiments prove that detection techniques currently employed by Yelp can be bypassed, as we were able to engineer tens of reviews that were not filtered.

We first exploit relations between a user's location and those of the venues she and her friends review, to define a user rating. We introduce SpiDeR , an algorithm that detects review campaigns by identifying spikes generated by low rated reviewers.

We note that our contributions complement and are further motivated by the work of Byers et al. [BMZ12]. Byers et al. [BMZ12] argue that Yelp reviews mentioning Groupon may be low, due to other reviews being artificially high from actions taken by businesses. Our goal is precisely to identify such artificial reviews.

When tested on more than 16,000 venues with over a 1 million reviews we collected from Yelp, SpiDeR shows that spikes generated by low rated reviewers are frequent: we have identified hundreds of venues likely to have been the target of review campaigns. Furthermore, We explore a special form of campaign: Yelp Events. We collected data from different Yelp events and analyze the short and long term impact they have in the rating of venues. Our experiments show that there is a statistically significant dependency between number of reviews and the long and short term impact.

We tested SpiDeR on ground truth data extracted both from our review campaign experiments and from 27,622 reviews filtered by Yelp from 2,718 venues. This enabled us to not only experimentally set SpiDeR parameters but also to establish the novelty of our approach: Yelp does not detect review spikes generated by low rated reviewers.

Finally, when tested on data collected from more than 10,000 Yelp users, we were able to detect more than 150 users (1.5%) that took part in at least two campaigns and even several users who participated in more than 10 campaigns. This shows that while most Yelp users are honest, review campaigns are not isolated incidents.

5.2 System Model

Yelp [Yel] hosts the system, consisting of (i) information about venues, representing businesses or events with an associated location, e.g., restaurants, shops, offices, concerts, etc, and (ii) user accounts. Users can register and receive initial service credentials, including a unique user id. Yelp also supports queries from users, registering more than 70 million unique visitors per month [Wik12].

Reviews provided by users for venues have a numerical component, a *rating* ranging from 1 to 5, with 5 being the highest mark. Yelp associates an *average rating* value for each venue, computed over all the ratings of reviews left by users. Users can further leave pre-defined feedback for other reviews, by clicking on "useful", "funny" or "cool" buttons.

Yelp rewards "influential" reviewers with a special, yearly "Elite" badge. The reviews of Elite yelpers are often given priority when shown on a venue's page. Yelp also organizes "events" for Elite yelpers. Such events are hosted by a venue (chosen by Yelp) and a Yelp page is generated for this event.

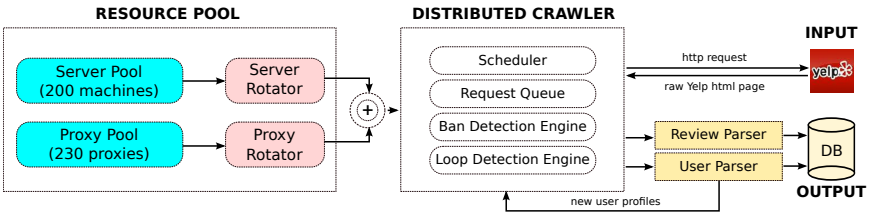


Figure 5.1: Crawler architecture.

5.2.1 Yelp Data

Data Collection

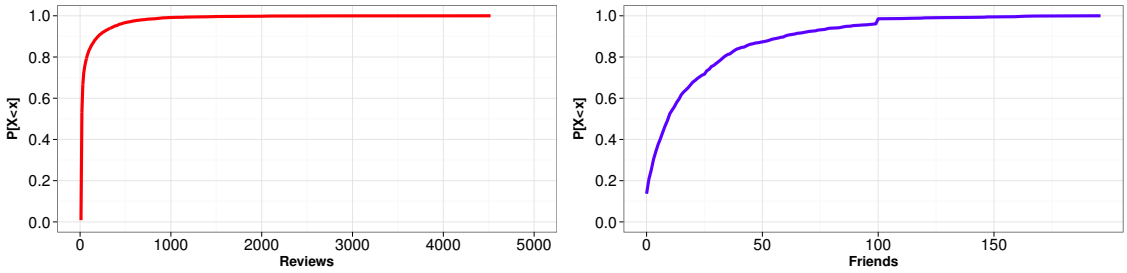
The crawler. We have developed a crawling engine to automatically collect data from Yelp's user and venue pages. The crawler uses a *resource pool* (see Figure 5.1) consisting of a set of servers and a set of proxies. For every request, the crawler randomly picks a server from the server pool and pairs it with a proxy from the proxy pool. The request is then made from the server, through the proxy. For each successful request, the crawler fetches the raw HTML page from Yelp and parses the required information. If the request is not successful, a new request is made using a different proxy. A centralized scheduler maintains a request queue to ensure there are no loops in the crawling process, i.e., avoids crawling the same page multiple times if referenced from several sources. When Yelp picks an anomalous proxy, any request made from this IP will return a blank HTML page or a page with error. Our crawler automatically detects this and changes the proxy. Furthermore, to minimize the load on Yelp's servers, and avoid detection, we introduce long inter-request intervals.

Crawling Yelp. In order to collect a representative sample of Yelp data, we used stratified sampling [TD00]. First, we selected a list of 10 major cities in the U.S. and we collected an initial random list of 100 venues from each of these cities as a seed dataset. It is important to understand that our strata (cities) are mutually

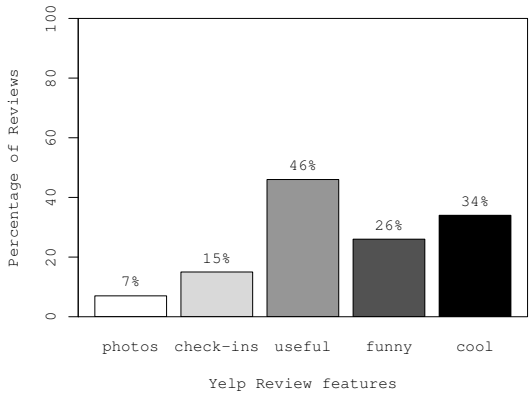
exclusive, i.e. venues do not belong to two or more different cities. This way we avoid bias towards high degree nodes, which is a common problem when crawling social networks [GKBM10]. We then randomly selected 10,031 Yelp users who reviewed these venues, and collected their data, including their id, location, number of friends and all their reviews, for a total of 646,017 reviews.

Given the list of 10,031 collected Yelp users, we merged the lists of the venues reviewed by those users (to avoid duplicate venues) and we randomly selected 16,199 venues, including venues from cities outside the U.S. (e.g., London, U.K, Vancouver, CA, etc). For each venue we have collected its name, location and type, along with all the reviews received, for a total of 1,096,044 reviews. For each review we extracted the reviewer id, the date the review was written, the number of check-ins performed and the photos uploaded by the reviewer at the venue, as well as feedback received by the review itself (number of users who thought the review was "useful", "funny" or "cool").

Figure 5.2a shows the cumulative distribution function (CDF) of the number of reviews per user. While only 20% of users have more than 100 reviews, the record user has 4,000 reviews. Figure 5.2b shows the CDF of the number of friends per user. Only 15% of users have no friends but 50% of users have more than 10 friends. Furthermore, Figure 5.2c shows the percentage of reviews that have associated photos, check-ins and user feedback. While 15% of reviews have an associated check-in, a respectable 46% of reviews have been labeled as "useful". This shows that Yelp is an active social network, whose users widely embrace its rich features.



(a) Distribution of the number of reviews (b) Distribution of the number of friends.



(c) Percentage of reviews with feedback.

Figure 5.2: Yelp user stats.

5.2.2 Yelp Events

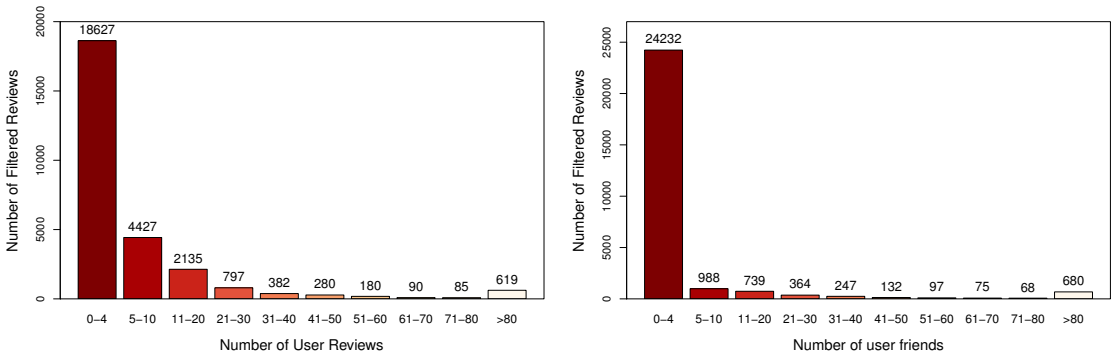
Yelp rewards users that write popular reviews with a special, *Elite* badge status. The Elite badge is awarded to users who not only write many reviews and have many friends, but whose reviews receive significant recognition (e.g., feedback) from other users. The reviews of Elite yelpers are never filtered and are often shown at the beginning of a venue's Yelp page.

Yelp organizes special *Elite events*, at select venues, where only Elite badge holders are invited. For each event, Yelp creates a separate Yelp page, containing the name of the event and the name, address and information for the hosting venue. Attendees are encouraged to review the event account, which then lists the reviews, just like a regular venue.

5.2.3 Yelp Event Collection

We have collected Yelp events from 60 major cities covering 44 states of USA. The remaining states had no significant Yelp events or activities (WY, VT, SD, NE, WV, ND). After identifying an Elite event, we identified the hosting venue through either its name or address. We used the crawler previously described to collect a majority of the available Yelp events and hosting venues, for a total of 149 pairs.

For each Yelp event and corresponding venue, we have collected their name, number of reviews, star rating and all their reviews. For each review, we have collected the date when it was written, the rating given and the available information about the reviewer, including the Elite status, number of friends and number of reviews written. In total, we have collected 24,054 event/hosting venue reviews.



(a) Distribution of the number of reviews written by the writers of the filtered reviews

(b) Distribution of the number of friends of the writers of the filtered reviews.

Figure 5.3: Yelp filtered reviews stats.

5.2.4 Ground Truth Data Collection

As part of our procedure to build a ground truth to validate our results, we manually collected fake data. Our collection process started by identify fake venues that were flagged by Yelp users in the Yelp Talk service. We also picked yelpers that had fake photos (collected from Google Images) or things that looked suspicious. While most of the reviewers have no friends and a majority only wrote one review, we were able to collect several users with more than 100 reviews and 20 friends.

Yelp filtered reviews. We have collected the reviews filtered by Yelp from 2,718 of our venue dataset, for a total of 27,622 filtered reviews. Since Yelp uses captchas to protect the access of filtered reviews, we created a tool that queries Yelp in the background, extracts the captcha and displays it to a captcha solver (a human) through a GUI. The captcha solver attempts to solve the challenge and submits the answer. If successful, the software leverages the current session to download all reviews that were unlocked by the CAPTCHA solver. For each filtered review, we have collected information about the review writer, such as number of friends and number of reviews, the location of the writer and if the user is currently an elite member of Yelp. Figure 5.3a shows the distribution of the number of reviews

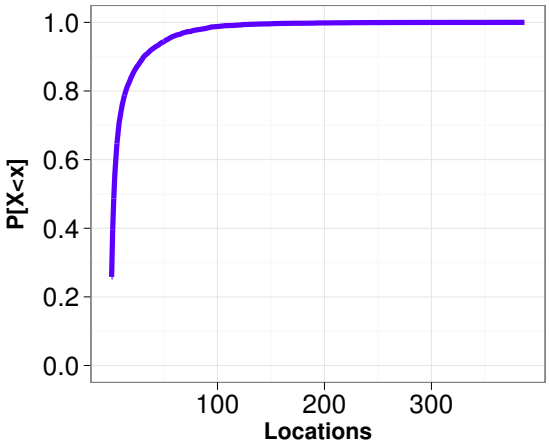
written by the writers of filtered reviews and Figure 5.3b shows the distribution of the number of friends of those users. They show that 87% of the writers of filtered reviews have at most 4 friends and 67% have written at most 4 reviews. We will use these findings to define the notion of user ratings (see Section 5.3).

5.3 User and Venue Analysis

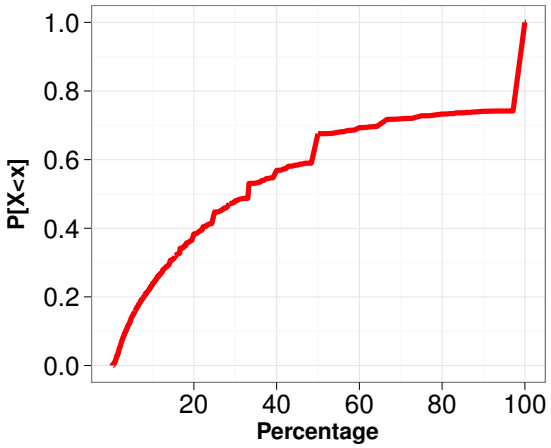
In this section we investigate spatial and temporal dimensions of information contained in user accounts and exploit our findings to introduce user ratings.

Spatial dimension. We focus first on the per-user clusters of locations of reviewed venues: locations where a user tends to write more reviews. Figure 5.4a shows the distribution of the number of cities where the collected users wrote reviews. 80% of users wrote reviews in less than 20 cities. However, several users exceed 300 cities, reaching as far as 378 cities. Then, should we expect a user to write most of her reviews in her home city? Figure 5.4b shows the CDF of the percentage of reviews written by the collected users in their most frequented city. It shows that as expected, 23% of users have almost all their reviews in the same city (the spike at the end of the curve) and 60% of users have more than a quarter of their reviews in the same city. Users that have a small percentage of their reviews in their most frequented city are likely to have reviewed venues in many cities.

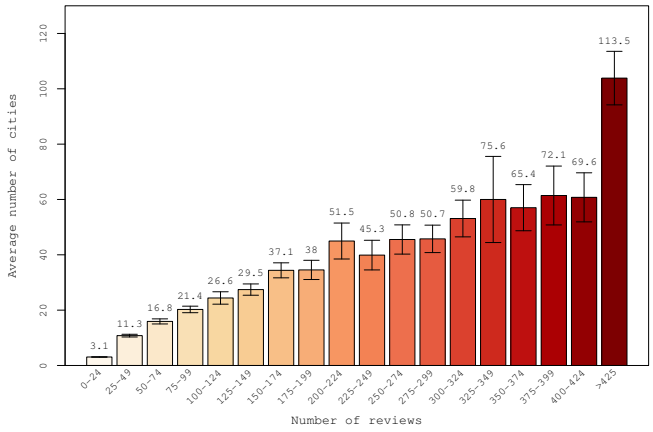
Then, Figure 5.4c shows the average (and corresponding 95% confidence intervals) number of cities where a user writes reviews, as a function of the number of reviews written by the user. Excluding the users with more than 425 reviews, the plot shows a sublinear increase in the number of cities. Venues reviewed by users with 400-425 reviews are in less than 70 cities, whereas those of users with more than 425 reviews are in more than 110 cities.



(a) Distribution of the number of cities per user, where a user wrote reviews.



(b) Distribution of the percentage of reviews written by users in their most frequented city.



(c) Per-user expected number of cities of venues reviewed given a range of written reviews.

Figure 5.4: Statistics of User Reviews

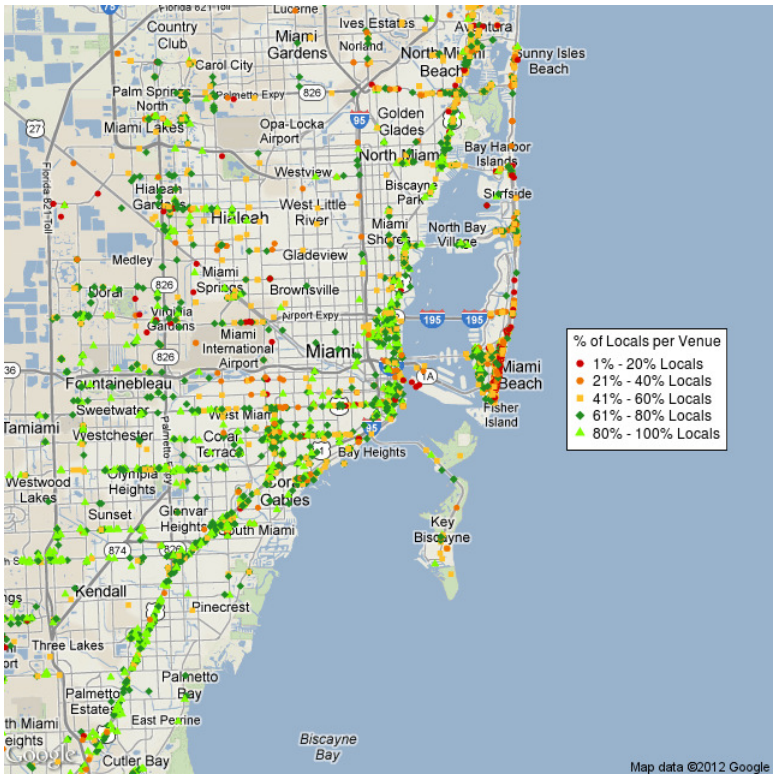


Figure 5.5: Geographic distribution of venues with more than 4 Yelp reviews, in Miami-Dade county, FL.

Our Yelp-Fakes experiment shows that no reviews were written by locals, i.e., users whose home city coincides with the reviewed venue's city. Then, in a second investigation, we have collected all the 7,699 venues registered in the Miami-Dade (FL) county; Figure 5.5 shows the geographic distribution of these venues, shown as small dots whose color represents the percentage of reviews from locals vs. the total number of reviews received by the venues. Red dots represent venues with many visitors (few locals) and green venues represent venues with few visitors. The plot only shows data from the subset of 4418 venues that registered more than 4 reviews. While venues around Miami Beach and the airports justifiably experience more visitors, we note that several venues in the western and southern parts of the county stand out - reds surrounded by greens. We exploit this observation in the definition of SpiDeR .

User timelines and ratings. We use the results of the above experiments to define the notion of user ratings. First, however, we introduce the helper notion of user timelines:

Definition 5.3.1 *(User Timeline) Let* $H_U = \{(V_i, R_i, \bar{R}_i, T_i) | i = 1..h\}$ *denote the timeline of a user* U*, a set of tuples, ordered by their timestamp* T_i*, where* V_i *is a venue reviewed by* U *and* R_i *is the rating assigned at time* T_i*, when the venue's average rating is* $\bar{R}_i$*. Furthermore, let* $AH_U = \{(V_i, R_i, \bar{R}_i, T_i | R_i \neq 0 \wedge \bar{R}_i \neq 0, i = 1..h_a\}$ *be the active timeline of user* U*, the subset of* H_U *containing only non-neutral reviews of* U *provided for non-neutral venues (whose average rating is not 3).*

We consider three rating types: positive ($R_i = 1$, for a star rating of 4 or 5), negative ($R_i = -1$, for a star rating of 1 or 2) and neutral ($R_i = 0$, 3 star rating reviews). $\bar{R}_i$, the average rating of V_i at time T_i, can also take one of three values: -1 if the average rating of venue V_i is below 3, 0 if equal to 3, and +1 if above

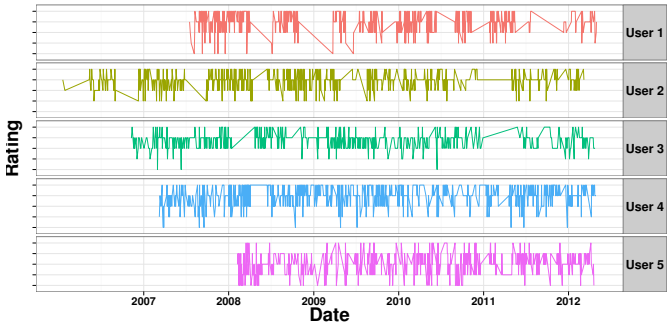


Figure 5.6: Visualization of the timelines of a sample set of users plotted against the review rating they assigned, since they became yelpers.

3. Figure 5.6 shows several timelines, of users sampled from the Yelp datasets. Diagonal lines denote periods of inactivity.

We rely on user timelines to introduce user ratings, a metric for differentiating dishonest from honest users. A high rating denotes an active user, with expertise in the areas reviewed and that has review active friends. We exploit the conclusion of Figure 5.4c to define the *expertise* Exp_V of a user U for a reviewed venue V, to be $Exp_V = c_V/h_a$, where c_V is the number of reviews written by U in the vicinity of V, not counting the review at V, and $h_a = |AH_U|$, is the number of active reviews of U (see Definition 5.3.1). We define the vicinity of a venue to be the circle centered at the location of the venue having a predefined radius (e.g., 50 miles in our experiments).

Let f_0 denote the number of friends of a user U that have at least T_r reviews. We define the *rating* of user U, combining U's spatial and temporal dimensions, to be:

$$R_U = \begin{cases} 0, & if\ (h < T_r \wedge f_0 < T_f) \\ \frac{\sum_{i=1}^{h_a} sgn(|R_i + \bar{R}_i|) Exp_i}{h_a}, & otherwise \end{cases} \tag{5.1}$$

where T_r and T_f are threshold variables whose values we discuss later, sgn is the sign function and Exp_i is U's expertise for the i-th reviewed venue. $sgn(|R_i + \bar{R}_i|)$

can only be 0 or 1. Thus, the rating of a user is defined to be 0 if the user has written less than T_r reviews and has less than T_f friends with at least T_r reviews each. This is suggested by the results of Figure 5.3a, Figure 5.3b.

Otherwise, the user's rating is a weighted average (over the length of its active history) of the user's concordance with her reviewed venues' average rating: if the venue and the user review both have either a positive or negative rating, the user's rating is incremented. The weight associated with a review is higher if the user has reviewed other close-by venues. We observe that $R_U \in [0, 1]$.

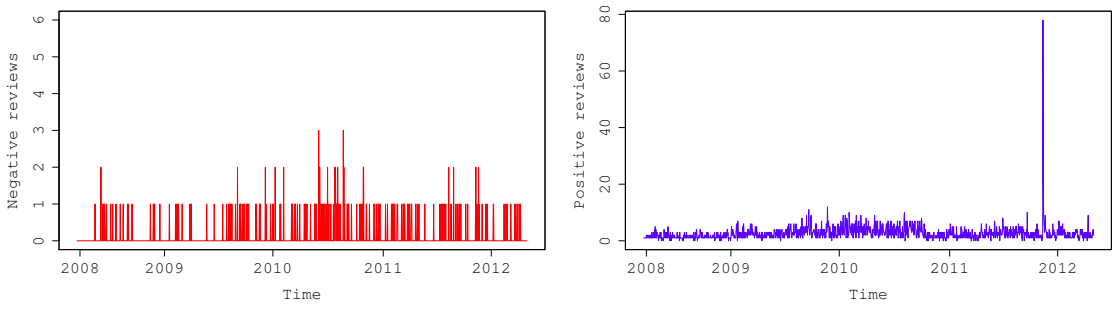
5.4 Detecting Review Campaigns

High user ratings seem to indicate honest users: well connected users, that write many clustered reviews, with ratings not going against the grain, are less likely to be malicious. However, by itself, this metric is not sufficient for detecting review campaigns. First, it suffers from a cold-start problem, as honest new users will likely have low ratings. Second, our Yelp-Fakes experiment shows that with additional effort, ratings can be engineered. Instead, in this section we propose efficient review campaign detection techniques that significantly raise the effort bar and thus the financial cost required to launch successful campaigns.

We first introduce the notion of venue timelines:

Definition 5.4.1 *(Venue Timeline) The timeline of a venue V is the set of tuples $H_V = \{(U_i, R_i, T_i)|i = 1..v\}$, the chronological succession of reviews R_i written for V by users U_i at time T_i.*

Figure 5.7 illustrates the venue timeline concept. Figure 5.7a shows the evolution in time of the number of negative reviews (1 and 2 star) and Figure 5.7b shows the



(a) Venue timeline with negative (1 & 2 star) reviews

(b) Venue timeline with positive (4 & 5 star) reviews

Figure 5.7: Venues timeline

evolution in time of the number of positive reviews (4 and 5 star) for a venue called "Ike's Place" in San Francisco, CA [IKE], whose first review was registered in 2008.

The number of daily negative reviews ranges between 0 and 3, and, with a total of 3,169 positive reviews in 1,220 active days, the number of daily positive reviews averages 2.59. However, on Nov. 7, 2011 (a Monday), the venue records a spike of 78 positive reviews. We propose then to detect such abnormal reviewing activities by analyzing venue timelines.

5.4.1 Review Spikes

We exploit the observation that in order for a review campaign to have an impact on the aggregate rating of a subject, it needs to contain a sufficient numbers of reviews. Such reviews, taken over an adequate time interval (days, weeks or months), will then stand out. We investigate the use of several techniques for retrieving ranges of abnormal reviewing activity, spikes or outliers in a venue's timeline.

Box-and-Whisker plots We first propose the use of measures of dispersion of Box-and-Whisker plots [TD00], consisting of, quartiles and interquartile ranges

(IQRs), to detect outliers. The idea is the following. Given a venue V, we first compute the quartiles and the IQR of the positive reviews from V's timeline H_V (negative reviews are handled similarly). We then compute the upper outer fence (UOF) value using the Box-Whiskers plot [TD00]. For each day d during V's active period, let P_d denote the set of positive reviews from H_V written during day d. If $|P_d| > UOF$, we output P_d, i.e., a spike has been detected. For instance, the aforementioned Ike's Place has a UOF of 9 for positive reviews: any day with more than 9 positive reviews is considered to be a spike. The advantages of this approach are that (i) it makes no distributional assumptions and it does not depend on a mean or standard deviation and (ii) its use of quartiles makes it less sensitive to extreme values.

Hampel Identifier The interquartile based approach however may not be adequate for a small sample size [IH93], such as venues that only have reviews during a small number of days. For such venues we propose an alternative approach. Given a venue V, we first compute the median and the MAD (Median Absolute Deviation) scale [Ham71] standard deviation S of the positive reviews for V (negative reviews are handled similarly). Then, we detect the outlier reviews using Hampel's identifier [HRRS86]: Set the threshold point TP for V as $t \times S$ where t is the dispersion value and typically $2 \leq t \leq 5$ (set to 4 in our experiments). For each day d during V's active period, let P_d denote the set of positive reviews from H_V recorded during d. If $|P_d| > TP$, output P_d.

5.4.2 SpiDeR

We now propose SpiDeR (Spike Detection Ranges), an algorithm that identifies review campaigns by combining detected review spikes with user ratings. Instead

Algorithm 3 SpiDeR : Fake review campaign detection.

```
1.campaign[]SpiDeR (V : venue, H_V : history)
2.  spikes[];      #list of spikes
3.  Rev_S[] : Review;   #reviews ∈ spike
4.  U_S[] : User;      #review writers
5.  campaign[];      #review campaigns
6.  T_s, w_r, T_p : float;   #threshold values

7.  spikes := detectSpikes(H_V);
8.  for s := 1 to spikes.size do
9.      Rev_S := getReviews(spikes[s]);
10.     if (Rev_S.size < T_s) then continue; fi
11.     U_S := getReviewers(Rev_S);
12.     counter := 0;
13.     for u := 1 to U_S.size do
14.         U := U_S[u];
15.         if (U.rating < w_r & U.city ≠ V.city) then
16.             counter := counter + 1;
17.     fi od
18.     if (counter/U_S.size > T_p) then
19.         campaign.add(spikes[s]); fi
20. od
21. return campaign[]
22.end
```

of flagging all review spikes as suspicious, SpiDeR carefully analyzes each review that is part of a spike: it counts the number of reviews in each spike written by users with low ratings and flags as suspicious only spikes made up by more than a threshold of low quality reviews.

Algorithm 3 shows SpiDeR 's steps, taking as arguments a venue and its timeline. First, it uses the abnormal behavior detection techniques of Section 5.4 to identify spikes in the number of either positive or negative reviews received by the venue (Algorithm 3 line 7). Each spike is processed separately (lines 8-20). For each spike, SpiDeR retrieves its component reviews and ignores the spike if the number of reviews is below a threshold T_S (lines 9-10). In our experiments we focus on

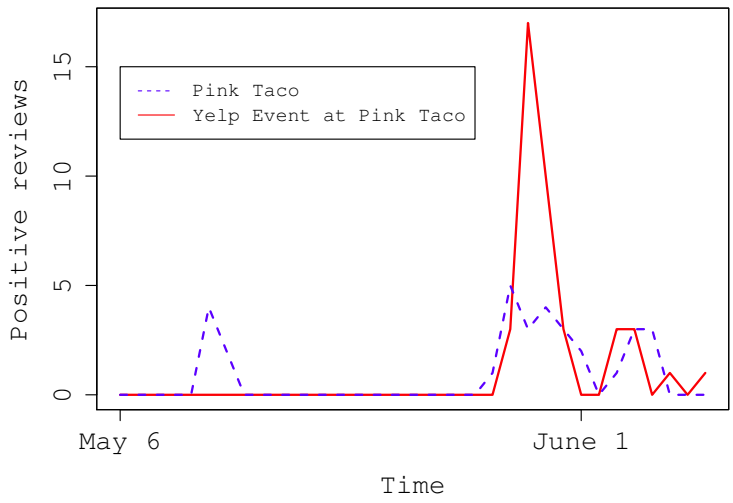


Figure 5.8: The timeline of "Pink Taco 2" (Los Angeles) and of the Yelp event for this venue. Note the correlation between the two.

day-long spikes and set T_S to twice the average number of daily reviews recorded by the venue.

Otherwise, for each reviewer who created one of those reviews (lines 11-14), SpiDeR marks her review as suspicious if (i) her rating (see Section 5.3) is below a threshold w_r and (ii) her city differs from the venue's city (lines 15-16). If the percentage of suspicious reviews in the spike exceeds another threshold T_p, SpiDeR marks the spike as a review campaign (lines 18-19). SpiDeR returns the list of identified campaigns (line 21). We evaluate SpiDeR in Section 5.5.3 and propose values for the parameters introduced here.

SpiDeR uses a range tree [Lue78] data structure to identify spikes in a time range. The range trees allows SpiDeR to aggregate spikes that are deemed to be suspicious in up to one week.

5.4.3 Yelp Events = Review Campaigns?

We introduce the hypothesis that Yelp events are a special type of review campaigns: Yelp selects the hosting venue, notifies the venue owners sufficiently in advance so they can ensure a good experience for their customers and allows only Elite yelpers to attend. Yelp creates a separate Yelp page for the event, where the attendees are encouraged to write the reviews of their experience of the event [Yel09]. While the declared goal of the event venues is to prevent unfairness to venues that do not host events, we study the impact of events on the venues hosting them. The question we ask is whether Elite events help improve the short and long term venue ratings. If such events have a positive effect, we believe they can be used as an alternative to fake reviews.

Our approach relies on the notion of positive venue *timelines*: the evolution in time of the number of daily positive reviews received by a venue. We use the venue timeline to identify abnormally high numbers of positive reviews received by the venue within a short time interval. This enables us to mark spikes that occur within a short timeframe of an event hosted by the venue, by using SpiDeR . We then compute the impact of the event on the venue, as the difference between the average rating of the venue at a given time following the event and its rating before the event.

Figure 5.8 shows a different type of correlation, for the venue "Pink Taco 2" located in Los Angeles. It displays the venue's timeline and the timeline of the Yelp page associated with the Yelp event. We emphasize that the venue's latest two spikes coincide with the spikes of the event. We study the effects of Yelp Elite events, organized for the benefit of Elite reviewers, on the image of the hosting venues. To this end, we introduce WatchYT, a tool for identifying venues receiving abnormally

Algorithm 4 WatchYT: Yelp campaign detection tool.

```
1.WatchYT(events[] : YelpEvent, ΔT : Time)
2.  campaings[];      #campaigns detected
3.  campaigns := newVenue[];
4.  for i := 0 to events.size() do
5.      YelpEvent e := events[i];
6.      Date eDate := e.getDate();
7.      Venue V := e.getVenue();
8.      Timeline H_V := V.getTimeline();
9.      TimeRange[] spikes := SpiKeR(H_V);
10.     if (spikes.correlated(eDate, ΔT)) then
11.         campaigns.add(V); fi
12. od
13 return campaings;
```

large numbers of reviews in a short time and use them to detect correlations between events and hosting venues. We evaluate our results in Section 5.5.2.

WatchYT: event/spike correlations. We introduce WatchYT (Watch Yelp Timelines), an algorithm that relies on SpiDeR to detect correlations between Yelp events and increased review activity concerning the venues hosting the events. Algorithm 4 shows the pseudocode of the approach. Specifically, given a set of Yelp events (*events*) and a time interval ΔT (system parameter), WatchYT determines the set of venues that benefit from an event within an interval ΔT of the event's date. WatchYT processes each Yelp event separately (lines 4-12). It first retrieves the date of the event, as representing the date when the first review was written for the event (line 6). It then retrieves the venue hosting the event (line 7), collects its reviews and reconstructs its timeline (line 8). WatchYT runs SpiDeR to detect abnormal review behavior over the timeline (line 9). If a spike occurs within an interval ΔT from the date of the event (line 10), it adds the venue to the list of detected campaigns (line 11).

5.5 Experimental Evaluation

WatchYT Implementation We have prototyped SpiDeR as part of a system we call WatchYT (Watch Yelp Timelines), that we made publicly available [WAT]. WatchYT consists of two components, a web server and a browser extension running in the user's browser. We use Apache Tomcat 6.0.35 to route requests (exposed to the client through a REST interface) to our server-side component. The server-side component relies on the latest servlet v3.0 which offers additional features including asynchronous support, making the server-side processing more efficient. We implemented the browser extension for the Chrome browser using HTML, CSS and Javascript. The plugin interacts with Yelp pages and the web server, using content scripts (Chrome specific components that let us access the browser's native API) and cross-origin XMLHttpRequests. If our content script receives content from another web site, it inspects it for cross-site scripting attacks before injecting the content into the current page (e.g., to protect the user from a hijack attack). To store and process review and user data for each venue, we use the SQLite 3.7.12.1 as the DB server.

The browser plugin becomes active when the user navigates to a Yelp page. For venue pages, the plugin parses their HTML file and retrieves their reviews. We employ a stateful approach, where the server's DB stores all reviews of venues previously accessed by users. This enables significant time savings, as the plugin needs to send to the web server only reviews written after the date of the last user's access to the venue's page. The initial access to a venue is likely to be slower, requiring the plugin to access multiple pages of reviews. Given the venue's timeline, the web server runs SpiDeR to identify any fake review campaigns. The server sends back details of any identified campaigns to the user's browser plugin, which includes

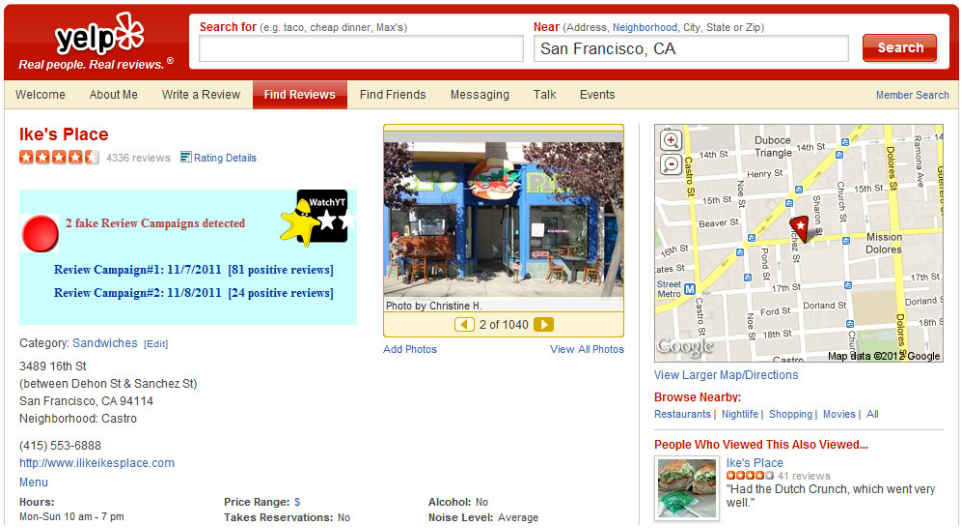
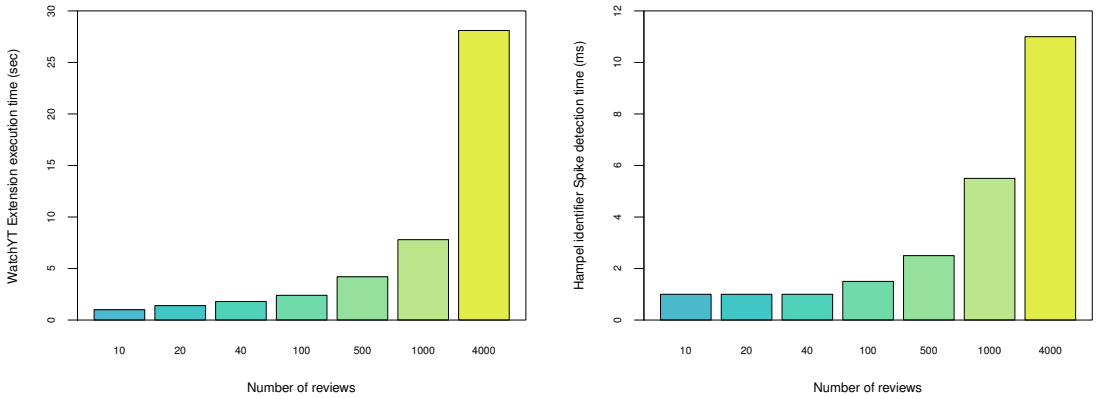


Figure 5.9: Snapshot of WatchYT 's plugin functionality for the venue "Ike's Place".

it in the browser. Figure 5.9 shows the output of WatchYT for the venue "Ike's Place". We note also that this approach may cause the user to be blocked from Yelp if several reviews are necessary to collect long history. This is specifically the case of venues that have lots of reviews.

We have evaluated the overhead of WatchYT in real deployments. Figure 5.10a shows the overhead of WatchYT when collecting the reviews of a venue browsed by the user, as a function of the number of reviews the venue has. It includes the cost to request each review page, parse and process the data for transfer. The experiments were performed on a Dell laptop equipped with a 2.4GHz Intel Core i5 processor and 4GB of RAM. It exhibits a sub-linear dependence on the number of reviews of the venue (under 1s for 10 reviews but under 30s for 4000 reviews). This is because Yelp's delay for successive requests on the same venue page decreases. While even for 500 reviews the overhead is less than 5s, we note that this cost is incurred only once per venue. Subsequent accesses to the same venue, by any other user will no longer incur this overhead.



(a) Collecting reviews from venues.

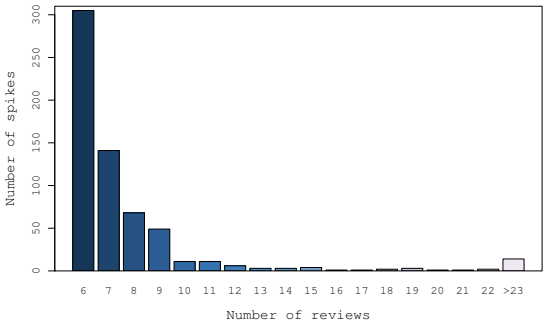
(b) Server-side SpiDeR processing of collected reviews.

Figure 5.10: WatchYT overheads

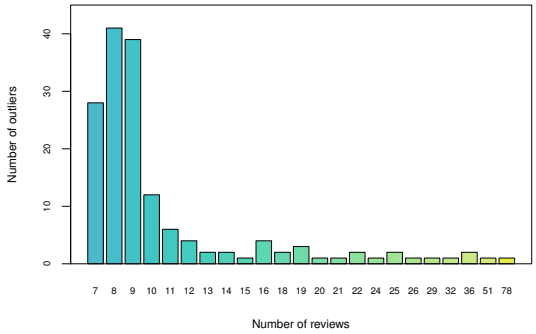
Furthermore, we have implemented the server-side SpiDeR using Java. Figure 5.10b shows the overhead of SpiDeR running on the same Dell laptop, as a function of the number of reviews. Even for 4000 reviews the overhead of detecting review spikes is less than 11ms.

5.5.1 Spike Detection Evaluation

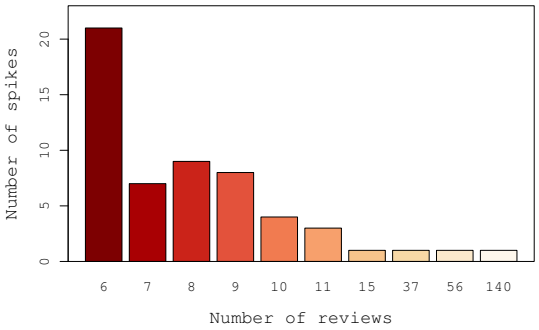
We now investigate the effectiveness of the review spike detection tools described in Section 5.4.1. Figure 5.11a shows the output of the Box-and-Whisker plot detection technique (see Section 5.4) when applied to the positive reviews of the 16,199 venues collected across the U.S.: the distribution of the amplitude (the number of reviews) of the spikes detected. It shows that the amplitude has a long-tail distribution. The aforementioned Ike's Place has the spike with the highest amplitude - 78 positive reviews in one day. Several spikes correspond to Yelp events. Figure 5.11b shows the output of the Hampel identifier technique when applied to the same venues. While the Hampel identifier detects significantly fewer small spikes than the Box-



(a) Dist. of review spikes. (Box-and-Whisker plots)



(b) Dist. of review spikes. (Hampel Identifier)



(c) Dist. of the number of reviews per negative spike. (Box-and-Whisker plot)

Figure 5.11: Performance of Outlier Detection Techiniques

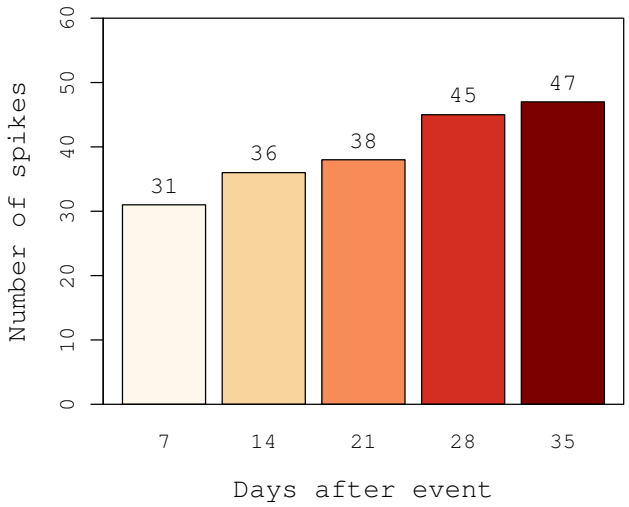


Figure 5.12: Yelp events: Spike count as a function of ΔT.

and-Whisker plot technique, it is able to detect a similar number of higher amplitude spikes (e.g., consisting of more than 20 reviews).

We have also used Box-and-Whisker plots to identify negative review spikes. Figure 5.11c shows the distribution of the number of reviews per negative spike. While we identified fewer negative spikes, they are more consistent than the positive ones, e.g., the "626 Asian Night Market" with 140 negative reviews.

5.5.2 An Analysis of Yelp Events

To validate our hypothesis that Yelp events are a special type of review campaigns, we have used the Box-and-Whisker approach to identify the impact of Yelp events on the hosting venue. Specifically, for each Yelp event we collected (see Section 5.2.1), we mark the corresponding hosting venue that has a spike occurring within a pre-defined interval ΔT after the date of the event.

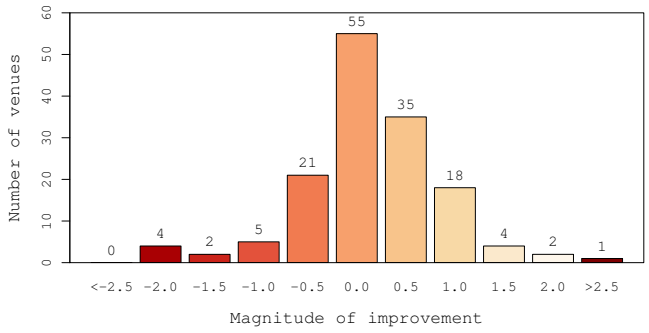


Figure 5.13: Yelp events: Distribution of the immediate impact of Yelp events on the venues' ratings.

Figure 5.12 shows the dependence between the spikes detected by the Box-and-Whiskers approach and the size of ΔT, ranging from 1 to 5 weeks. For instance, when ΔT is 14 days, we were able to detect 36 spikes on the 149 venues – in the 2 weeks following the event. Some venues had more than one spike within those 14 days. The total number of venues with at least one spike is 24, accounting for around 17% of the venues. Furthermore, Figure 5.13 shows the short term impact of a Yelp event on the hosting venue. We define the impact as the difference between the average rating of a venue 2 weeks after the event and its rating before the event. Almost twice as many venues benefit from Yelp events, when compared to those negatively impacted by it.

Furthermore, to understand the long term impact of Yelp events, we compared the current ratings of the 149 venues with their ratings before the events. Figure 5.17 shows the distribution (over the 149 venues) of the difference between the current rating and the rating before the events. While we see a balance between the number of venues showing an improvement versus a negative impact (16 positive vs. 14 negative) we note that the negative impact is only half a star. The positive impact reaches up to 3.5 stars!

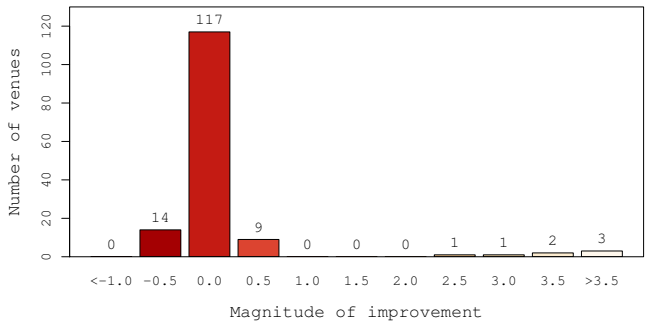


Figure 5.14: Yelp events. Distribution of the improvement due to events.

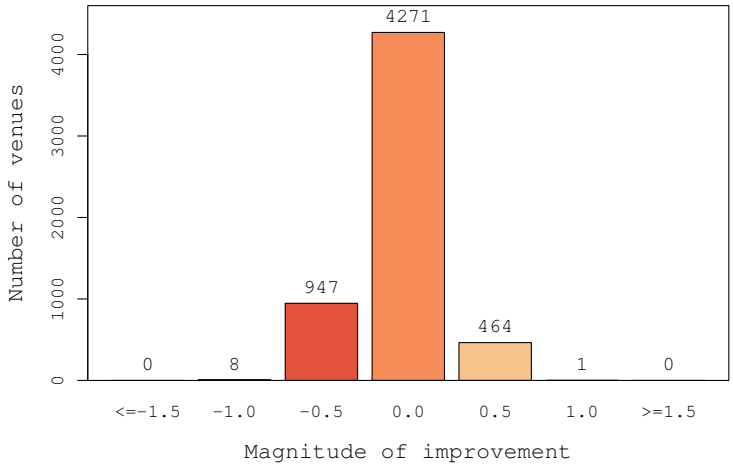


Figure 5.15: Yelp events. Distribution of the improvement with a random date.

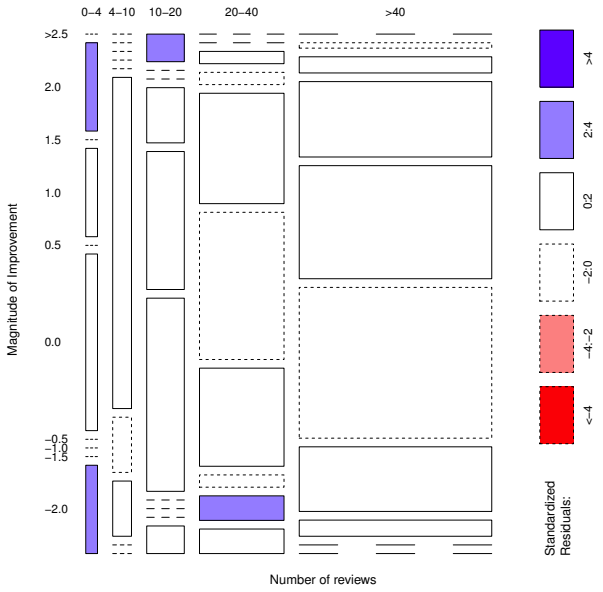


Figure 5.16: Dependency between the short term rating change of venues due to events and their number of reviews. Importance given by standardized residuals.

To validate these results, we have used the data of the 16,199 venues we have collected (see Section 5.2.1). We have filtered out venues exhibiting less than 30 reviews, leaving us with 5,691 venues. For each venue, we simulated an event as taking place at the median of the (149) Yelp events above. Then, we computed the average rating of the venue at the simulated event and subtracted it from the current rating of the venue. The distribution of the difference is shown in Figure 5.15. We note that twice as many venues show a negative impact vs. an improvement. Thus, these results suggest that Yelp events may pay off: venues that host events are more likely to experience a rating boost than a rating cut.

We then study the possibility of a relation between the number of reviews of a venue and the short term impact an event has on the venue. We observe that the impact of an event is quantified with fractions of rating, which means that we are dealing with a categorical variable. Therefore, we cannot use methods for linear or

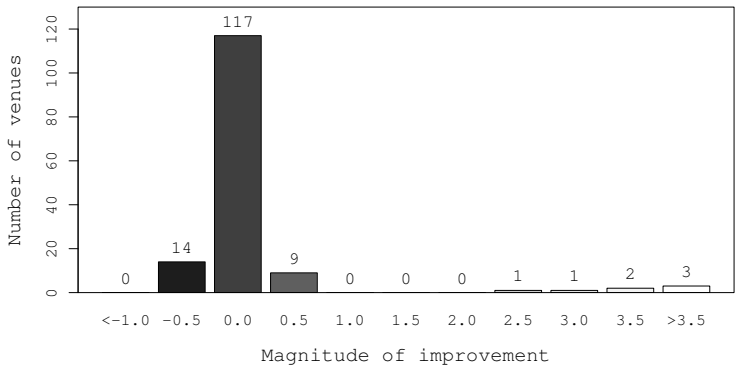


Figure 5.17: Yelp events: Distribution of the improvement due to events

non-linear association, e.g. correlation coefficient. Instead, we tested the hypothesis of independence, using a χ^2 test [TD00], between the rating impact and the number of reviews, a discrete variable. The test gave us a $\chi^2 = 58.6837$ with 36 degrees of freedom, which is highly significant with a p-value of 0.009854. Thus, we reject the hypothesis of independence.

Figure 5.16 shows the mosaic plot depicting this relation. Each rectangle corresponds to a set of venues, that have a certain review count range (the x axis) and having been impacted by a certain measure within two weeks of an event (the y axis). The shape and size of each rectangle depict the contribution of the corresponding variables, so a large rectangle means a large count in the contingency table. Blue rectangles indicate that they are more than two standard deviations above the expected counts. Then, the figure shows that more than half of the (149) venues have more than 40 reviews. Moreover, we notice that the venues having more than 40 reviews set the trend of Figure 5.13: while roughly one third of the venues show no impact, twice as many venues show a positive impact vs. a negative one.

We now focus on the long term impact of Yelp events. For this, we compare the current ratings of the 149 venues with their ratings before the events. Figure 5.17

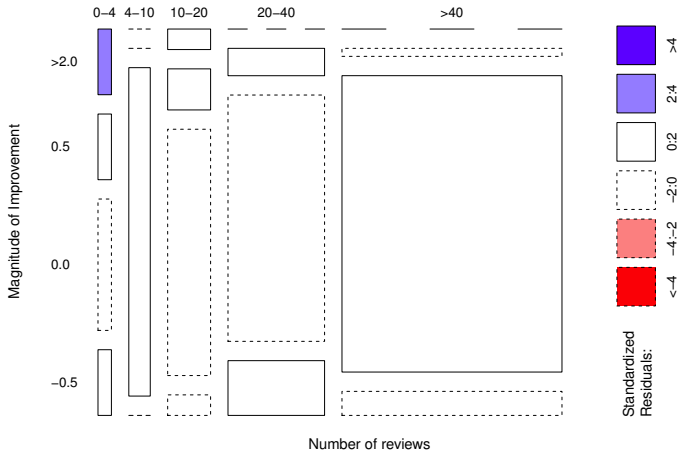


Figure 5.18: Dependency between the long term rating change of venues due to events and their number of reviews.

shows the distribution (over the 149 venues) of the difference between the current rating of the venues and their rating before the events. 78% of venues show no improvement. Furthermore, we see a balance between the number of venues showing an improvement versus a negative impact (16 positive vs. 14 negative). However, we emphasize that the negative impact is only half a star, while the positive impact reaches up to 3.5 stars.

We conduct a χ^2 test to verify the dependence of the long term impact of events on venues on the number of ratings of the venues. The test was highly significant with $\chi^2 = 29.2038$, 12 degrees of freedom and a p-value of 0.003674. Figure 5.18 shows the mosaic plot: a vast majority of the venues having more than 40 reviews have no impact on the long term. This shows that review spikes have a smaller impact on constantly popular venues.

Conclusions on Yelp Events. On the long term, events do not seem to impact the ratings of hosting venues. We believe this is because high numbers of regular

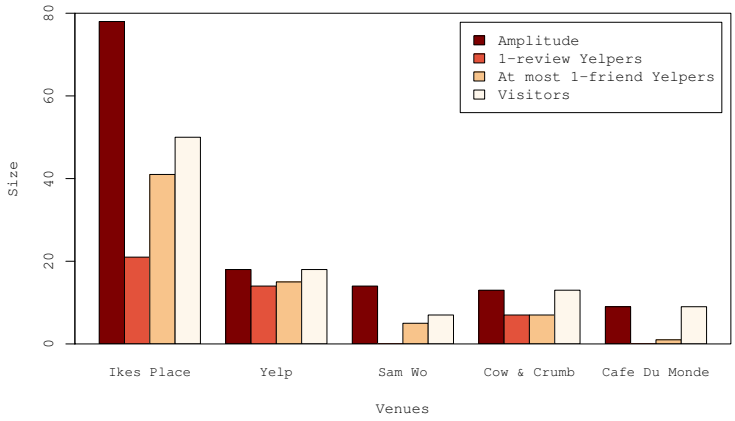


Figure 5.19: SpiDeR output. Zoom-in of Figure 5.11a.

reviews tend to overwhelm the impact of event spikes. However, Yelp events show a noticeable short term positive impact. Even a short term increase in popularity may act as a motivation to host such events [AM12].

5.5.3 SpiDeR Evaluation

We implemented SpiDeR in Java. Our implementation pre-loads a compressed version of the database of venues in memory, for efficient computation of spikes, avoiding the I/O overhead incurred by the database engine, e.g. SQLite in our case.

Experimental parameter setup. Figure 5.19 zooms in into several (non-Yelp event) spikes from Figure 5.11a, showing, for each venue, the spike's amplitude, the number of reviewers that have only one review, the number of reviewers that have at most one friend and the number of out-of-town reviewers. It identifies several other suspicious spikes and venues (including Yelp's own venue!), registering between 25-85% reviewers with 1 review, between 52-90% reviewers with at most one friends and between 64-100% out-of-town reviewers. Thus, SpiDeR detects the spikes of these venues even with very low parameter values: $T_r = 2$, $T_f = 2$, $w_r = 0$ and $T_p = 0.25$.

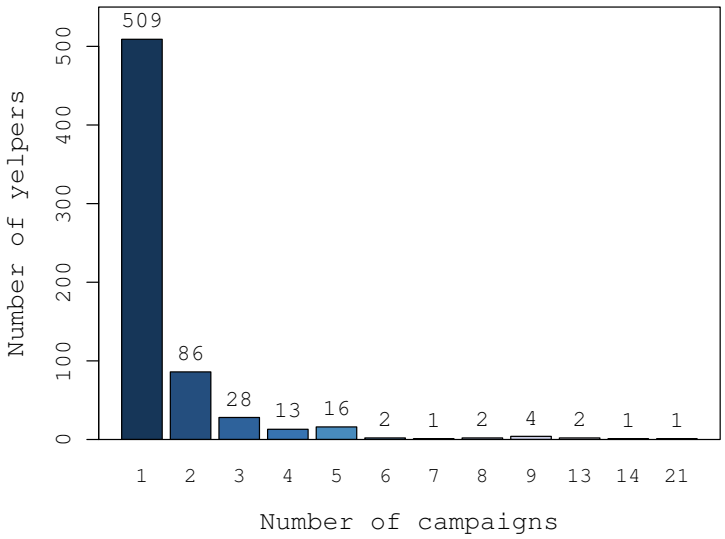


Figure 5.20: Distribution of the number of campaigns in which users participated, including Yelp events.

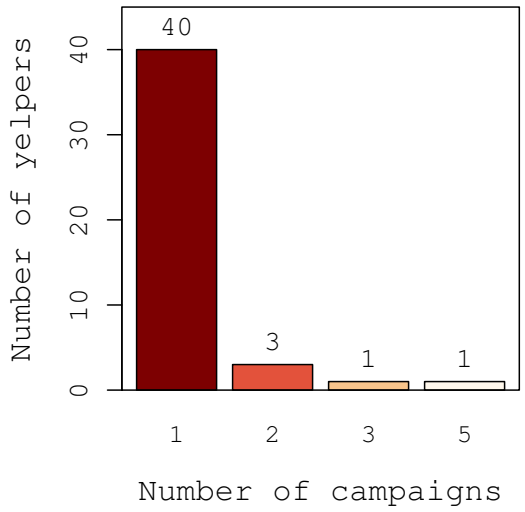


Figure 5.21: Distribution of the number of campaigns in which the fake users participated.

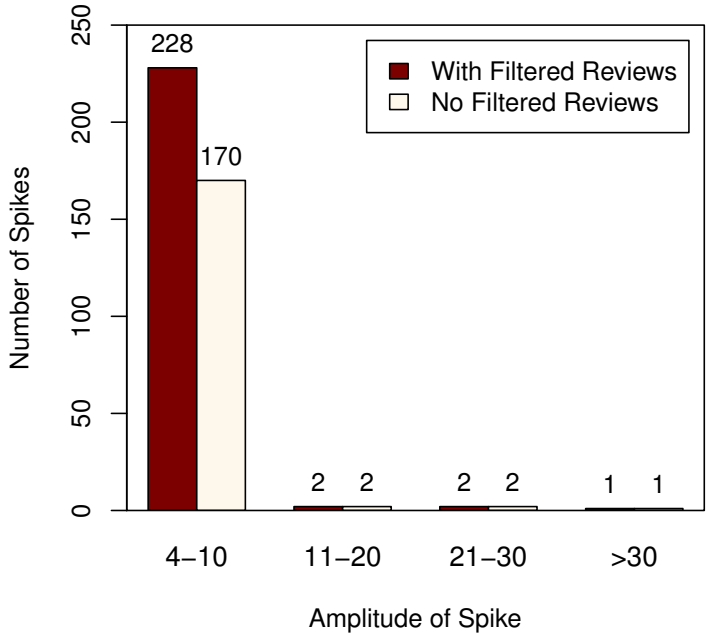


Figure 5.22: Distribution of the amplitude of spikes detected by SpiDeR when considering only Yelp non-filtered reviews and when considering also filtered reviews.

The ground truth. In a first experiment, we evaluate the ability of SpiDeR to detect users that have participated in multiple review campaigns, using the data we have collected from the fake reviewers identified and described in Section 5.2.4. Figure 5.21 shows our results: 45 users are identified as participants in fake campaigns. The reason for not identifying all the fake reviewers is that several users have written neutral (3-star) reviews, thus were not considered by SpiDeR .

In a second experiment, we compare SpiDeR and Yelp's filtering mechanisms. For this, we use the data collected from the 2,718 venues, from which we have also collected the filtered reviews. First, for each venue, we feed SpiDeR with all the reviews registered by the venue, including the filtered reviews. Second, for each venue we run SpiDeR on only the non-filtered reviews. In both runs, we set T_p = 0.5, T_r = 2 and T_f = 2. That is, SpiDeR only returns spikes that have at

least 50% reviews written from accounts having at most 2 friends and at most 2 written reviews. Figure 5.22 compares the distribution of the amplitude of the spikes detected in the two SpiDeR executions. It shows that the reviews filtered by Yelp belong to 58 spikes of amplitude ranging between 4-10 reviews. Thus, while Yelp filters reviews based on the number of friends and reviews of their writers, Yelp is *not* detecting review campaigns. By discovering clusters of time-clustered reviews written by low-rated reviewers, SpiDeR complements Yelp's defense mechanisms.

Repeat campaigners. Figure 5.20 shows the distribution of the number of review campaigns in which yelpers from our dataset have participated, when considering also Yelp events. While we expected a long-tail distribution, we note the high number of participants in review campaigns - over 650, with more than 150 users participating in 2 or more campaigns. We emphasize the user with 21 review campaigns, while 16 users have participated in 5 campaigns for different venues.

5.5.4 Conclusions and Limitations

We were able to detect several hundred review campaigns, that include both (i) Yelp events that impacted the rating of hosting venues and (ii) malicious behavior exhibited by other parties, e.g., venue owners or the competition.

Also, we were able to find dependency between Yelp features and long and short term impact on the venues. Although it is unlikely that for a fake campaign to impact in the long term, a venue may harness short term impact for quick revenue.

While WatchYT can be bypassed by motivated adversaries, the financial effort of a successful campaign is increased: campaigners with good reputations need to be recruited. A good reputation entails having at least 2 active friends and at least 2 written reviews.

CHAPTER 6

CONCLUSIONS

This chapter describes the most important conclusions of our work and future research directions.

6.1 Summary

In this dissertation, we have focused on three important problems that arise in location based services, including geosocial networks. In order to address the problem of ensuring data accuracy, we proposed an important geographic database operator to perform data preprocessing and knowledge discovery using spatio-textual constraints. Specifically, we defined the SpsJoin operation and implemented our solution in MapReduce. We have shown that our algorithm harnesses both, textual and geographic attributes to compute relevant pairs between two datasets. We defined metrics to score importance of geographic objects and we leveraged entropy definitions to improve the performance of our algorithms. Our ground truth definition allowed us to evaluate the precision of the join algorithm.

In order to address privacy concerns, we introduced an application to motivate the use of LBSs and to share sensitive information in a private manner. In particular, we envisioned a safety awareness LBS that leverage both spatial and nonspatial data. We have proposed several techniques for evaluating the safety of users based on their spatial and temporal dimensions in order to motivate participation and tackled privacy concerns. We implemented our concepts with an application in Safe Cities. We have shown that data collected by geosocial networks bears relations with crimes. We have proposed a holistic approach toward evaluating the safety of a user, that combines the predicted safety of the user's location with the aggregated safety

of the people co-located with the user. We proposed iSafe, a privacy preserving algorithm for computing safety of users at geographic locations. Our Android and browser plugin implementations showed that our approach is efficient both, in terms of the computation and the communication overheads.

Finally, to ensure correctness of the data, we explored malicious behavior in LBSs in the form of fake review campaigns. We collected large amounts of data and build an algorithm to identify malicious behavior in a review based LBS (e.g. Yelp). In particular, we have shown that review campaigns in geosocial networks are possible and inexpensive. We have developed algorithms that enabled us to detect hundreds of venues targeted by review campaigns as well as hundreds of participating transgressors, some in as many as 20 campaigns. Through our prototype evaluation, we prove that our Yelp extension efficiently alerts users when browsing suspicious venues, even when the venues have hundreds of reviews.

We also have studied the problem of review campaigns organized by Yelp and involving elite yelpers. We have shown that Yelp events are also effective review campaigns, collecting good reviews from influential users. We have proposed SpiDeR, an approach that identifies positive and negative review spikes in the timelines of venues. We have introduced WatchYT, an SpiDeR extension and a browser plugin that finds venues that have spikes correlated with events they organized. We have used venue and event data collected from Yelp to investigate the impact of Yelp events. We have shown that while a short term positive effect can be seen, in the long run, the effects of events are normalized by the reviews of regular users.

6.1.1 Future Directions

This research can be extended in the following directions.

1. From the geospatial databases perspective in LBSs, we believe that Spsjoin has a promising future. One possible direction is to include semantics similarity in our content similarity function. Our current work does not handle semantics and this will be a great addition to the overall approach. The use of ontologies and *Latent Semantic Indexing* (LSI) is definitively a good research direction,but there is caveat: the performance of the algorithm can be affected as semantic computations demand large amounts of data lookups.

2. Another interesting direction is the *Fuzzy Spatio-Semantic* query (FSS) for geocoding. The FSS scores objects based on the current location. When the geocoder transforms a string into coordinates, it will first retrieve the current position and finds the context in which the query was done. FSS will return results based on the importance in the current position. For instance, if someone is in Miami, OH, a query to "Miami" will likely to return objects from Miami, FL. The FSS identifies the current context and returns corresponding results, putting Miami, OH objects ranked first.

3. SpsJoin can be extended to handle multiway joins. Although our solution suggests how it can be done, further research is necessary to define efficient algorithms. A possible approach leverages parallel computation with Mapreduce. It requires clustering using all datasets envolved in the join. Further research is required to handle heterogeneous datasets and different types of schemas.

4. Safe cities is also another niche of future research. The possibility of incorporating safety information in LBSs opens up new ideas for interesting applications, such as finding the "safest route". The safest route tells the user of the LBS system the directions that she must follow to arrive to a certain location. Inspired in the shortest route, found in Google Maps and Terrafly, the

safest route receives an initial point or source and a destiny. Then, the system computes the aggregated safety information based on *safety footprints* that iSafe users leave in the system and computes the trajectory that maximizes the safety. This system may also involve safety information from venues and social networks.

5. Our mechanisms for detection of fake reviews in review based LBSs can be extended in several directions. We have observed that plagiarism is a common practice in fake review campaigns. One possible reason is that it is cheaper for the attacker to copy text from a well-written review in a venue and post it in another venue. Plagiarism can be detected by computing similarity self-joins on large datasets of reviews. We have found several users that seem to be engineered specifically to post fake reviews and to exercise the social network. If successful, those users may create a connected component, strong enough to bias the opinion in different venues and also to create long term impact, which is the ultimate goal of these reviews campaigns. Careful analysis, attacks and defenses can be a great topic for research in security.

6. Our work can be extended to handle adversary attacks that attempt to increase the rating of a given venue through a well known attack called ballot stuffing attack. Another interesting type of attack is the de-anonymization attack. A possible approach is to use geosocial networks data and use SpsJoin to de-anonymize a a geosocial network data set. In this case, the approach does not use the network as its major source of attack, but rather, the spatial and nonspatial attributes of the data.

7. It would be interesting to apply other machine learning algorithms for classification of fake reviews. However, the lack of ground truth is a major problem in supervised learning. Building such a ground truth is definitively an important

contribution in solving the problem of fake reviews. In our work, we developed our own ground truth to evaluate the performance of our solution but it has flaws that may be fixed by producing a more complete golden dataset that allows researchers to further experiment in this exciting topic.

BIBLIOGRAPHY

[AAGY01] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 96–105, New York, NY, USA, 2001. ACM.

[ABL10] Sattam Alsubaiee, Alexander Behm, and Chen Li. Supporting location-based approximate-keyword queries. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 61–70, New York, NY, USA, 2010. ACM.

[AGK] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. Efficient exact set-similarity joins. In *VLDB'06*.

[AI08] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008.

[AM12] Michael Anderson and Jeremy Magruder. Learning from the crowd: Regression discontinuity estimates of the effects of an online review database. *Economic Journal*, 122(563):957–989, 2012.

[Ama] Amazon. `www.amazon.com`.

[Apa12] Apache. Hadoop. `http:\\hadoop.apache.org`, 2012.

[APR+98] Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, and Jeffrey Scott Vitter. Scalable sweeping-based spatial join. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 570–581, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[ASE13] ASEANSTARTUP.com. Three-quarters of smartphone owners use location-based services. `http://aseanstartup.com/2013/04/23/foursquare-from-location-based-service-to-big-data-provider/`, 2013.

[AWS] Amazon web services. `http://aws.amazon.com/`.

[BCR11] Jaime Ballesteros, Ariel Cary, and Naphtali Rishe. Spsjoin: parallel spatial similarity joins. In *Proceedings of the 19th ACM SIGSPATIAL*

International Conference on Advances in Geographic Information Systems, GIS '11, pages 481–484, New York, NY, USA, 2011. ACM.

[BCR+13] Jaime Ballesteros, Bogdan Carbunar, Mahmudur Rahman, Naphtali Rishe, and S.S. Iyengar. Towards safe cities: A mobile and social networking approach. *Parallel and Distributed Systems, IEEE Transactions on*, 2013. Accepted.

[BCRR13] Jaime Ballesteros, Bogdan Carbunar, Mahmudur Rahman, and Naphtali Rishe. Yelp Events: Making Bricks Without Clay? In *Proceedings of the 5th International Workshop on Hot Topics in Peer-to-peer computing and Online Social Networking (HOTPOST). In conjunction with IEEE ICDCS 2013*, Philadelphia, PA, July 2013.

[BGM12] Panagiotis Bouros, Shen Ge, and Nikos Mamoulis. Spatio-textual similarity joins. *Proc. VLDB Endow.*, 6(1):1–12, November 2012.

[BMBR11] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: when bots socialize for fame and money. In *The 27th Annual Computer Security Applications Conference, (ACSAC)*, pages 93–102, 2011.

[BMS07] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. *Proceedings of the 16th International Conference on World Wide Web (WWW)*, May 2007.

[BMZ12] John W. Byers, Michael Mitzenmacher, and Georgios Zervas. The groupon effect on yelp ratings: a root cause analysis. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 248–265, New York, NY, USA, 2012. ACM.

[Bos11] Big Boss. Location spoofer. http://goo.gl/59HMk, 2011.

[BRC+13] Jaime Ballesteros, Mahmudur Rahman, Bogdan Carbunar, Rahul Potharaju, Radu Sion, Nitya Narasimhan, and Naphtali Rishe. POSTER: Towards Detecting Yelp Review Campaigns. In *Proceedings of the 34th IEEE Symposium on Security and Privacy*, San Francisco, CA, May 2013.

[BRCR12] J. Ballesteros, M. Rahman, B. Carbunar, and N. Rishe. Safe cities. a participatory sensing approach. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 626–634, 2012.

[BSBK09] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, 2009.

[BSV98] N. Barberis, A. Shleifer, and R Vishny. A model of investor sentiment. *Journal of Financial Economics*, 49:307–243, 1998.

[BYC07] Ricardo Baeza-Yates and Carlos Castillo. Crawling the infinite web. *J. Web Eng.*, 6(1):49–72, March 2007.

[CDBN09] A. Caragliu, C. Del Bo, and P. Nijkamp. Smart cities in europe. Serie Research Memoranda 0048, VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, 2009.

[Cen10] United States Census. 2010 census. http://2010.census.gov/2010census/, 2010.

[Coc11] Kira Cochrane. Why TripAdvisor is getting a bad review. The Guardian, `http://www.guardian.co.uk/travel/2011/jan/25/tripadvisor-duncan-bannatyne`, January 2011.

[Com79] Douglas Comer. Ubiquitous b-tree. *ACM Comput. Surv.*, 11(2):121–137, June 1979.

[CP12] Bogdan Carbunar and Rahul Potharaju. You unlocked the Mt. Everest Badge on Foursquare! Countering Location Fraud in GeoSocial Networks. In *To appear in Proceedings of the 9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, 2012.

[CR05] S. Chainey and J. Ratcliffe. *GIS and Crime Mapping*. Wiley, 2005.

[Cri] James Cridland. Mapping the riots. `http://james.cridland.net/blog/mapping-the-riots/`.

[CRKH11] D. Christin, A. Reinhardt, S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928 – 1946, 2011.

[CRS02] S Chainey, S Reid, and N Stuart. *When is a Hotspot a Hotspot? A Procedure for Creating Statistically Robust Hotspot Maps of Crime*. Kidner, D and Higgs, G and White, S, 2002.

[CSHR09] Ariel Cary, Zhengguo Sun, Vagelis Hristidis, and Naphtali Rishe. Experiences on processing spatial data with mapreduce. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, pages 302–319, Berlin, Heidelberg, 2009. Springer-Verlag.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[CTU08] Spencer Chaineya, Lisa Tompson, and Sebastian Uhlig. The utility of hotspot mapping for predicting spatial patterns of crime. *Security Journal*, 21:4 – 28, 2008.

[CWR] Ariel Cary, Ouri Wolfson, and Naphtali Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In *SSDBM'10*, pages 87–95.

[Dep] Miami-Dade Police Department. Crimeview community. http://crimemaps.miamidade.gov.

[DG] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 2008*, page 51(1).

[DG92] David DeWitt and Jim Gray. Parallel database systems: the future of high performance database systems. *Commun. ACM*, 35(6):85–98, June 1992.

[EA07] Volkan S. Ediger and Sertac Akar. Arima forecasting of primary energy demand by fuel in turkey. *Energy Policy*, 35:1701–1708, 2007.

[ECC+05] John E. Eck, Spencer Chainey, James G. Cameron, Michael Leitner, and Ronald E. Wilson. Mapping crime: Understanding hot spots. Special, U.S. Department of Justice, Office of Justice Program, National Institute of Justice, August 2005.

[Eng12] 2011 England riots. Wikipedia, at `http://en.wikipedia.org/wiki/2011_England_riots`, Last accessed on July 12, 2012.

[Est10] Deborah L. Estrin. Participatory sensing: applications and architecture. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010.

[FAdO+10] Vasco Furtado, Leonardo Ayres, Marcos de Oliveira, Eurico Vasconcelos, Carlos Caminha, Johnatas DOrleans, and Mairon Belchior. Collective intelligence in law enforcement the wikicrimes system. *Information Sciences*, 180(1):4 – 17, 2010.

[FIU] High Performance Database Research Center Florida International University. Terrafly. http://terrafly.com/.

[FLI] Flickr. `http://www.flickr.com/`.

[fou] Foursquare. `https://foursquare.com/`.

[Fra12] 2005 civil unrest in France. Wkipedia, at `http://en.wikipedia.org/wiki/2005_civil_unrest_in_France`, Last accessed on July 12, 2012.

[FXGC12] Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. Distributional footprints of deceptive product reviews. In *Proceedings of the Sixth International Conference on Weblogs and Social Media (ICWSM)*, 2012.

[Gas04] William I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the EATCS*, 82:72–107, 2004.

[GHW+10] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y. Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 35–47, 2010.

[GKBM10] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proceedings of IEEE INFOCOM '10*, San Diego, CA, March 2010.

[GO01] Gorr and A. Olligschlaeger. Crime hot spot forecasting: Modeling and comparative evaluation. Draft final report, U.S. Department of Justice, Office of Justice Program, National Institute of Justice, 2001.

[GPS] Gpscheat! http://www.gpscheat.com/.

[Gua] The Guardian. Uk riots: every verified incident. http://www.guardian.co.uk/news/datablog/2011/aug/09/uk-riots-incident-listed-mapped.

[Gut84] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, SIGMOD '84, pages 47–57, New York, NY, USA, 1984. ACM.

[HA01] H.Brian Hwarng and H.T Ang. A simple neural network for arma(p,q) time series. *Omega*, 29(4):319 – 333, 2001.

[Ham71] Frank R. Hampel. A general qualitative definition of robustness. *Annals of Mathematical Statistics*, 42:1887–1896, 1971.

[HHLM07] Ramaswamy Hariharan, Bijit Hore, Chen Li, and Sharad Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, SSDBM '07, pages 16–, Washington, DC, USA, 2007. IEEE Computer Society.

[HJR97] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Spatial joins using r-trees: Breadth-first traversal with global optimizations. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 396–405, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[Hor] Richard Hornsby. Florida criminal penalty chart. http://www.richardhornsby.com/criminal/penalties/.

[Hou87] Piet Houthuys. Box sort, a multidimensional binary sorting method for rectangular boxes, used for quick range searching. *The Visual Computer*, 3(4):236–249, 1987.

[HRRS86] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. New York: Wiley, 1986.

[IBM] IBM. Ibm smarter cities. http://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/index.html.

[IH93] B. Iglewicz and D. Hoaglin. *How to detect and handle outliers*, page 87. ASQC Press, 16 edition, 1993.

[IKE] IKE's Place. `http://www.yelp.com/biz/ikes-place-san-francisco`.

[iSa] iSafe: Context Aware Safety. `http://users.cis.fiu.edu/~mrahm004/isafe/`.

[Jef99] E. Jefferis. A multi-method exploration of crime hot spot: A summary of findings. Technical report, U.S. Department of Justice, Office of Justice Program, National Institute of Justice, 1999.

[JL08] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 219–230, New York, NY, USA, 2008. ACM.

[JLL10] Nitin Jindal, Bing Liu, and Ee-Peng Lim. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1549–1552, New York, NY, USA, 2010. ACM.

[JS] Edwin H. Jacox and Hanan Samet. Spatial join techniques. *ACM TODS'07*.

[KS98] Nick Koudas and Kenneth C. Sevcik. High dimensional similarity joins: Algorithms and performance evaluation. *Proceedings of the Fourteenth International Conference on Data Engineering (ICDE)*, February 1998.

[KSM05] Byung-Jae Kwak, Nah-Oak Song, and Leonard E. Miller. Performance analysis of exponential backoff. *IEEE/ACM Transactions on Networking*, 13(2), April 2005.

[KW10] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. *Computer Communication Review*, 40(1):112–117, 2010.

[Lab] MIT Media Lab. Smart cities. `http://cities.media.mit.edu/`.

[LAR12] 1992 Los Angeles riots. Wikipedia, at `http://en.wikipedia.org/wiki/1992_Los_Angeles_riots`, Last accessed on July 12, 2012.

[LHYZ11] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI '11, pages 2488–2493, 2011.

[LNJ+10] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 939–948, New York, NY, USA, 2010. ACM.

[LNS11] Hongrae Lee, Raymond T. Ng, and Kyuseok Shim. Similarity join size estimation using locality sensitive hashing. *Proc. VLDB Endow.*, 4(6):338–349, March 2011.

[LR94] Ming-Ling Lo and Chinya V. Ravishankar. Spatial joins using seeded trees. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 209–220, New York, NY, USA, 1994. ACM.

[Lue78] George S. Lueker. A data structure for orthogonal range queries. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, pages 28–34, Washington, DC, USA, 1978. IEEE Computer Society.

[Lyn11] Matthew Lynley. Yelp CEO: IPO window is still open, Yelp on track. Venture Beat `http://venturebeat.com/2011/09/13/yelp-ipo-on-track-disrupt/`, September 2011.

[MAT10] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[ME92] Priti Mishra and Margaret H. Eich. Join processing in relational databases. *ACM Comput. Surv.*, 24(1):63–113, March 1992.

[MLG12] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *ACM WWW*, 2012.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[Nau] Robert F. Nau. Decision 411 forecasting. http://www.duke.edu/ rnau/411avg.htm.

[Net] Netbeans. `https://netbeans.org/`.

[New12] IT News. Study: Geolocation apps draw users, despite privacy concerns. `http://www.itnews.com/mobile-applications/44048/study-geolocation-apps-draw-users-despite-privacy-concerns`, 2012.

[NH94] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 144–155, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[NLT] NLTK Project. Natural Language Toolkit. `http://nltk.org/`.

[NNMF06] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 83–92, 2006.

[oC] Florida Department of Corrections. Florida criminal punishment code. http://www.dc.state.fl.us/pub/sen_cpcm/cpc_manual.pdf.

[OCCH11] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Oll97] A.M Olligschlaeger. Artificial neural networks and crime mapping. Studies/research report, U.S. Department of Justice, Office of Justice Program, Drug Market Analysis Program, 1997.

[ON10] Christopher Olston and Marc Najork. Web crawling. *Found. Trends Inf. Retr.*, 4(3):175–246, March 2010.

[Pat10] Z. Patton. Sensors make cities smarter. `http://www.governing.com/topics/public-justice-safety/Sensors-Make-Cities-Smarter.html`, April 2010.

[PD96] Jignesh M. Patel and David J. DeWitt. Partition based spatial-merge join. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, SIGMOD '96, pages 259–270, New York, NY, USA, 1996. ACM.

[PM] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *VLDB 2000*.

[R D11] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[Sam90] H. Samet. Hierarchical spatial data structures. In *Proceedings of the first symposium on Design and implementation of large spatial databases*, SSD '90, pages 193–212, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[SF] E. Steel and G. Fowler. Facebook in privacy breach. `http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html`.

[SIG01] Bluetooth SIG. Specification of the bluetooth system, 2001.

[SSBL05] Tobias Sing, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. Rocr: visualizing classifier performance in r. *Bioinformatics*, 21(20):3940–3941, October 2005.

[SW09] Stefan Saroiu and Alec Wolman. Enabling New Mobile Applications with Location Proofs. In *Proceedings of HotMobile*, 2009.

[TBGE10] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *8th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98, 2010.

[TD00] A. C. Tamhane and D. D Dunlop. *Statistics and data analysis: From elementary to intermediate*. Upper Saddle River, NJ: Prentice Hall, 2000.

[Ter] Terrafly Project. Crimes and Incidents Reported by Miami-Dade County and Municipal Police Departments. `http:`

//vn4.cs.fiu.edu/cgi-bin/arquery.cgi?lat=25.81&long=-80.12&category=crime_dade.

[TMLS09] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association.

[Tri] Tripadvisor. http://www.tripadvisor.com/.

[TSK05a] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 1 edition, May 2005.

[TSK05b] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[TT02] Fang-Mei Tseng and Gwo-Hshiung Tzeng. A fuzzy seasonal arima model for forecasting. *Security Journal*, 126:367 – 376, 2002.

[UCL] Urban Sensing CENS UCLA. Walkability project. http://urban.cens.ucla.edu/projects/walkability/.

[VCL10] Rares Vernica, Michael J. Carey, and Chen Li. Efficient parallel set-similarity joins using mapreduce. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 495–506, New York, NY, USA, 2010. ACM.

[WAT] WatchYT: Watch Yelp Timelines. http://users.cis.fiu.edu/~mrahm004/watchyt.

[Wik12] Wikipedia. Yelp Inc. http://en.wikipedia.org/wiki/Yelp,_Inc., Last accessed on July 30, 2012.

[WXLY11] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. Review graph based online store review spammer detection. In *ICDM '11*, pages 1242–1247, 2011.

[XWLY] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. In *WWW'08*.

[YBL+08] Anna Yu, Athanasios Bamis, Dimitrios Lymberopoulos, Thiago Teixeira, and Andreas Savvides. Personalized Awareness and safety with mobile phones as sources and sinks. In *International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense08)*, 2008.

[Yel] Yelp. http://www.yelp.com.

[Yel09] Yelp. Yelp Elite Events: What's the deal? `http://officialblog.yelp.com/2009/03/yelp-elite-events-whats-the-deal.html`, 2009.

[YG09] Kyung Hyan Yoo and Ulrike Gretzel. Comparison of deceptive and truthful travel reviews. In *ENTER*, pages 37–47, 2009.

[YSK+09] Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B. Gibbons, and Feng Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 283–298, Washington, DC, USA, 2009. IEEE Computer Society.

[ZC11] Z. Zhu and G. Cao. APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-based Services. In *Proceedings of IEEE INFOCOM*, 2011.

[ZHLW09] Shubin Zhang, Jizhong Han, Zhiyong Liu, and Kai Wang. Sjmr:parallelizing spatial join with mapreduce on clusters. *IEEE CLUSTER 2009*, August 2009.

[Zic12] Kathryn Zickuhr. Three-quarters of smartphone owners use location-based services. Pew Internet & American Life Project, `http://pewinternet.org/Reports/2012/Location-based-services.aspx`, 2012.

VITA

JAIME BALLESTEROS

2005	B.Sc., Computer Science Pontificia Universidad Javeriana Bogota, Colombia
2012	M.S., Computer Science Florida International University Miami, Florida
2013	PhD., Computer Science Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

* **Jaime Ballesteros**, Ariel Cary, and Naphtali Rishe. *Spsjoin: parallel spatial similarity joins.* In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11, pages 481-484, New York, NY, USA, 2011. ACM.

* Bogdan Carbunar, Mahmudur Rahman, **Jaime Ballesteros** and Naphtali Rishe. *Private Location Centric Profiles for GeoSocial Networks.* Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Redondo Beach, CA, November 2012

* **Jaime Ballesteros**, Mahmudur Rahman, Bogdan Carbunar, and Naphtali Rishe. *Safe cities. A participatory sensing approach.* In Local Computer Networks (LCN), 2012 IEEE 37th Conference on, pages 626-634, 2012.

* **Jaime Ballesteros**, Bogdan Carbunar, Mahmudur Rahman, and Naphtali Rishe. *Yelp Events: Making Bricks Without Clay?* In Proceedings of the 5th International Workshop on Hot Topics in Peer-to-peer com- puting and Online Social Networking (HOTPOST). In conjunction with IEEE ICDCS 2013, Philadelphia, PA, July 2013. Best Paper Award.

* **Jaime Ballesteros**, Mahmudur Rahman, Bogdan Carbunar, Rahul Potharaju, Radu Sion, Nitya Narasimhan, and Naphtali Rishe. POSTER: *Towards Detecting*

Yelp Review Campaigns. In the 34th IEEE Symposium on Security and Privacy, San Francisco, CA, May 2013.

* **Jaime Ballesteros**, Bogdan Carbunar, Mahmudur Rahman, Naphtali Rishe, and S.S. Iyengar. *Towards Safe Cities: A Mobile and Social Networking Approach.* IEEE Transactions on Parallel and Distributed Systems, 2013. Accepted.