Selective Information Dissemination for Mobile Computing

André Panisson
Computer Science Department
University of Torino

Reference advisor: Prof. Giancarlo Ruffo
Co-Advisor: Dr. Ciro Cattuto (ISI, Turin, Italy)

A dissertation submitted to the University of Torino
for the degree of Doctor of Philosophy

Year 2011/2012

Abstract

Data-driven services are rapidly filling the worldwide mobile market with innovative and popular applications. In addition, users can communicate with each other by means of a broader set of wireless network interfaces. At the same time, the rapid proliferation of such mobile applications has created great demand for the limited spectrum of infrastructure networks, thereby leading to deteriorating quality for subscribers. Network operators have started facing network capacity issues, such as bottlenecks and reduced download speeds in densely populated urban areas and during peak usage hours. However, certain specific mobile applications may not require real-time connectivity. In addition, being constantly connected to access such applications leads to systematic battery overuse. In this scenario, opportunistic communications can be exploited to collect relevant information during off-line user's activity and to improve network performance in densely populated areas and during peak usage hours. Moreover, such approaches can be combined with selective information dissemination through distributed collaborative filtering algorithms.

The main purpose of this thesis is to understand how information can be effectively disseminated among communities of interest within the scope of mobile computing. Our first steps in this direction are to understand and statistically quantify the process of information diffusion in real-world human mobility networks. Based on the results of these first steps, we propose new strategies for selective data dissemination by using collaborative filtering approaches to reduce information overload and effectively send the useful information to communities of interest. We evaluate such strategies using experimental datasets, providing research findings in the scope of opportunistic networking that are useful in developing mobile computing applications. We show that this approach is effective in disseminating information of interest to communities with similar preferences. We also show that the selective nature of collaborative filtering approaches plays a role in reducing information overload.

Contents

1 Introduction **1**

1.1 Outline 2

2 User Mobility Datasets for Opportunistic Networks **5**

2.1 Empirical Datasets 7

2.1.1 Collecting mobility traces 8

2.1.2 Collecting contact traces 9

2.1.3 Available empirical datasets 10

2.2 Synthetic Datasets 11

2.2.1 Random Walk Model (RW) 12

2.2.2 Random Waypoint Model (RWP) 13

2.2.3 Random Direction Model (RD) 15

2.2.4 Truncated Lévi Walk (TLW) 15

2.2.5 Social Models 16

2.3 SocioPatterns – Active RFID-based experimental framework 17

2.3.1 Datasets 18

2.3.2 Meta data 19

2.3.3 Visualization 21

3 Human Proximity as Complex and Dynamical networks **24**

3.1 Complex Systems and Network Analysis 25

3.1.1 Graph theory for network analysis 27

3.1.2 Topological measures on graphs 28
3.1.3 Generative models 31
3.2 Temporal Networks 35
3.2.1 Measures of Temporal-Topological Structure 36
3.3 Proximity Networks 37

4 Exploring and Visualizing Dynamical Networks 40
4.1 Preliminaries 40
4.2 Visualizing Network Data 41
4.3 Representational Formats 43
4.4 Tools 43
4.5 Graph Streaming 47
4.5.1 Coupling Tools with Graph Streams 50
4.5.2 Real-Time Visualizations with Graph Streaming 51
4.5.3 Other Real-time Visualizations 53
4.6 Conclusion and Future Work 54

5 Impact of User Mobility in Opportunistic Data Dissemination 56
5.1 Related Work 59
5.2 Dynamics of information spreading 60
5.2.1 Information spreading process 61
5.2.2 Analysis methodology 63
5.2.3 Fastest Route Tree structure 63
5.2.4 Arrival times 64
5.2.5 Delivery time metrics 65
5.3 Comparison with data generated by synthetic models 69
5.4 Conclusion 72

6 Collaborative Filtering for Selective Information Dissemination 74
6.1 Background 76

6.2 Related work on Recommender Systems 77
6.3 Evaluating Recommendation Systems and algorithms 78
6.4 Content-based Recommendation 80
6.5 Classical Collaborative Filtering Algorithms 80
6.5.1 Memory-based algorithms 81
6.5.2 Model-based algorithms 82
6.6 Improved Collaborative Filtering Algorithms 83
6.6.1 Jointly Derived Interpolation 83
6.6.2 Matrix Factorization 84
6.6.3 Restricted Boltzmann Machines 85
6.6.4 Blending it all together 86
6.7 Conclusion and Discussion 86

7 A Practical Experience on Collaborative Filtering for Digital Recorders 87

7.1 The Experimental Environment 89
7.2 Data Extraction 91
7.3 Recommendation 93
7.3.1 Algorithms Overview 94
7.3.2 Computing Neighborhood 96
7.4 Experimental Results 98
7.4.1 Metrics 98
7.4.2 Evaluation 99
7.5 Conclusion 103

8 Epidemic Collaborative Filtering in Opportunistic Networks 105

8.1 Related Work 107
8.2 Similarity Networks 109
8.3 Collaborative Filtering via Opportunistic Communications 110

8.4 Experimental Analysis 113
8.4.1 Evaluation Methodology 116
8.4.2 Results in the Reference Scenario 117
8.4.3 Simulation of Ad-Hoc Scenario 119
8.4.4 Overhead and Scalability considerations 124
8.4.5 Impact of correlation between user similarity and contact frequency 124
8.4.6 Comparison with synthetic contact traces 126
8.5 Discussion 127

9 Conclusions and Future Work **128**

A Beyond Simulation: the MobHinter Implementation **152**
A.1 Technologies used in the prototype development 152
A.2 Software Architecture 153
A.3 Software Interface 155
A.4 Phonegap Plug-ins 156
A.5 iOS Version 158

B Mobility Model Implementations **159**
B.1 Random Walk Mobility Model 159
B.2 Random Waypoint Mobility Model 160
B.3 Truncated Lévi Walk Mobility Model 161

"It makes me so happy. To be at the beginning again, knowing almost nothing. People were talking about the end of physics. Relativity and quantum mechanics looked as if they were going to clean out the whole problem between them. A theory of everything. But they only explained the very big and the very small. The universe, the elementary particles. The ordinary-sized stuff which is our lives, the things people write poetry about - clouds - daffodils - waterfalls - ... these things are full of mystery, as mysterious to us as the heavens were to the Greeks."

—Valentine in Arcadia, by Tom Stoppard

Chapter 1

Introduction

The widespread adoption of rich data-driven services and powerful mobile devices has created unprecedented potential for innovative and popular mobile applications. Most mobile phones are now equipped with geolocation features, meaning that more and more applications and tools can use location-based services to bring together location and people in interesting ways.

For example, social networking services and self-organized communities are rapidly filling the worldwide mobile applications market with highly popular applications, many of which are based on data sharing. Even if the architectural models vary from rigid client/server implementations to pure distributed systems, users are invited to join a given community and to distribute their own files—such as photos (e.g., Flickr), videos (e.g., YouTube, Joost), music (e.g., eMule, Gnutella), blogs, personal data, and other information—and even to just be connected to other people (e.g., eBlogger, MySpace, Facebook).

In fact, more complex applications could take advantage of emergent community geosocial patterns, which can be inferred from the history of social interactions and the places people visit. Such applications could enhance and personalize a user's geosocial experience by, for example, recommending newly identified items or places.

In addition to using the myriad of data-driven services available for mobile phones, devices can communicate by means of an increasingly broad set of short- and wide-range wireless network interfaces (e.g., Bluetooth, Wi-Fi, WiMax, GSM, UMTS). Such interfaces enable users to communicate with each other by using short-range communications and at the same time accessing the Internet with higher bandwidths.

The rapid proliferation of such mobile applications has created great demand for the limited spectrum of infrastructure networks, leading to deteriorating quality for subscribers [203]. Network operators have started facing network capacity issues, with bottlenecks and reduced download speeds in densely populated urban areas and during peak

usage hours. From an environmental point of view, the systematic overuse of batteries due to 3G connections may transform constant connectivity into a disadvantage. Furthermore, certain mobile device application may not need real-time connectivity.

Nevertheless, there is an increasing demand for the use of opportunistic communications, which can be exploited to collect relevant information during times of ordinary off-line user activity and/or when WAN connectivity is difficult (e.g., underground, in an airplane), expensive, or simply superfluous. In addition, computers are embedded in smart vehicles to host vehicular computing and networking solutions addressing safer driving, dynamic route planning, and a new generation of entertainment applications in mobile computing.

As it turns out, device usage and spatial distribution have a direct impact on 3G connection cost and quality. However, the weakness of such networks when dealing with situations of high usage becomes an advantage when using opportunistic communications, since opportunistic networks perform best in densely populated areas and during peak usage hours [209]. Energy consumption is one of the most important issues for the deployment of mobile applications, and the use of such networks supports lower battery usage due to reduced range requirements. Moreover, hybrid architectures based on Wi-Fi hot zones and mixed zones with mobile-to-mobile communication may also be an interesting solution from a network operator's point of view.

In such a scenario, opportunistic networking can be combined with effective and selective information dissemination through adaptive collaborative filtering algorithms. This approach can play a leading role in the diffusion of information of interest to specific communities with similar preferences. At the same time, the selective nature of collaborative filtering approaches can support the reduction of information overload and help filter out useless information.

The main purpose of this thesis is to understand how information can be effectively disseminated among communities of interest [57] within the scope of mobile computing. As a first step, we try to understand and statistically quantify the process of information diffusion in real-world human mobility networks. Based on this understanding, we propose new strategies for selective data dissemination, using collaborative filtering approaches to reduce information overload and effectively send useful information to communities of interest. We evaluate such strategies using experimental datasets, providing findings that are useful in developing mobile computing applications through opportunistic communications.

1.1 Outline

This work presents the original research results of a three-year PhD program in computer science. The findings belong to several research macro areas, including the analysis of

human proximity networks, the analysis and visualization of complex and dynamical networks, collaborative filtering algorithms and recommendation systems, and information dissemination in opportunistic networks. The work is structured as follows.

Chapter 2 presents *user mobility datasets* to model and simulate information spreading over opportunistic networks. This is basically the raw material used to simulate and evaluate our hypothesis. From models used to generate *synthetic datasets* to *empirical datasets* created from real-world experiments, we show some of the common characteristics of the different types of data and their advantages and disadvantages. We also present the *SocioPatterns platform*, an active RFID-based experimental framework used to collect empirical data on human proximity, and the datasets collected using this platform that are used in the following chapters.

Chapter 3 presents the scientific tools used to model, analyze, and characterize the data presented in Chapter 2. Based on the multidisciplinary science of *complex systems* and *complex networks*, we show that *human proximity* data can be conveniently modeled as *dynamical networks* and that most of the concepts used in the analysis of static complex networks, later extended to the domain of dynamical networks, can be used to characterize these data. Chapter 4 discusses *techniques to explore and visualize* such data. We present tools that focus on analysis and visualization, some developed by the author, and which are used as instruments to understand the dynamics behind such networks.

Chapter 5 uses some of the scientific tools presented in Chapter 3 to analyze and characterize data presented in Chapter 2. We propose a novel type of analysis to understand and statistically quantify the process of *message spreading* in *real-world proximity networks*. This analysis takes into account user behavior heterogeneity, since it is visible and quantifiable in real-world collected data. The proposed analysis process shows results that are *universal across different experiments* and independent of the distribution of contacts throughout time. Finally, we show that some of the characteristics of synthetic models widely used to generate contact data differ from real-world data.

Chapter 6 discusses some *collaborative filtering* techniques used to obtain selective information dissemination. In order to understand the peculiarities of the collaborative filtering domain, it is extremely important to implement and test recommendation strategies that use collaborative filtering approaches in an experimental field. In one of the projects that took place during the PhD course, we had the opportunity to experience the complete lifecycle of a *recommendation application for personal video recording*. Chapter 7 describes these experiences, starting from context analysis and data extraction to the implementation of different recommendation strategies and their evaluation using a dataset of thousands of active users.

Chapter 8 presents an *epidemic collaborative filtering approach* that allows a mobile device to identify similar neighbors from opportunistic communications and exchange information in a *selective way*. Collected information is used to incrementally refine lo-

cally calculated recommendations without needing to interact with a remote server or access the Internet. In a simulated environment, we show how recommendation accuracies observed in the mobile domain using *experimental datasets* converge to values that are comparable to the best ones in the centralized scenario. Moreover, we empirically demonstrate how selective spreading strategies significantly reduce the cold start problem and perform similarly to epidemic strategies.

Finally, Chapter 9 draws the conclusions of our work and suggests interesting directions for further research.

Chapter 2

User Mobility Datasets for Opportunistic Networks

In this chapter, we present the user mobility datasets that can be used to model and simulate information spreading over opportunistic networks. This is basically the raw material that will be used to simulate and evaluate our hypothesis. From models used to generate synthetic datasets, to empirical datasets created from real world experiments, we show some of the common characteristics of each type of data, the advantages and the drawbacks of using them. We also present the SocioPatterns platform, an active RFID-based experimental framework used to collect empirical data of human proximity.

In the last decades, researchers have studied complex networking infrastructures based on layered models, such as the ISO/OSI model or the TCP/IP stack. Hence, during the fast and somehow unpredictable spread of computer networks along the world, scientists often coped with the need of abstracting the underlying structures in order to focus on a particular application or protocol. Without any a priori knowledge on the behavior of a given data transport channel, and before traffic invariants had emerged from exhaustive measurement studies, researchers have often opted for *random models* representing the complexity of unknown dynamics.

When routing in mobile and delay-tolerant networks became a hot topic, researchers assumed, once more, traffic and node mobility to be random. Unfortunately, in this case the underlying infrastructure cannot be trivially virtualized and flattened to a data transport channel, where communicating nodes are processes that respond to a given protocol. In fact, in this domain, nodes usually *piggyback* individuals that follow autonomous behaviors. However, by assuming the randomness of node mobility, models like *Random Walk* and *Random Waypoint* were considered as good approximations to reality.

Then, the study of social networks and other types of complex systems involving

human behavior showed that it was wrong to assume that node mobility is random. It was observed in collected empirical datasets that general characteristics such as the distribution of inter-meeting times between nodes show *broad distributions* [108]. Other statistical attributes, like pause times and inter-contact times, were found to be exponentially distributed, and some studies found power-law distributions. Models like *Lévy Walk* became a choice in order to reproduce some of the observed characteristics [112, 93].

At this point, synthetic datasets became a step closer to reality when observing general distributions of collected data, but little work was done to understand how these statistical attributes are relevant to the spreading process of messages. It was found, for example, that the structural properties of the aggregated network present a *scale-free structure* [47], exactly as the social networks studied in complex network disciplines. Moreover, this type of topology strongly influences the dynamics of message spreading [17]. These observations led the ad-hoc networking community to adopt *social models* to represent such behavioral patterns. The idea of improving multi-hop routing using social-based opportunistic delay tolerant strategies is extremely attractive indeed, and it has largely motivated many investigations during the last years. However, the understanding of dynamics of human interactions is a challenging task, which has not yet been explored in great depth.

In this context, there are two types of datasets used to evaluate information dissemination protocols in mobile networks: (i) *Empirical datasets*, collected by capturing data from the real behavior of mobile devices, i.e., by storing the real traces of mobile devices or (ii) *Synthetic datasets*, created by using mobility models, where we try to reproduce the moving behavior of mobile devices without using real mobility traces.

Real world data is invaluable to understand human interactions and complex patterns of human activities. It is also key to understanding how social dynamics occurs. It can be collected in different formats depending on specific context requirements. For example, one could be interested in temporal-spatial location, creating datasets with users' locations at different moments. Other could be interested in proximity sensing, and the dataset is formed by *dynamic contact networks* that reflect human interactions or human proximity. Hence, by capturing real traces, it is possible to observe real behavior of mobile devices.

However, in dynamic environments like MANETs, to collect real traces is not an easy task. Thus, sometimes it becomes necessary to use synthetic models in order to represent device mobility. Mobility models are essential building blocks in simulation-based studies of mobility networks. Researchers in this area can choose from a variety of models that have been developed in the wireless communications and mobile computing community during the last decades [24, 122, 94, 208, 133]. Moreover, well-known motion models from physics and chemistry – such as Random Walk or Brownian Motion [69] – as well as models from transportation theory [88] are used in simulations of mobile networks.

2.1 Empirical Datasets

Social interaction and proximity patterns among individuals have a direct impact on diverse phenomena studied in various research areas. Clear-cut examples are the transmission of infectious diseases by the respiratory or close-contact route and collective opinion formation. The availability of representative data on such patterns has long been a concern since it used to be notoriously difficult to collect it. Some available methods still rely on surveys and paper-diary methodologies [105], which are often slow, inaccurate, and intrusive.

Nevertheless, novel technologies such as Bluetooth and Wifi provide new and promising means of collecting proximity data. Several studies have demonstrated the potential of using these technologies for collecting data on both the structural and temporal aspects of proximity patterns [97, 66, 119, 179]. However, the spatial resolution in these experiments is at best on the order of 10 meters, and the temporal resolution is on the order of 2-5 minutes. Finally, these studies concern small groups and there is a scarcity of details and large sparsity in the collected data.

Empirical mobility datasets are indeed much needed. Recent studies of e-mail [67] and cellular phone call exchanges [149, 83], collaboration networks [144], and mobility by air travel [15], have revealed the presence of complex properties and heterogeneities. In particular, the number of interaction partners from one individual to the other is subject to large fluctuations that have non-trivial consequences on the dynamical processes taking place on these networks [153, 29, 17]. A detailed characterization of these structures is, therefore, of utmost importance for the understanding of many phenomena, and crucially depends on the availability of representative empirical data.

As mobile phones are carried by the same individual during their daily routine, mobile phone datasets are used to explore the mobility pattern of individuals [83]. They found that the distribution of displacements over all users is well approximated by a *truncated power-law*. However, this approach has some limitations, as (a) the irregular call pattern formed by the time between consecutive calls, and (b) the service area of each tower, which covers an average area of approximately $3km^2$. This last restriction prevents us to provide the level of detail required to study the achievement of information exchange through mobile devices, which can reach other devices in a short range. In this latter case, study of social mobility cannot be addressed by using such datasets. Nevertheless, the understanding of short range interaction patterns is relevant to implement data exchange.

Yoneki [204] also points out the importance of collecting real world data when studying contact networks. He lists some experimental data sets available for analysis, where most of them use Bluetooth to measure device connectivity, while others rely on WiFi. While Bluetooth experiments give special importance to collect short-range proximity contacts, it is not possible to affirm that such data can be used to analyze human interactions, as its range is much larger than face-to-face contacts. He also emphasizes that real-world

data needs to drive modeling, and the derived network models should be accurate and parametrized with the real data.

Kim et al. [111] extract mobility models from user traces, focusing on node localization and path tracing: the analyzed characteristics are node speeds and pause times that follow a *log-normal distribution*. Hui et al. [97] present an experiment that involved about 40 participants at the Infocom 2005 conference, and report *power-law distributions* for the time intervals between node contacts.

The feasibility of collecting real world data on human proximity given by the current technologies can allow us to understand complex patterns of human activities. We can focus on two different approaches when collecting experimental data: (i) by collecting detailed data about *node localization* and *path tracing* or (ii) by collecting only *user proximity* data. Therefore, we will concentrate on these approaches and list some of their advantages and shortcomings.

2.1.1 Collecting mobility traces

When collecting mobility traces, detailed characteristics about user mobility are gathered: position, direction, velocity, pause times. The highest is the required level of accuracy, more complex infrastructures and more expensive devices with special sensors are required. For example, in order to collect device position, sensors like *GPS* or radio-based positioning could be used. It is also possible to use sensors like accelerometers to obtain details like velocity, direction and pause time.

Mobile network operators are increasingly interested in developing *location based services* for mobile devices, and one way to improve such services is by being aware where a user is located. The combination of Geographic Information Systems, Internet, wireless communication, location finding techniques and mobile devices made a major impact on the level of positioning accuracy. Particular services have different position accuracy requirements, and this should be acquired at the lowest possible cost and with minimal impact on network and the equipment.

Distinct technologies give different levels of positioning accuracy. For example, in data collected from GSM/UMTS Cellular Networks, the accuracy depends on the type of cell where the mobile device is located, and the position of a user can be determined using various techniques. By using cell identification, angle and time of arrival, the achieved positioning accuracy is typically in the range of 50-150m [117]. More advanced techniques like *Assisted GPS*, which involves more expensive handsets, can achieve an accuracy in the range of 10-100m.

Depending on the required level of accuracy and the experiment's environment, more accurate positioning methods are needed. The level of accuracy of GPS sensors, for example, is approximately 20m, and the service can be strongly affected in indoor ex-

periments. In this case, to collect data with a higher positioning accuracy and in indoor environments, a complex infrastructure must take place. By using the *Received Signal Strength Indication* technique [181], for example, a set of static reference nodes is placed at preset coordinates. The average error of coordinates computed in one experiment using this method is around 1.7m.

When user mobility traces are available, simulation and evaluation of different protocols can be done with the collected data. With knowledge of user position, user contacts are simulated by setting different parameters of transmission range. In mobile networks, a device can only send data to another device if both devices are within transmission range of one another. However, this type of simulation neglect many phenomena that may occur during transmission, causing it to fail (interference, physical obstacles, power problems, etc.). All these phenomena can affect data transmission, and it is difficult to take them into consideration when collecting mobility traces. As a result, most of the times, simulations of data spreading using mobility traces simply omit these phenomena, and always consider as a contact when two devices are within a transmission range.

2.1.2 Collecting contact traces

An alternative approach for collecting all user mobility details is based on collecting only *user contacts*. In this case, we choose to collect contact traces in place of mobility traces, and we omit all details about position, direction, velocity, interference and obstacles in order to keep up only the details strictly necessary to simulate communication between nodes. Most of the times, sensors like Bluetooth, already present in mobile devices, can be used to acquire such data.

The restricted number of communication range values is one limitation of this approach. For example, by using Bluetooth devices to contact other devices around, the contact range is fixed in 5-10 meters. If we need to know what would be the results of a communication protocol by using devices that communicate in a lower range of, let's say, 2 meters, it is necessary to run the experiment again using a distinct setup, with sensors configured to generate contacts in different ranges. This limitation does not occur when collecting mobility traces, because we can produce contact traces in different ranges once the mobility traces are available.

One way to overcome this problem and represent contacts in different ranges is by collecting contacts with their *radio power level* when received by the destination (e.g., in dBm units). In this way, it is possible to fix distinct radio power thresholds to create datasets that represent different contact ranges. Moreover, by using sensors which are able to measure the strength of received packets, it is possible to reduce the number of transmitted packets and increase the number of ranges measured during a single experiment.

2.1.3 Available empirical datasets

The *Community Resource for Archiving Wireless Data At Dartmouth*, a.k.a. CRAWDAD [120], was created to cope with exigences from researchers who work with wireless networks or mobile computing. Most research was based on analytical or simulation models, severely limited by the complexity of real-world radio propagation and the lack of understanding about behavior of wireless applications and users. Researchers were seriously starved for data and in need of a wireless network data repository. Therefore, the CRAWDAD repository was turned into a community resource; an archive with the capacity to store wireless trace data from many contributing locations, with a staff to develop tools for collecting, anonymizing, and analyzing the data.

Currently, there are tens of datasets available on this repository. But some datasets are particularly interesting to our work. Following, we list some datasets with useful information about user mobility that can be used to simulate spreading processes over it.

Cambridge/Haggle

In [97] the authors describe an experiment that involved forty-one participants at the Infocom 2005 conference, and report power-law distributions for the time intervals between node contacts.

They used the *Intel iMote* device to collect user contact traces and mobility statistics. Intel iMote is a small platform designed for embedded operation, comprising an ARM processor, Bluetooth radio, and flash RAM. The devices were packaged in dental floss boxes to be worn by participants of the experiment. Fifty-four boxes were distributed to attendees at the IEEE Infocom conference in Miami in March 2005 – a conference with a total of 800 attendees. Most of them carried the boxes on their pockets, keeping the devices with them for as much time as possible during the conference. The volunteers were chosen from more than thirty different organizations. Amidst some failures with battery and packaging, 41 devices yielded useful data at the end of the experiment.

The user contact traces were collected by performing a Bluetooth baseband layer "inquiry" discovering the MAC addresses of other Bluetooth devices in range. The inquiry duration was of *five seconds*, and after it the devices were placed in sleep mode for a duration of 120 seconds, ± 12 seconds in a uniform random distribution. The randomness was added in order to avoid *inquiry synchronization*, since two devices performing inquiry simultaneously cannot see each other. The collected contacts were stored locally in flash RAM as contact period tuples, containing MAC address, start time and end time. A total of 22459 contacts were recorded between iMotes. An anonymized version of the dataset was released to the public and is available at the CRAWDAD repository.

MIT/Reality Mining

In order to log Bluetooth MAC addresses of mobile phones, Eagle et al. [66] designed the *BlueAware application*, that runs passively in the background on MIDP2-enabled mobile phones, more specifically on Symbian Series 60, performing repeated Bluetooth scans of the environment every 5 minutes. The application records and timestamps the Bluetooth MAC addresses encountered in a proximity log and makes them available to other applications, similarly to the Jabberwocky project developed by Paulos et al. [154].

Bluedar, on the other way, is a device that is independent of a mobile phone, comprised of a Bluetooth beacon coupled with a WiFi bridge. It can be placed in social settings and continuously scan for visible devices. After collecting the Bluetooth MAC addresses, it sends the results over a 802.11b network to the Reality Mining server. With a Bluetooth chipset class 2, it is able to detect any visible Bluetooth device within a range of up to 25 m.

The Reality Mining experiment is one of the largest mobile phone projects attempted in academia. The data was collected from 100 Nokia 6600 smart phones over the course of 9 months. Seventy-five users are either students or faculty in the MIT Media Laboratory, while the remaining twenty-five are incoming students at the MIT Sloan business school adjacent to the Laboratory. The study has generated data collected by 100 human subjects over the course of the 2004-2005 academic year that represent over 350,000 hours (40 years) of continuous information about users' location, communication and device usage behavior. The anonymized dataset was released to the public and is available at the CRAWDAD repository and at the project site[1].

2.2 Synthetic Datasets

Capturing user traces enables the observation of mobility patterns present in real life systems, but it could be a difficult task when they involve a large number of participants and a long observation period. However, protocols and applications for mobile networks are not easily modeled, and its performance is not easily evaluated if user traces have not been created yet. In this case, in order to produce user traces without recurring to complex infrastructures for data collection, it is necessary to use *synthetic models*. Synthetic models attempt to represent realistically the behavior of mobility nodes without the use of user traces.

Camp et al. [42] present several synthetic mobility models that have been proposed and used in the performance evaluation of ad hoc network protocols. The authors emphasize the need to devise accurate mobility models in order to represent realistic movements of the mobile users, exploring the limitations of current modeling strategies. They clas-

[1] `http://reality.media.mit.edu`

sify mobility models in two types: *individual mobility models* and *group mobility models*. Individual mobility models or *memoryless models* [24, 208] describe nodes whose actions are independent from each other. These are the simplest and most used models when evaluating wireless mobility networks, as each device is independent of other devices and there's no correlation with others behavior. On the other hand, group mobility models [94, 158, 197] are more complex and less used and aim at representing the behavior of nodes as they move together.

Rhee et al. [169] model human contact networks using a generative model of human walk patterns based on *Lévy flights*, and reproduce the fat-tailed distribution of inter-contact times observed in empirical data of human mobility. This model is later used to characterize the routing performance in human-driven DTNs [93], predicting the message delivery ratio.

2.2.1 Random Walk Model (RW)

The starting point of the research on random walk is usually attributed to the letter that Karl Pearson sent to Nature asking help with the "recurrence" problem [157], and the relative answer from lord Rayleigh a week later [167]. However, the first theory describing the Random Walk model was put forward by Einstein in 1905 [69], as *Brownian Motion*. It was developed to mimic the random drifting of particles suspended in a fluid.

In this mobility model, we start with a rectangular system area of size $X_{max} \times Y_{max}$, and a total number of nodes N. In each step, a node moves from its current position to a new position by randomly choosing a direction and speed in which to travel. The speed is either constant or randomly chosen from a uniform distribution, and the direction angle is chosen from the $[0, 2\pi]$ uniform distribution. Each step in the Random Walk model occurs in either a constant time interval t or a constant traveled distance d [18, 79], at the end of which a new speed and direction are calculated. Listing B.1.1 in Appendix B shows an example of implementation in Python using a constant distance traveled in each step.

The Random Walk Mobility Model is a memoryless mobility pattern that retains no knowledge about its past positions and speed values. This aspect leads this model to produce unrealistic movements such as sudden stops and sharp turns. If the specified step time interval (or traveled distance) is short, then the movement pattern is a *random roaming pattern*, restricted to a small portion of the simulation area. This pattern is useful to investigate the performance of semi-static networks. If the goal of the investigation is to evaluate a more dynamic network, then a larger value should be used for the specified step time or distance.

Many derivatives of the Random Walk model have been developed, including d-dimensional walks. In the case of mobile network simulations, the two-dimensional

derivative is used. Two variants proposed by Nain et al. [143] also consider different strategies when the node reaches a simulation boundary:

Wrap Around Model In the wrap around model, when the mobile hits the edge, it wraps instantaneously to the opposite edge and continues its movement with the same direction and speed, by creating a torus-shaped simulation area. As a result, the direction in which the mobile is moving remains unchanged between two consecutive movements.

Reflection Model In the reflection model, the node bounces off the simulation border with an angle determined by the incoming direction. When a mobile node reaches any edge of the simulation area, the node changes its angle of movement and its velocity remains constant. The approach employing reflection could generate more accurate movement patterns, if you consider that in real life mobile nodes are more likely to reflect their movement when reaching obstacles.

2.2.2 Random Waypoint Model (RWP)

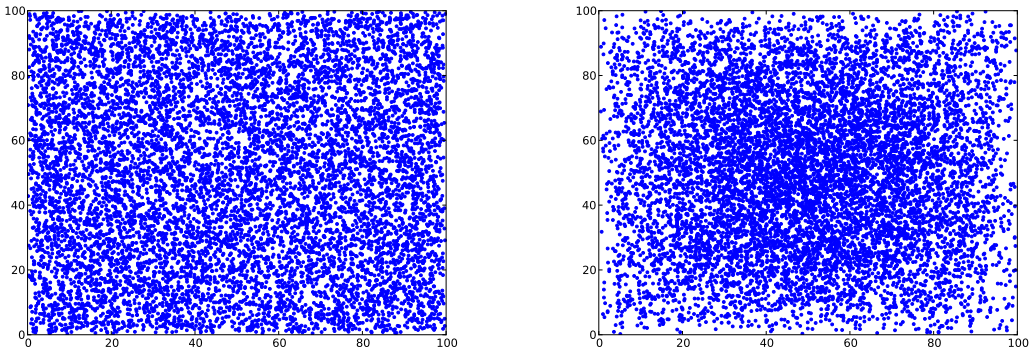
A popular and commonly used mobility model is the Random Waypoint model (RWP). This mobility model is a simple and straightforward stochastic model that describes the movement behavior of a mobile network node in a given system area.

In this mobility model, we start with a rectangular system area of size $X_{max} \times Y_{max}$, and a total number of nodes N. Each node starts in a random initial position (x, y), where x and y are both uniformly distributed over $[0, X_{max}]$ and $[0, Y_{max}]$, respectively. Every node is then assigned a target (x', y'), randomly chosen over the system area, and a speed v, which is either constant or randomly chosen from a given distribution. A node will then start moving toward the destination on a straight line at speed v. Upon reaching the target (x', y'), the node stays there for a pause time p, either constant or randomly selected from a given distribution. Upon the expiration of the pause time, a new destination and speed are chosen in the same way, and this process repeats until the simulation ends. Listing B.2.1 in Appendix B shows an example of implementation in Python.

The mobility model parameters directly affects the performance measures of routing protocols, and one of the most relevant parameters is the *node speed*, either as a constant value or a specific distribution [161, 42]. In the RW model, the average node speed is often believed to be half of the maximum speed, or some value between 0 and V_{max} when node speeds are chosen from a uniform distribution $(0, V_{max}]$. However, simulations actually show that the average speed consistently decreases if the lower bound of the distribution V_{min} is zero [205]. In this case, it was proven that the system does not reach a steady state: the average node speed consistently decreases on time. In order to reach a positive

average speed, the typical solution is by setting a *positive minimum speed* for the mobility model. There will be a period of speed decay at the beginning of the simulation; if V_{min} is positive, the average speed eventually stabilizes to a positive v, and if V_{min} is close to zero it may take a long time before the average node speed stabilizes. In general, the more the minimum speed is close to zero, the larger will be the period of speed decay. As it is desirable for the simulation model to reach stability as soon as possible, the minimum speed should not be too small. In this case, the average node speed quickly converges to a constant and stable value.

The *spatial distribution of nodes* resulting from their RWP movement in the rectangular area is another characteristic that demands a proper warm up time in order to reach stability. At the beginning of a simulation, all nodes are typically uniformly distributed. However, as it has been observed in [25, 28, 171], this distribution changes as the nodes start to move. Figure 2.1 shows the difference in the spatial distribution of nodes for the RW model in the start of the simulation and after 1000 steps. This is because nodes close to the border have a higher chance to find target waypoints in directions towards the center of the area. As time goes on, the node distribution get more and more nonuniform, with a maximum in the middle of the area and a low probability density at the border. Finally, for a long running time of the movement process, a stationary distribution is achieved [24]. Another reason for the nonuniform spatial distribution of nodes is the nonuniform distribution of the chosen direction angles. As pointed in [26], the probability density function of the chosen angles is determined by the shape of the system area and the starting waypoint of the node.



(a) (b)

Figure 2.1: Spatial distribution of nodes for a Random Waypoint model simulation in an area of 100×100 and with ten thousand nodes. (a) shows the spatial distribution in the start of the simulation and (b) shows the spatial distribution after 1000 steps.

The main shortcoming of this non-uniform spatial distribution of nodes is the production of *density waves* in the average number of neighbors. Depending on the node

position in the simulation area, the node has different probabilities of having a number of neighbors. While the nodes passes near the center, the probability increase, and when the node reach positions near the border, the probability decreases. This behavior is undesirable in most of the cases, and some solutions were proposed to overcome this problem.

2.2.3 Random Direction Model (RD)

The Random Direction Model [171] was proposed as a solution to overcome the non-uniform spatial distribution of nodes produced by the Random Waypoint Model and, as consequence, to avoid density waves in the average number of neighbors. In order to promote a uniform spatial distribution of nodes, we choose a *random direction* in which to travel instead of a random destination.

In the same way as the Random Waypoint model, we start with a rectangular system area of size $X_{max} \times Y_{max}$, and a total number of nodes N. Every node starts in a random initial position (x, y), where x and y are both uniformly distributed over $[0, X_{max}]$ and $[0, Y_{max}]$. Each node is then assigned a direction angle, chosen from the $[0, 2\pi]$ uniform distribution, similarly to the Random Walk model, and a speed v, which is either constant or randomly chosen from a given distribution. After choosing the direction angle, the node travels to the border in that direction at velocity v. Once the node reaches the boundary, it stops there and chooses a pause time p, either constant or randomly selected from a given distribution. Upon the expiration of the pause time, a new direction and speed are chosen in the same way, and this process repeats until the simulation ends.

Since the nodes travel to, and usually pause at the border, the average distances between mobile nodes are much higher than other models. As a consequence, the average hop count for data packets will be much higher than other models, and network partitions will be more likely. To overcome this limitation, a slight modification to the Random Direction model was proposed [171]. In the modified version, each node chooses a random direction and selects a destination anywhere along that direction of travel. The node then pauses in that destination before choosing a new direction.

2.2.4 Truncated Lévi Walk (TLW)

Humans hardly move in random manners, but it is possible to extract some Lévy walk patterns from empirical data. Human trajectories are often approximated with various random walk or diffusion models. Early measurements on nature suggested that animal trajectory is approximated by a Lévy flight [112], and this finding has been generalized to humans [169]. By using these results, Hong et al. [93] devised the Truncated Lévy Walk model (TLW), which is a variant of the Random Waypoint model with the following

characteristics:

- Flight lengths follow a truncated power law with exponent α: $p(l) \sim |l|^{-(1+\alpha)}, l < l_{max}$
- Pause times follow a truncated power law with exponent β: $\psi(t) \sim t^{-(1+\beta)}, 0 < t < t_{max}$
- Turning angles follow a uniform distribution: $[0, 2\pi]$
- Velocity increases as flight lengths increase

It is known that the *Inter Contact Time* (ICT) distribution of human walks exhibits a power-law tendency up to some time after which it shows exponential decay [107]. The ICT distribution generated from the TLW model exhibits a proper fit to empirical ICT distributions [169]. Although there could be other types of mobility patterns that could generate the same ICT distributions, the actual mobility that generates these characteristics in empirical data is more closely modeled by Lévy walks than Random Walk.

Furthermore, the ICT distribution patterns of various mobility models are closely related to their *diffusion rates*. In RWP model, the mobility is the most diffusive and in RW it is the least. In TLW, the diffusivity is in-between, and with a smaller value of α, it becomes more diffusive. The more diffusive the mobility is, the shorter tail its ICT distribution becomes.

2.2.5 Social Models

Synthetic mobility models such as the Truncated Lévi Walk produce traces closer to reality when observing general distributions of collected data. However, while these statistical attributes are relevant to the spreading process of messages, other applicable structural properties of the aggregated network display nontrivial topological features, like a *scale-free structure* [47]. Moreover, these network topologies strongly influence the dynamics taking place on the networks [17]. These observations led the mobile networking community to adopt *social models* to represent such behavioral patterns, and a number of papers have been devoted to modeling the dynamics of social interactions. In particular, community formation [121, 106] and the evolution of dynamics of opinions and social ties [61, 134, 91, 195] are important issues being investigated in this context.

An interesting approach for modeling contact traces is presented by Stehlé et al. [187]. In this approach, social behavior of nodes is simulated by agents in social interaction, producing dynamical and bursty contact networks. These networks are formed by disconnected cliques of different sizes. In each step, agents are randomly chosen to make

transitions from being isolated to being part of a clique or vice versa. By using this approach, different distributions of contact durations and inter-contact durations can be obtained by considering different transition probabilities. More than a simple mobility model, this is a complete modeling framework that can be easily extended, opening perspectives for systematic investigations of dynamical processes occurring over dynamical networks.

2.3 SocioPatterns – Active RFID-based experimental framework

In this section we present the *SocioPatterns platform*[2] [47], an experimental framework aimed to gather data on face-to-face interactions between interacting individuals with high spacial resolution in a spatially bounded setting, such as a set of offices or a conference.

During a SocioPatterns experiment, the participants are asked to carry small RFID tags [74] embedded on badges, henceforth called *beacons*, to collect short range proximity relations. These beacons continuously broadcast ultra-low power radio packets in an effective sampling frequency under one second. Furthermore, they sense their neighborhood and assess directly packets exchanged with nearby tags. This way, they are not only simple beacons that passively emit signals to be received and processed by a centralized set-up; they rather exchange messages with one another in a peer-to-peer fashion.

When individuals wear the beacons, persistent packet exchanges between the RFID devices can be used as a proxy for proximity information. The spatial resolution is tunable; a variable spatial resolution is attained by setting different radio power levels on the proximity sensing devices. Configurable radio power levels can reach ranges from several meters down to face-to-face proximity of less than $1 - 2$ meters. At the highest spatial resolution, by using ultra-low radio power transmission, the human body acts as a RF shield at the carrier frequency used for communication. In this case, by assuming that the subjects wear the badges on their chest, exchange of radio packets between badges is only possible when two persons are at close range and facing each other.

Regardless of the proximity range settings, when a relation of proximity (or "contact", as we will refer to in the following) is detected, the devices broadcast a report message at a higher power level. The reports contain a time stamp, the ID of the relaying station and the ID of the tags which participate in the contact event. These reports are received by stations installed at fixed locations in the environment (*RFID readers*). The readers can store reports locally to be processed afterwards, or they can relay the reports to the monitoring infrastructure by means of a Local Area Network in order to process the

[2] `http://www.sociopatterns.org`

information for live visualization.

The devices are programmed in such a way that the proximity of two individuals wearing badges can be assessed with a probability in excess of 99% over an interval of 20 seconds. This time scale is fine enough to resolve social interactions at social gatherings. False positives are exceedingly unlikely, as the radio packets used for proximity sensing cannot propagate further than the programmed range, and a sustained number of packets is needed in order to signal a proximity event. If a sensed contact persists for a few seconds, then, given the short range and the face-to-face requirement, it is reasonable to assume that the experiment is able to detect an ongoing social contact (as e.g. a conversation). Once a contact has been established, it is considered ongoing as long as the involved devices continue to exchange at least one radio packet for every subsequent interval of 20 seconds. Conversely, a contact is considered terminated if an interval of 20 seconds elapses with no packets exchanged.

The RFID beacons and RFID readers used in SocioPatterns experiments were created by and obtained from the *OpenBeacon project*[3]. For a more detailed description of the sensing platform and some of its deployments, see Ref. [47, 191, 5]. Figure 2.2 shows an image of the tag used in one of the experiments.

One of the first deployments of the measuring infrastructure took place during the workshop "Facing the challenge of Infectious Diseases" at the ISI Foundation on October 13 − 17, 2008. Participants to the workshop were offered to volunteer to participate to the experiment, and a large part agreed. This allowed us to gather data in a very dynamical context with periods of high social interaction (coffee and lunch breaks) and other periods in which the participants sit together but (almost) do not interact in a pairwise fashion. The experiment involved about 50 attendees over four days. The reporting stations were placed in the main areas in which people were expected to be during the sessions and breaks – namely the conference room, the bar (where coffee breaks were taking place) and the cafeteria area (where lunch was served). A station was also positioned in the lobby which is also suited for discussions (see Figure 2.2).

2.3.1 Datasets

The SocioPatterns RFID platform was deployed at different events to collect data at gatherings of different scale, with different proximity-sensing ranges. The first deployment took place at the 25$^{\text{th}}$ Chaos Communication Congress (*25C3*) in Berlin, Germany, from December 27th to December 30th, 2008. Proximity between RFID badges was recorded within a comparatively long range of 10-12m. The second deployment was at the XX$^{\text{e}}$ Congrès de la Société Française d'Hygiène Hospitalière (*SFHH*) in Nice, France, on June 4th and 5th, 2009. In this case, contacts between individuals were detected

[3] `http://www.openbeacon.org/`

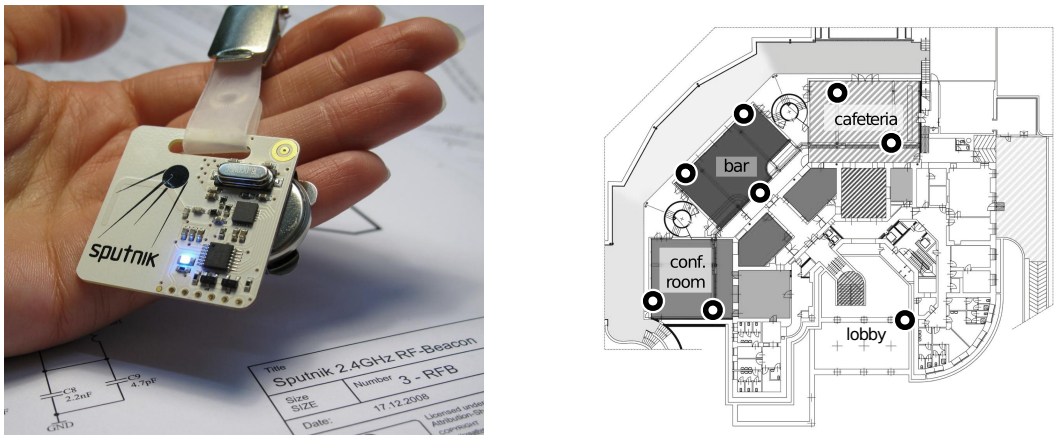


Figure 2.2: Left: photo of a beacon (Courtesy of M. Meriac, OpenBeacon project). Right: map of the experiment premises. The circles denote the positions of the reporting stations in one of the deployments.

Event	Event Type	Participants	Hours	Contacts
25C3	conference/gathering	684	57	1,457,520
SFHH	conference	413	32	199,966
HT09	conference	113	77.5	41,276

Table 2.1: Characteristics of the data sets.

based on face-to-face proximity within 1-1.5m. The third deployment happened at the 20$^{\text{th}}$ ACM Conference on Hypertext and Hypermedia (*HT09*) in Turin, Italy, from June 29th to July 1st 2009. Also in this case, contacts were recorded when individuals were in close-range face-to-face proximity. Table 2.1 reports some quantitative features of the data collected at the above gatherings.

It is important to remark that the SocioPatterns platform does not perform accurate spatial localization and trajectory tracing. Rather, it focus on accurately mining for proximity between individuals, i.e., on topological and temporal properties of mobility and not on metric properties. While other approaches use information about node localization to calculate node proximity, this platform directly sense and record "contacts" between nodes, using the exchange of low-power packets as a proxy for contacts.

2.3.2 Meta data

In some SocioPatterns experiments, a further step was accomplished by integrating data collected from the contact sensing platform with data from semantic web and online social networks [191]. This setup was successfully deployed at HT2009 and other conferences. Personal profiles of the participants were automatically generated using several Web 2.0

systems and academic data sources, and integrated in real-time with the face-to-face contact network derived from wearable sensors. Integration of these heterogeneous data layers made it possible to offer various services to conference attendees to enhance their social experience such as visualization of contact data, and a site to explore and connect with other participants.

Collecting user meta data is particularly important in this work, because it offers a further way to calculate user similarities and to infer user interests, as we will see in Chapter 6. As the amount and variety of semantic data available on the web is continuously growing, data from various conferences (e.g. ESWC, ISWC, WWW) has been consistently collected and published in recent years [140], and can be retrieved from sites such as `data.semanticweb.org`. This information has been merged with data from several publication databases (e.g. CiteSeer, DBLP) by the RKBExplorer system [81].

Social relationship information from online social networks could provide a useful substrate for constructing social services. However, since such networks generally capture only part of the actual social network, meshing this information with data coming from real-life social activities would greatly improve this potential. To this end, the Live Social Semantics experiment [191] was designed as the blending of data gathered by the SocioPatterns platform with data available in the TAGora project [130]. This experiment not only provided participants with various novel services, such as logs and summaries of their social interactions, but also provided participants to integrate this with information from people's social profiles of interest, scientific communities of practice, and their online social contacts. At HT2009, the semantic web, the social web, and the physical world were brought together to create a rich and integrated network of information. Acquiring and integrating these heterogeneous, but overlapping, data sources provided a new experience and services to conference attendees. The main goal was to encourage conference participants to network, to find people with similar interests, to locate their current friends, and to make new ones.

The Live Social Semantics experiment was also deployed for 4 days (1-4 June 2009) during the European Semantic Web Conference (ESWC), which was located in Crete. More than 300 people attended the conference, out of which 187 accepted to participate in using our application. Each participant was issued with a uniquely numbered RFID badge. Users were asked to enter their RFID ID number on a website dedicated to this social application. On this website, users were also able to provide their Delicious, Flickr, and lastFM account names, as well as activating a Facebook application that collected their social contacts. Out of the 187 participants who accepted to take part in the experiment by wearing RFID badges, 139 of them also created accounts in the application site.

Data from various Web 2.0 sources were imported using online APIs or screen scraping, and subsequently converted to RDF. The aim was to provide a service endpoint that supports the collection and reasoning over the data. Data in the virtual world was

sourced from social networking sites, to obtain social tagging data and contact networks, as well as the Semantic Web, to obtain information about publications, projects, and co-operations. The framework also processed an individual's interests (e.g., favorite music artists) their tagging activities. In turn, the framework automatically suggested to users a list of interests that they could edit, and elect to expose to other participants.

For privacy concerns, permission was sought from all participants for collecting and using their data. A form was prepared which explained what the data is, how it was going to be used, and for how long. Users were shown how the RFID badges are used, and the geographical limits of where their face-to-face contacts can be detected (conference building). When creating an account on the application site, each user was given the option of destroying their data after the end of the event.

2.3.3 Visualization

Some of the conducted SocioPatterns experiments were accompanied with publicly displayed dynamic visualizations of the contacts between individuals. This is achieved by defining a *dynamic contact network* in which the beacons/persons are nodes and the contacts are edges.

Two different types of *real-time visualizations* were provided. The first, the spatial view, was publicly displayed on large screens in the main lobby area. The second, the user focus view, was accessible by means of a web browser on the available local network, and is linked to from each user's account page on the application site. Both are dynamic visualizations driven by regular updates received through a TCP socket connection with the local post-processing server.

The *Spatial view* (Figure 2.3) provides an overview of the real-time contact graph. It represents the RFID-badge wearing participants within range of the RFID readers, as well as ongoing social contacts. Each participant is represented by a labeled yellow disc or, when available, by the Facebook profile picture. The contacts are represented by edges whose thickness and opacity reflects the weight of the contact. The edges are decorated, where applicable, with small icons representing the social network from which the relationship was retrieved (e.g., Facebook, Flickr, lastFM), marking the occurrence of that relationship in the respective social network. This visualization is primarily concerned with the contact network topology, and the precise position of the participants is of lesser concern. However, an approximate localization of the participants is achieved by representing the RFID readers in a fixed layout and the badges next to the RFID readers that receive their reports. The location of the participant is driven by a *force-directed layout* algorithm [96], where springs are associated with both the explicitly shown contact edges and the edges between beacons and stations, which are not shown, and the length of these springs is inversely proportional to the strength of the respective contact or beacon-station proximity estimations. This approach adds spatial structure to the

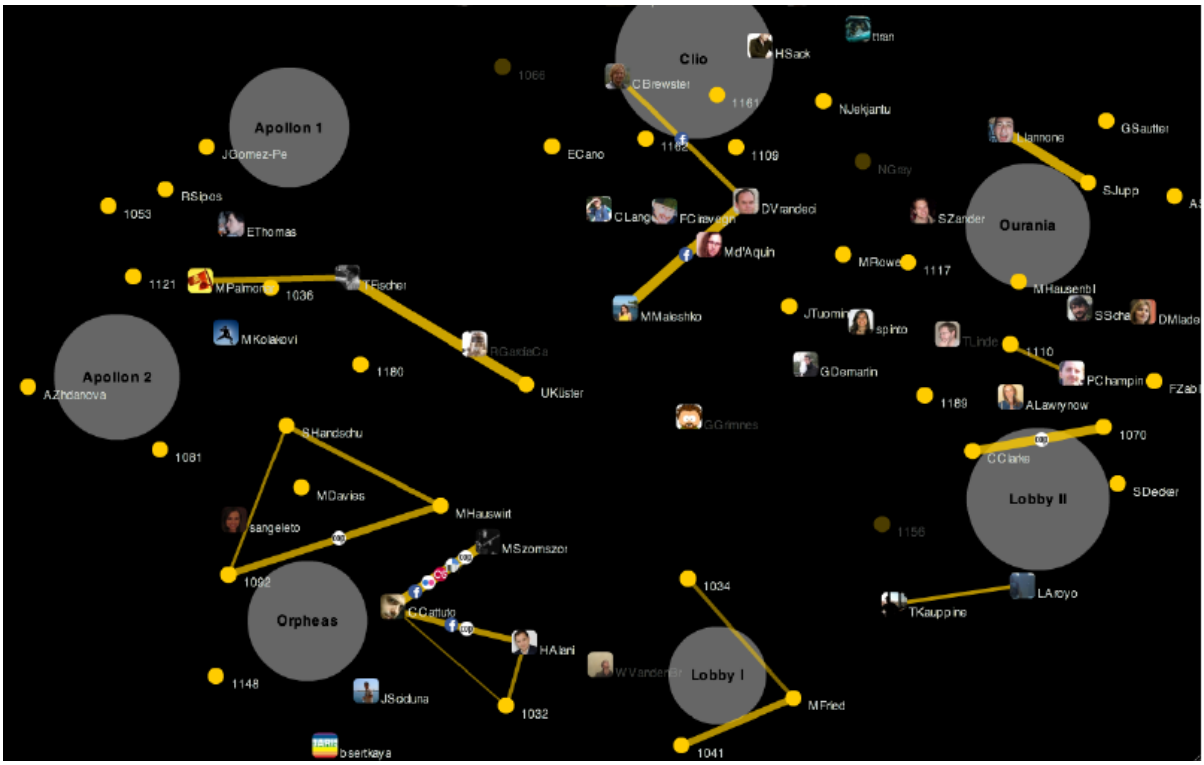


Figure 2.3: A snapshot of the spatial view grabbed during a session. This view shows the instantaneous state of the contact network at a given time. The beacons are labeled with available metadata (such as e.g. the actual names of the persons) or as their IDs if metadata was not given. The edges represent relations in social networks or contacts in the SocioPatterns framework. The beacons are positioned near the stations from where their signals are received.

contact graph representation.

The *User-focus* view (Figure 2.4) provides an overview of the user's social neighborhood. It shows the set of participants with whom the selected user has ongoing contact, or had significant cumulative contact so far, or is connected by means of a social link in one of the available online social networks gathered by the metadata collection layer. Contacts and social connections between the user and their neighbors are shown as edges. Contacts and social connections between the neighbors themselves are shown as edges as well. Social connections are decorated with icons representing the social networks from which it was retrieved.

The model is regularly updated based on the live data feed, and the view is updated after each iteration of the algorithm, up to 25 times per second. The result is a continuously morphing network representation in which the marks of beacons that are in contact

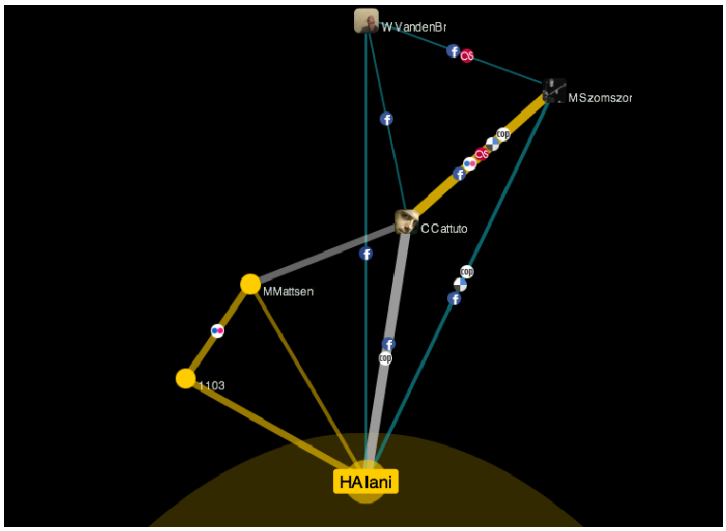


Figure 2.4: User-focus visualization in which user HAlani has the focus. He has ongoing contacts with MMattsen and an anonymous user with badge id 1103, as indicated by the yellow edges. These two users are also in contact, and they are connected in the Flickr network as indicated by the yellow edge and the Flickr icon that decorates it. There has been significant contact between HAlani and CCattuto, as indicated by the thick gray line. They are also Facebook friends and share a COP. The cyan colored edges indicate that the users are (only) linked in one or more of the social or COP networks.

try to occupy adjacent positions, and to move towards the marks of the closest stations. Sample movies can be viewed on the website of the SocioPatterns project[4].

[4] `http://www.sociopatterns.org`

Chapter 3

Human Proximity as Complex and Dynamical networks

In this chapter, we present the scientific tools used to model, analyze and characterize data presented in the previous chapter. Based on the multidisciplinary science of *complex systems* and *complex networks*, we show that *human proximity* data can be conveniently modeled as *dynamical networks* and that most of the concepts used in the analysis of static complex networks, which were later extended to the domain of dynamical networks, can be used to characterize this data. Tools focused on analysis and visualization are important instruments to understand the dynamics behind such data.

Many phenomena in nature can be modeled as a network, and human proximity is one of them. Many complex structures have been studied as *Complex Networks* [6]: brain connections [38], protein-protein interactions [56], social interactions, the Internet and the World Wide Web. All such systems can be represented in terms of nodes and edges as *graphs*. In the World Wide Web, for example, web pages are represented as *nodes* and the links between pages are represented as directed or undirected *edges*. A fundamental characteristic that is long known is that some of these complex networks exhibit *scale-free* structure. Moreover, they display substantial nontrivial topological features, with patterns of connections that are not random, but have a more structured architecture. The modeling and the characterization of these complex systems are challenges presented in complex network research.

Although the complex structure of these systems are often modeled as graphs, their understanding requires most often to consider the *dynamics* of their evolution. Some of their characteristics are relevant only if taking temporal attributes in consideration. *Dynamical networks* differ from traditional complex networks in their dynamism: they are evolving networks, and their analysis must take the time domain into account. Time intervals are natural attributes of nodes and edges, defining for how much time they are active in the network. Node and edge properties also can have temporal attributes that

can change over time. Given their potential as a theoretical model and their promising applications, dynamical networks have been the subject of increasing interest.

Significant progress has been achieved in the last decade or so in the study and characterization of complex networks, focused on static configurations, in which the temporal dimension was not considered. In case of dynamical networks more is bound to come, given the increasing availability of massive networked data sets with temporal information. Characterizing properties like duration, frequency, concurrency and causality in the temporal dimension is absolutely necessary to study the dynamics of interaction in dynamical networks.

In fact, *causal effects* are somewhat underestimated in the domain of opportunistic networks, although they are fundamental in the process of information dissemination. For instance, if a device A first contact B then C, information can flow from B to A and then to C, but not from C to B. By taking only the static network in consideration, the set of contacts allow information flowing in both cases. Few large-scale empirical observations of dissemination processes exist to validate any assumption that causal effects are greatly affected by temporal aspects. However, the few cases in which temporal aspects have been considered in more detail, indeed revealed significant consequences [68, 13, 90, 193, 101, 119, 194, 97, 179].

Human proximity networks can be naturally modeled as dynamical networks. Users and devices are represented as nodes that can appear and disappear in the network. Proximity relations or contacts are represented as edges constantly changing the structure of the network. This abstraction is useful to evaluate how information propagates in mobility networks, or to simulate the dynamics of disease spreading by using compartmental models known in epidemiology [110].

3.1 Complex Systems and Network Analysis

People in a variety of disciplines have always attempted to find universal patterns in the world around them. And in doing so, they search for regularities in what is perceivable, and make use of habitual ways of thinking to explain it. These habitual ways of thinking, or conceptual frameworks, are themselves a product of the perception that reflects the observations that have been made.

Computers, like telescopes and microscopes, have opened a whole new world of possible observations. The computational power available to scientists have made it possible to explore the consequences of simple interactions in complex wholes in a way never before possible. And, in a variety of disciplines, it is emerging that these new observations can enable the understanding of phenomena long believed to be too complex for analysis.

Simple things interacting in simple ways can yield surprisingly complex outcomes [85].

The science of *Complex Systems* is a quite general conceptual framework usable in a variety of different disciplines. It studies how parts of a system give rise to the collective behaviors of the system, and how the system interacts with its environment. The field of complex systems focuses on questions about parts, wholes and relationships; questions that are relevant to all traditional disciplines of science.

Even if it is hard to converge on a widely accepted notion of "complexity", a quite general definition has been given by Barrat et al. [17]:

> *"Complex systems consist of a large number of elements capable of interacting with each other and their environment in order to organize in specific emergent structures".*

In fact, most of the natural and artificial environment in which we are wrapped up reveal patterns of interaction at both macroscopic and microscopic scales. In natural ecosystems, predator-prey relationships can result in interactions as complex as the interactions that occur between proteins to carry out a biological function. Technological networks as the connections between routers of the Internet can be so complex as social networks of friendship between people. All these systems are intelligible examples of *networked systems*, and are objects of study for the so called *Network Science* [200, 33].

The science of Complex Networks is a relatively young field of research, and also involves different disciplines. Not surprisingly, each field concerned with complex networks has its own nomenclature. In fact, every system that encapsulates a relation or interaction between its constituent elements admits some abstract representation in terms of a *network*, or, from a mathematical perspective, of a *graph*. The complex network discipline focus on the empirical study and systematic modeling of real-world networks with the goal of mining the information hidden by the complex patterns, understanding the underlying dynamics of interaction and evolution and learn how to improve the effectiveness and efficiency of artificial complex systems.

The large availability of datasets from many different sources and fields in the last years triggered a significant effort in the investigation of all kinds networked environments using the paradigm of the complex systems analysis. Such very interdisciplinary study allowed to discover very soon striking regularities in diverse networks. Several technological, biological or social self-organizing systems [144, 7] have been found to be characterized by the *small world* property [201], that determines a logarithmic growth of the average distance between pairs of nodes with respect to the total number of nodes. Many of these networks have been also found to be *scale-free*, namely characterized by a node connectivity distribution that decays as a power law[14, 41]. The discovery of these and many other invariants has given strength to the area of complex systems analysis as a cross-discipline science useful to detect and exploit ubiquitous hidden patterns in biological life and technological structures.

Since then, the perspective on the data-driven study of graph-based systems has widened considerably and the results of exploratory analysis of big networks [3, 104, 127], together with the increasingly richness of data on human activity, mobility, and interactions [176] opened the way to devise services that can profitably combine several data sources of interrelated complex systems to provide services to people. Besides the numbers of new recommendation and suggestion tools for online social networks, notable examples are the control of spreading of viruses in computer or human contact networks [152, 55], the monitoring of face-to-face interactions inside hospitals to minimize infections within the structure [102], the "outdoor advertising" on mobile phones [165] based on the mobility patterns of people in a city, and the spreading of information between mobile devices based on spontaneous affinities of users - which is the subject of this work.

In this part we use complex systems analysis to study social, concept, and similarity networks. In the following we provide some fundamental notions of graph theory and the descriptions of network analysis techniques and tools that are at the basis of our investigations.

3.1.1 Graph theory for network analysis

Not by chance, the natural framework for a rigorous mathematical description of networks is found in *Graph Theory*. The general theoretical framework grounded on graph theory can be applied to model a vast variety of complex systems.

The study of graphs, known as graph theory, is a branch of mathematics, and was first systematically investigated by D. König in the 1930s [80]. However, the work written by Leonhard Euler in 1736 on the Seven Bridges of Königsberg is regarded as the first paper in the history of graph theory. His work solves the Königsberg bridge problem, a famous problem precursor to graph theory, and proves the nonexistence of a so-called Eulerian cycle across all seven bridges of Königsberg.

The main sources for the *Graph Theory* formalizations the books by Chartrand & Lesniak [51] and Bollobas [31]. In the most used formalization, an *undirected graph* G is defined by a pair of sets $G = (V, E)$, where V is a set of *vertices* or *nodes*, and E is a set of *unordered* pairs of different nodes, called *edges* or *links*. By referring to a node by its order i in the set V, an edge (i, j) joins the nodes i and j, which are said to be *adjacent*, *connected*, or *neighbors*.

The formalization of *directed graphs* differs on the edge definition. In a directed graph G, V is a set of *vertices* or *nodes*, and E is a set of *ordered* pairs of different nodes that are called directed *edges*. If the graph is directed, the edges can be also called *arcs*, and the node from which the arc originates is called *source* while the other one is called *target*.

A graph is visually represented as a collection of points or circles, and lines connecting

them. The points represent the graph vertices and the edges are represented as lines connecting the vertices. When representing directed graphs, the ordered nature of the edges is usually depicted by means of an arrow, indicating the direction of the edge.

Following the concept of graphs, a *tree graph* is a hierarchical graph where any two vertices are connected by exactly one simple path. In other words, any connected graph without closed loops (cycles) is a tree. A *forest* is a disjoint union of trees. If the tree is directed and there is a parent node from which the whole structure arises, then it is known as the *rooted tree*.

From a mathematical point of view, it is convenient to define a graph by means of a *adjacency matrix* $V \times V$, such that the entry $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ if $(i, j) \notin E$. For undirected graphs the adjacency matrix is symmetric, and therefore it conveys redundant information. For directed graphs, the adjacency matrix is not necessarily symmetric. The relationship between a graph and the eigenvalues and eigenvectors of its adjacency matrix is studied in *spectral graph theory*.

The pair of nodes in an edge, be the graph directed or undirected, are not necessarily distinct. If the nodes are not distinct, i.e., the edge originates and terminates in the same node, then the edge is a *loop*. And if the edges are not unique, i.e., there are edges with the same source and target, then the graph is a *multigraph*. If the graph contains no loops and no multiple edges then it is a *simple* graph.

Nodes and edges can be annotated with attributes, and specifically, a *weighted* graph has numeric weights associated to edges. In this case, the graph can be represented as a weighted adjacency matrix W. Like the adjacency matrix, the weighted adjacency matrix can be used to represent undirected weighted graphs and directed weighted graphs. The weighted graph representation provides a richer description by considering the topology along with quantitative information. This representation is in correspondence with many real networks, which display large heterogeneity in the capacity and intensity values of edges.

3.1.2 Topological measures on graphs

Complex network analysis is based on a core of topological measures that describe the main structural properties of the graph [199, 148]. In the following we review some of them.

Node Degree. In undirected graphs, the *degree* k of a node is definded as the number of edges connected to it. In case of directed graphs, we can distinguish the *in-degree* k_{in} and *out-degree* k_{out}, respectively the number of edges for which the node is a successor and predecessor. More specifically, the in-degree of a node is defined as the number of edges pointing to it, and its out-degree is defined as the number of edges departing from it. The total degree of a node in directed graphs is defined by the sum of its in-degree

and its out-degree.

Node Strength. In weighted graphs, the *strength* of the node is the sum of the weights of the edges incident on a node; in-strength and out-strength are defined for weighted directed graphs, similarly to in- and out-degree.

Nearest Neighbors. The *nearest neighbors* of a node i are the nodes to which it is connected directly by an edge. The number of nearest neighbors of the node is equal to the node degree.

Path. A *path* that connects two nodes v_0 and v_n is an ordered collection of nodes $P = \{v_0, v_1, ..., v_n\}$ and a collection of edges $EP = \{(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)\}$ such that, for each node $v_i \in P - \{v_n\}$, there is an edge (v_i, v_{i+1}) in EP.

Cycle. A *cycle* is a path in which $v_0 = v_n$ and all nodes and all edges are distinct. Any connected graph without cycles is also a *tree*.

Distance. The distance or shortest path length between two nodes i and j is defined as the length of the shortest path going from nodes i to j. It is also called *geodesic distance* [34].

Eccentricity. The *eccentricity* of a node i is the greatest distance between i and any other vertex j. It can be thought of as how far a node is from the node most distant from it in the graph.

Diameter. The *diameter* of a graph is defined as the maximum distance between any two nodes in the graph, or the maximum eccentricity of any node in the graph. That is, the diameter is the longest of all shortest paths among all possible node pairs in a graph. It states how many edges need to be traversed to interconnect the most distant node pairs.

The diameter is used to assess the width of the network; however, it is very susceptible to outliers, since just a single long shortest path can determine a high diameter. To avoid this problem, the *effective diameter* measure have been proposed, that is the minimum distance d such that it includes the 90% of the node pairs [150].

Radius. The *radius* of a graph is the minimum eccentricity of any node.

Density. To measure how densely the graph is connected, the density of a graph is defined as the ratio between the number of existing edges and the number of maximum possible edges. In a simple graph, the density is specified as:

$$D = \frac{|E|}{|V|(|V| - 1)}. \tag{3.1}$$

In directed graphs, density is multiplied by a factor 2 since the possible number of connections is double than the number in undirected graphs. If the number of edges in a graph is close to the maximum number of possible edges, it is said to be a dense graph. If the graph has only a few edges, it is said to be a sparse graph.

Clustering coefficient. The *clustering* measures the tendency of the neighbors of a node to be connected to each other. It tries to answer the question: "are my friends also friends of each other?" Quantitatively, the *clustering coefficient* indicates the degree to which the neighbors of a particular node are connected to each other. The clustering of a node i is measured by a coefficient $C(i)$ as introduced by Watts and Strogatz [201] for the analysis of small-world networks is computed as the ratio between the actual number of edges between i and its neighbors e_i and the maximum possible value of edges possible between its neighbors which is $k_i(k_i - 1)/2$, thereby giving us:

$$C(i) = \frac{2e_i}{k_i(k_i - 1)} \tag{3.2}$$

This measure is meaningful only for $k_i > 1$. If $k_i = 1$ then we consider $C(i) = 0$. The clustering coefficient of the network, which measures the overall degree of clustering of the graph, is simply defined as the average clustering:

$$\langle C \rangle = \frac{1}{N} \sum_i C(i) \tag{3.3}$$

Centrality and centralization. The role that specific nodes and edges have with respect to the global topology of the graph is one of the main insights to characterize the network. The importance of a node or edge is assessed by centrality measures [77], that may be defined on several structural features of the graph, like its connectivity or its position with respect to the other nodes. The most commonly used centrality measure is the *degree centrality*, that coincides with the degree of the node. The degree of a node is a very basic indicator of its centrality, but it is a local measure that does not take into account the global properties of the network. The *Bonacich power index* not only takes into account the degree of a node but also the degree of the nodes connected to a node. *Closeness centrality* approaches compute the distance of a node to all others. *Reach centrality* computes what portion of all other nodes can be reached from a node in one step, two steps, three steps, and so on. The *eigenvector* approach is an attempt to find the most central node in terms of the global or overall structure of the network. Finally, *betweenness centrality* is a measure that aims to describe a node's position in a network in terms of the flow it is able to control.

Mixing patterns.

Structural and hierarchical ordering of many networked systems is often given by the tendency of individuals to be connected with others that share some feature in common. This pattern has been detected in datasets coming from different fields like human and computer sciences, ecology, and epidemiology and it is known as *assortative mixing*. Simmetrically, a *disassortative mixing* is detected in many other real world networks where individuals are connected more likely with others with different properties [145, 146, 147]. For instance, social webs are assortative with respect to the age of the people

and the network of routers in the Internet is disassortative with respect to the number of connections of each router.

Even if mixing patterns can be defined with respect to any attribute of the vertices, the most studied mixing pattern involves the node degree. This mixing measures the likelihood that nodes have neighbors with similar degree. One of the measures of correlation between degrees of neighbors is given by the Pearson assortativity coefficient [145]:

$$r = \frac{\sum_e j_e k_e/|E| - [\sum_e (j_e + k_e)/(2|E|)]^2}{[\sum_e (j_e^2 + k_e^2)/(2|E|)] - [\sum_e (j_e + k_e)/(2|E|)]^2} \tag{3.4}$$

where j_e and k_e denote the degree of the nodes incident on edge e. Its values can range from -1 (perfectly disassortative network) up to 1 (perfectly assortative network). Nevertheless, Pearson coefficient can be misleading when a non-monotonous behavior of the correlation function is observed; in this case the measure gives more weight to the higher degree classes, which in several cases might not express correctly the variations of the correlation function behavior.

A more practical measure involves the average nearest neighbors degree of a vertex i:

$$k_{nn,i} = \frac{1}{k_i} \sum_{j \in \Gamma(i)} k_j. \tag{3.5}$$

Starting from this quantity for single nodes, we can define an average for all the nodes with the same degree class [151, 192]:

$$k_{nn}(k) = \frac{1}{N_k} \sum_{i/k_i = k} k_{nn,i}, \tag{3.6}$$

where N_k is the number of degree k nodes. In the presence of correlations, $k_{nn}(k)$ identifies two classes of networks. If $k_{nn}(k)$ is an increasing function of k, then highly connected nodes have a high probability of being linked with other high degree nodes, while a decreasing behavior reveals a disassortative mixing.

3.1.3 Generative models

Macro structural properties of complex networks do not arise by chance. These properties allow to classify them in different categories, and to define models to artificially reproduce the same qualitative features. There are some properties and features that are normally reproduced by generative models: the diameter, the clustering coefficient and the degree distribution. In the following, we present three well-known network categories and their corresponding models.

Lattice graphs

Lattices are simple deterministic models where nodes are connected according to regular grid-like patterns. There are conflicting definitions of the lattice graph. The simplest definition of n-dimensional lattices is a graph in which the vertices are placed at the integer coordinate points of the n-dimensional Euclidean space and each vertex connects to vertices which are exactly one unit away from it.

The *lattice ring graph* (or one-dimensional lattice) is originated by logically arranging nodes into a ring and connecting each node with its K closest neighbors (with K even), $K/2$ on each side. That is, if the nodes are labeled $n_0 ... n_{N-1}$, there is an edge (n_i, n_j) if and only if $|i - j| \equiv k \pmod{N}$ for some $|k| \in \left[1, \frac{K}{2}\right]$. The complete graph is a limit case of the lattice ring network, where K is sufficiently high to connect all its nodes.

The description adopted by Brualdi and Ryser [36] defines a lattice graph $L_(m, n)$ as the *line graph* of the *complete bipartite graph* $K_(m, n)$. A line graph (also called edge graph) of a simple graph G is obtained by associating a vertex with each edge of G and connecting two vertices with an edge iff the corresponding edges of G have a vertex in common. This definition unveils a different graph structure from the lattice graphs as defined previously.

Despite of the adopted definition of lattice graph, nodes in lattice networks are densely connected locally and therefore the clustering coefficient is high. On the other hand, the shortest path connecting two generic nodes in the network is long on average and consequently the diameter is high.

Random graphs (Erdös-Rényi)

Several random processes of graph growing were studiend in graph modeling [32], and the final degree distribution depends on the nature of the random process. Erdös and Rényi [70, 71] studied a model of growth for random graphs that consists in a random process of edge creation in which, at each step, two nodes are chosen uniformly at random and an edge is inserted between them. The process at the basis of the Erdös Rényi (ER) graphs is the most known model of growth for random graphs.

The generation of an ER random graph requires two parameters: the number of nodes n and the probability of attachment p. The graph $G(n, p)$ is generated through a memoryless process, where an edge is created between each pair of nodes with probability p. The probability distribution of node degrees in the generated graph is given by a binomial distribution as follows, where p_k is the probability that a hode has degree k:

$$p_k = \binom{n-1}{k} p_k (1-p)^{n-1-k} \tag{3.7}$$

As the value of n increases, the degree distribution tends to a Poisson distribution. In

this case, the range of variability of node degree is relatively small:

$$p_k \approx e^{pk} \frac{pn^k}{k!} \tag{3.8}$$

The connectedness of random graphs, especially infinitely large ones, is described in *Percolation theory*, introduced in the mathematics literature by Broadbent & Hammersley [35]. The graph connectivity question is: for a given value of n and p, what the probability is that $G(n, p)$ is connected? In fact, graph connectivity depends on values of n and p, following a sharp threshold behavior [4]. When $n \cdot p < 1$, it is very likely that the graph will be disconnected, with components of size $O(log(n))$. At the threshold value $n \cdot p = 1$ there is a high probability that the graph will have a connected component of size around $n^{2/3}$. Finally, if $n \cdot p > 1$, there is a high probability that the graph will have a giant connected component containing the vast majority of edges. The giant component of ER graphs has small diameter and low clustering coefficient.

Small world graphs (Watts-Strogatz)

Many real-world networks exhibit *small-world* properties, including short average path lengths and high clustering. These characteristics show that these networks are a midterm between random and regular graphs. Social networks, for example, show highly clustered social communities, but two generic individuals have few degrees of separation. The Watts and Strogatz (WS) model [201] is a random graph generation model that produces graphs with these small-world properties. It does so by interpolating between an ER graph and a regular ring lattice.

The generation of a WS graph requires three parameters: the number of nodes n, the mean degree K (assumed to be an even integer), and a probability β. The model generates the graph by constructing a lattice ring graph with N nodes each connected to K neighbors and, for each edge present in the graph, and rewiring it with probability β. Rewiring is done by replacing an edge's endpoint with another node uniformly chosen from the set of nodes, but avoiding self-loops and link duplication.

The resulting graph has n nodes and $\frac{nK}{2}$ edges. As the probability β increases, the properties of the graph change in a threshold-driven fashion. Approximatively, if $\beta < 10^{-3}$ the graph still preserves the main features of the lattice ring and if $\beta > 10^{-1}$ the shape of the graph becomes similar to a ER configuration. With β in the interval $[10^{-3}, 10^{-1})$ the graph assumes the desired small world properties. In particular, if compared with a random graph with the same size, the resulting graph has a comparable average shortest path length but a much higher clustering.

Scale Free network (Barabási-Albert)

The average path length and the clustering coefficient of many real networks are well reproduced by Watts-Strogatz graphs, but their degree distribution are different from the distributions found in scale-free networks, and therefore a model for this growth process is needed. The mostly widely known generative model for a subset of scale-free networks is Barabási and Albert's *preferential attachment* model [14]. In this model, the "rich get richer" strategy is the base of the process, and edges are created with a probability distribution which is not uniform, but proportional to the current degree of nodes.

The generation of a Barabási-Albert (BA) graph requires two parameters: the number of nodes n and a number of edges k to add in each step. The network begins with an initial random network with m_0 nodes with degree greater than 1 (otherwise the node remains disconnected from the network). A new node is added and connected to k nodes with probability proportional to the number of edges that the existing nodes already have. Formally, the probability p_i of connection with an existing node i is:

$$p_i = \frac{k_i}{\sum_j k_j}. \tag{3.9}$$

where k_i is the degree of node i.

Heavily linked nodes ("hubs") tend to accumulate even more edges, while nodes with only a few links have low probability to be chosen as the destination for a new edge. This "rich gets richer" strategy is similar to the growth process of real networks.

The degree distribution of BS graphs is distributed as a power law of the form $p(k) = c \cdot k^{-\alpha}$, that defines a *scale free* distribution. It has been observed that the typical exponent of the power law degree distribution of many real-world graphs has an exponent α ranging between 2 and 3.

The average path length in the Barabśi-Albert model (BA) grows approximately logarithmically with the network size. The clustering coefficient is higher than a ER graph with comparable size, but is still lower than the clustering in WS, whose clustering coefficient is more similar to the values found in real graphs.

Although both BA and WS models fails in modeling some macro statistical properties of real networks, the former by having low clustering coefficient and the latter by having a distinct node degree distribution, such models give a priceless contribution to the study and modeling of complex networks and provide useful abstractions for the comparison of properties of different systems.

Some derivatives of the Barabási-Albert model were proposed by changing the criterion of network growth. Basically, the probability p_i of connection with an existing node i can be proportional to any node property and not just the node degree. Fortunato et

al. [75], for example, propose a model where the probability is proportional to any prestige measure, be it topological or not. They show that the resulting network is scale-free when the probability distribution of p_i is any power-law function of its rank.

3.2 Temporal Networks

The dynamical evolution of networks can be generally modeled by introducing a time variable t that indicates the network changes in time. Some works focus on general models of network evolution [16], by modeling the growth of weighted networks in which the structural growth is coupled with the edges' weight dynamical evolution. In this case, synthetically generated dynamical networks show complex hierarchical structures characterized by clustering and connectivity correlations.

One of the earliest studies using Dynamical Network Analysis (DNA) [43] was the Sampson's monastery study [202], where Harrison C. White observed and analyzed the evolution of the network community structure in a New England monastery by taking snapshots of the network at different intervals. The study describes social relations among novices who were preparing to join a monastic order. The data is freely available in several formats and is represented as a network structure. The novices are represented as nodes, and the edges are the affect relations among the novices, which were collected at five moments in time, by asking them to indicate whom they liked most and whom they liked least. White divided the novices in different groups based on his observations and analyses.

The structure of social networks can be studied over time in order to find evidence of assortative mixing and temporal clustering of behaviors among linked nodes. This evidence is sometimes used to support claims of peer influence and social contagion, but homophily could also be used to explain such evidence. In fact, a great challenge when studying the dynamics in the structure of social networks is to distinguish peer-to-peer influence from homophily. In the former, assortative mixing and temporal clustering is used as evidence that a node directly influences or cause outcomes in its neighbors [52, 53]. The homophily principle, on the other hand, says that similarity breeds connection [136], and some outcome patterns among neighbors that look like direct contagion in fact have no direct causal influence, but are results of relationships with similar people that share common characteristics. A lot of debate is still ongoing among researchers who advocate both points of view, and also criticism to those who do not took into account both factors [182]. In this context, Aral et al. [9] propose a method to understand the mechanisms that drive contagions in networks. Such methods can be used to know how to propagate or to avoid propagation in domains as diverse as epidemiology, marketing, development economics, and public health.

In opportunistic networks, a *Contact Graph* $G(V, E)$ is usually used to describe a

dynamical network. In a contact graph, V is the set of nodes and E is the set of edges, with each edge being represented by a couple of nodes, the time when the contact started and the contact duration.

Here, we refer to the definition of *Temporal Networks* found in the review written by Holme and Saramäki [92]. In this reivew, temporal networks can be divided into two classes corresponding to two types of representations. In the first representation, there is a set of vertices V interacting with each other at certain times, and the durations of the interactions are not considered. In this case, the network can be represented by a *contact sequence*—a set of C *contacts*, triples (i, j, t) where $i, j \in V$ and t denotes time. Equivalently, one can represent the network as a graph $G(V, E)$ with a set of nodes V, a set of edges E, and, for $e \in E$, a non-empty set of times of contacts $T_e = \{t_1, \ldots, t_n\}$.

In the second class of temporal networks, *interval graphs*, the edges are not active over a set of times but rather over a set of intervals $T_e = \{(t_1, t'_1), \ldots (t_n, t'_n)\}$, where the parentheses indicate the periods of activity—the unprimed times mark the beginning of the interval and the primed quantities mark the end. The static graph with an edge between i and j if and only if there is a contact between i and j is called the *(time) aggregated graph*.

Like for static graphs, it can be useful to define an index function of whether a pair of vertices is connected at a given time. This is the *adjacency index*

$$a(i, j, t) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected at time } t \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Another definition of dynamical networks that is similar to the second class of representations - the *Evolving Graphs* - is described by Ferreira in [73]. Following this definition, an *evolving graph* $G(V, E)$ is a set of nodes V and a set of edges E, for each $e \in E$, a non-empty set of times in which e is present (*edge presence schedule*), and, for each $v \in V$, a non-empty set of times in which v is present (*node presence schedule*). The presence schedule indicates the time intervals in which the edge (or node) is present, and possibly other parameters they take during each interval. The presence schedule of a node n is represented as $P(n)$, and presence schedule of an edge e is represented as $P(e)$.

3.2.1 Measures of Temporal-Topological Structure

Some concepts in network analysis must be revisited when studying properties over time [37]. These include, for instance, the notions of paths, as follows:

Journeys. The time domain is incorporated into the definition of paths, and, in order to not confuse with the previous definition of path, we will call it *journey*. A journey from node v_0 to node v_n is an ordered collection of nodes $P = \{v_0, v_1, ..., v_n\}$, an ordered

collection of edges $\texttt{EP} = \{(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)\}$ and an ordered collection of instants $\texttt{TP} = \{t_0, t_1, ..., t_{n-1}\}$. For each node $v_i \in P - \{v_n\}$, there is an edge $e_i = (v_i, v_{i+1})$ in $\texttt{EP}$. For each edge e_i there is a *transfer instant* t_i in $\texttt{TP}$, which must be in the edge presence schedule $P(e_i)$. Finally, for each pair of edges (e_i, e_{i+1}), the associated instants (t_i, t_{i+1}) are in ascending order such that $t_i \leq t_{i+1}$.

It is important to notice that journeys connect two nodes over time, even in the case the nodes are never connected and there is no period in which there is a path between them. The *hop-count* or *length* of a journey is the number of edges in the journey, or $|\texttt{EP}|$. The *departure time* is the transfer instant for the traversal of the first edge in the journey. The *arrival time* is the transfer instant for the traversal of the last edge in the journey. The *journey time* is the elapsed time between the departure time and the arrival time.

Fastest Journeys. The *fastest journey* from node v_0 to node v_n is the journey taken over all journeys between v_0 to v_n such that the journey time is the minimal.

Foremost Journeys. The *foremost journey* from node v_0 to node v_n is given by the first journey arriving at v_n from v_0. The arrival time in the foremost journey, with v_n as destination, is also the *earliest arrival time* between v_0 and v_n. If there is no journey between v_0 and v_n, then the earliest arrival time is ∞.

Foremost Journey Trees. The *foremost journey tree* with root node v_0 is the collection of foremost journeys starting at node v_0 and with destination all the other nodes in the network that are reachable through a foremost journey.

In Chapter 5 the foremost journey tree is also referred as the *Fastest Route Tree* $\texttt{FRT}(v_0, t_0)$, which is the collection of foremost journeys with departure time occurring after t_0, starting at node v_0, and with destination all the other nodes in the network that are reachable through a foremost journey.

3.3 Proximity Networks

As remarked previously, datasets that represent proximity traces between devices can be abstracted as dynamical networks. Some datasets, like the MIT/Reality Mining dataset, are represented as contacts with start time and end time, and the representation of its contacts as graph edges with intervals of validity is straightforward.

For SocioPatterns datasets, we build the contact graphs in the following way. The raw data stream from the proximity-sensing platform is aggregated to build a time-ordered sequence of graphs. We coarse-grain time over an interval of duration $\Delta t = 20$s, over which the platform can assess proximity (or lack thereof) with a high confidence. For each consecutive time interval (*frame*) of duration Δt, we build a contact graph frame, where nodes represent individuals, and edges represent proximity relations between individuals

that were recorded during the corresponding frame. Within a frame, an edge is considered active from the beginning of the frame to the end of the frame. Edges and nodes appear or disappear at frame boundaries only. Figure 3.1 shows an example of a sequence of contact graph frames.

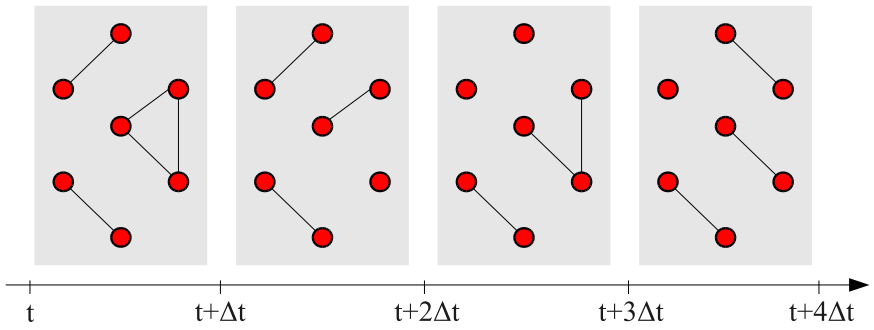
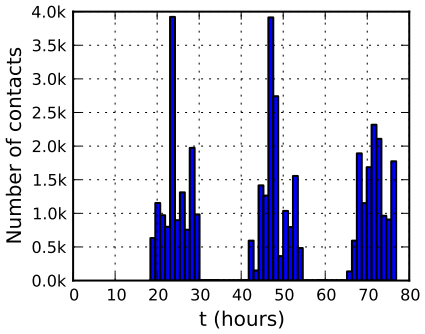
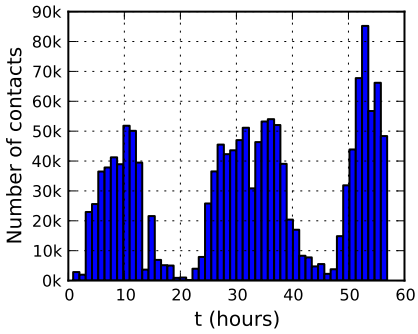


Figure 3.1: An example of a sequence of contact graph frames. Each frame corresponds to a time interval of duration Δt and aggregates all events reported during that interval.

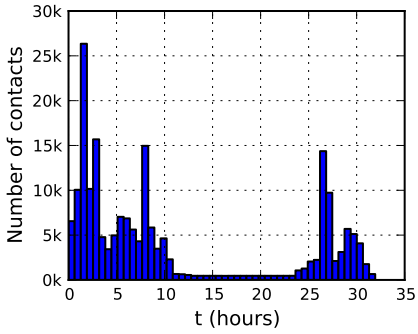
In a real-world deployment, such as a conference one, the number of contacts active in each frame can greatly vary along the deployment timeline. Figure 3.2 shows the number of contacts in the frames for each deployment, as a function of time. During the night, and whenever the social activity is low, the number of contacts is low. Over one day, contact density is highest during social activities like lunch and coffee breaks.



(a) HT2009



(b) 25C3



(c) SFHH

Figure 3.2: Number of contacts during three different SocioPatterns deployments. The bars show the absolute number of contacts occurred in the given time interval.

Chapter 4

Exploring and Visualizing Dynamical Networks

In this chapter we discuss techniques to explore and visualize dynamical networks. The exploration and visualization of temporal network data is extremely important to help hypothesis formation and support the explanation of some phenomena. Since human proximity can be represented as dynamical networks, tools that help such exploration and visualization are essential to understand data in a preliminary phase. We show some formats that can be used to represent such data, and tools that are currently used to explore both statical and dynamical networks. We discuss how to visualize dynamical networks in real-time, and we present a tool that we implemented in order to visualize streams of graph events.

4.1 Preliminaries

The increasing amount of digital information provided by the diffusion of online services, sensors, mobile devices, user-generated content and open data initiatives is leading the Network Science to a golden age. Data shared by online services, social networks, news sources, scientific research, government institutions, intelligent homes, buildings, industries and cities are available for everyone not only to download and use, but to access in real-time through data streams. However, while access to data is crucial, users can find themselves drowned in an information deluge. The direct access to such data without the appropriate instruments becomes useless and even harmful if it interferes in our capacity of understanding it.

While this data-rich scenario is quickly emerging, research on tools and methodologies to represent and explore such wealth of information in real-time is lacking. Methods to explore such type of data in its static form are available since the origins of Social

Network Analysis (SNA) [76], and the capacity to leverage such know-how in systems that constantly change in time is an unavoidable step. Visualizing such temporal data is not an easy task, but computers, like telescopes and microscopes, have opened a whole new world of possible observations, and are providing the computational power necessary to explore from the big picture to the extreme detail. We must start to fill the gaps between the static and the dynamic, in order to give to social network analysts the possibility of doing real social forecasting.

4.2 Visualizing Network Data

Presenting data through information visualization is an effective way to take full advantage of the human perceptual capabilities. Although visual representation of data can be achieved through a myriad of different methods - the creativity seems to have no limit for designers that create infographics - we focus on linked data that can be represented as dynamical networks of connections. When social entities can be represented as nodes and links, the most common way to present data to users is by using graphs. Graphs are probably the most general method to represent such information building blocks in a domain-independent way.

Graph entities are abstractions that can represent a wide range of real-world structures, from computer networks to human interactions, and there are a lot of standards to exchange graph data in different formats, from text-based formats to XML-based formats. However, real-world structures are constantly changing, and it is difficult to find formats that are suitable to exchange such type of dynamic data.

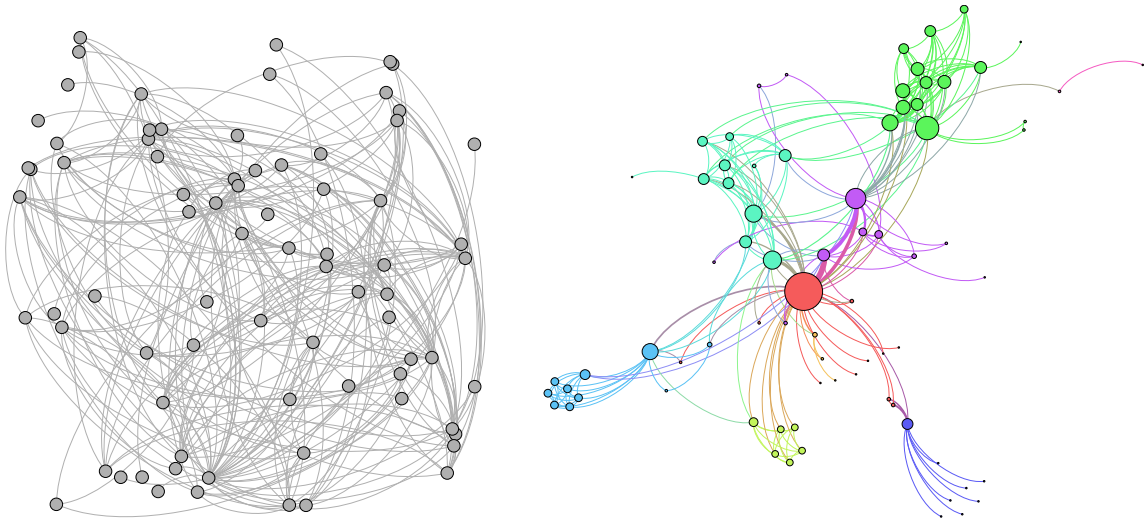
Either visualization, navigation and pattern searching in networks require an exploratory process. Some strategies use attribute ranking and coordinated views to help users systematically examine numerous measures [160]. Such strategies focus on relationships instead of individual elements, which is fundamental when studying how individuals interact and influence each other. A good graphical representation of a network can unveil its most important structural components, logically partition its different regions, and point out central nodes and edges that are responsible for its cohesion - nodes and edges on which information flows more frequently or quickly.

Network exploration tools need to be technically accurate, visually attractive and enable real-time visualizations. To summarize the general process steps of graph visualization, Scheiderman presented the Visual Information Seeking Mantra [185]: “Overview first, zoom and filter, then details-on-demand”. In order to supply the demanded steps for this general process, high quality visualization tools should support high quality layout algorithms, data filtering, clustering, statistics and annotation of graph entities.

Amongst the many research issues for graph visualization, visualizing temporal data is one of the most complex. The proposal of new types of data interfaces is a further

step to transform dynamic data into useful knowledge and insight. Moody et al. [141], recognizing that network visualization fosters theoretical insights, stress the importance of creating dynamic network visualizations, or network "movies". This work opens some technical questions about meaningful ways to link network changes with changes in the graphical representation. The authors divide representations into two types: static flip books, where node position remains constant but edges cumulate over time, and dynamic movies, where nodes change position in response to changes in relations.

In order to represent values for metrics such as degree and centrality, nodes and edges can be represented in different colors, sizes and forms. But one of the main problems in visualizing the complex network structure that arises is how to spatially represent nodes and edges. To this end, it is common to use layout algorithms to exploit different layout techniques, generating different visual representation for the same input graph. The most widely known layout algorithms are grounded in physical models of springs and forces [78], but sometimes this is not sufficient to highlight interesting patterns and trends. Figure 4.1 illustrates the representation of a graph using a force-directed layout that conveys to the observer much information about the structure of the graph.



(a) Random layout

(b) Structured layout

Figure 4.1: Two visualizations of the co-appearance weighted network of characters in the novel Les Miserables, used by D. E. Knuth to explain graphs [113]. (a) shows a representation with a random layout and no colors and (b) is a visualization where nodes are resized according to their degree, colored according to their maximum-modularity cluster and arranged through the Yifan Hu force-directed graph layout [96]

Such graph representations can, for example, unveil information about the dynamics of message spreading over online networks. For example, dynamic representations of information dissemination over the Twitter network can be found on the *Truthy*

Project[1] [166]. Truthy is a web service that tracks political *memes* in Twitter and helps detect *astroturfing*, smear campaigns, and other misinformation in near-real-time. This Web Service is based on an extensible framework that enables the analysis of meme diffusion in social media by mining, visualizing, mapping, classifying, and modeling massive streams of public microblogging events.

4.3 Representational Formats

In order to describe network structures, their associated data and dynamics in a interchangeable way, a graph representation sould provide a way to add a lifetime to nodes, edges and data.

The most commonly used format is the GEXF (Graph Exchange XML Format, [2]). Both network topology and data have attributes representing their lifetime. The whole graph, each node, each edge and their respective data values may have time limits, beginning with an XML-attribute start and ending with end. Attributes declared as dynamic are allowed to exist during a time scope. An example of GEXF representation is shown at Listing 4.3.1.

If the same node or edge exists at different intervals, different time ranges should be provided as XML `<spells>` elements. Listing 4.3.2 show how to use `<spells>` elements with corresponding start and end attributes inside a GEXF definition.

4.4 Tools

Many successful projects developed tools for visualization of large graphs in order to understand complex networks. Tools for complex network analysis normally focus on a single instance of a large network. For example:

- *Pajek* [21] (spider, in Slovene), for example, is one of the first visual exploratory tools for graph visualization and analysis. It is freely available for noncommercial use.
- *Cytoscape* [183] is another tool that is an open source software, mainly used for manipulation of biomolecular interaction networks. Cytoscape support large databases of protein-protein, protein-DNA, and genetic interactions, available for humans and model organisms. It is also an extensible software, with a plug-in architecture, allowing the development of additional features.

[1] `http://truthy.indiana.edu/`
[2] `http://gexf.net/`

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.gexf.net/1.2draft
  http://www.gexf.net/1.2draft/gexf.xsd" version="1.2">
    <meta lastmodifieddate="2009-03-20">
        <creator>Gexf.net</creator>
        <description>A Web network changing over time</description>
    </meta>
    <graph mode="dynamic" defaultedgetype="directed" timeformat="date">
        <attributes class="node" mode="static">
            <attribute id="0" title="url" type="string"/>
        </attributes>
        <attributes class="node" mode="dynamic">
            <attribute id="2" title="indegree" type="float"/>
        </attributes>
        <nodes>
            <node id="0" label="Gephi" start="2009-03-01">
                <attvalues>
                    <attvalue for="0" value="http://gephi.org"/>
                    <attvalue for="2" value="1" start="2009-03-01" end="2009-03-10"/>
                </attvalues>
            </node>
            <node id="1" label="Network">
                <attvalues>
                    <attvalue for="2" value="1" start="2009-03-01" end="2009-03-10"/>
                </attvalues>
            </node>
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" start="2009-03-01" end="2009-03-10"/>
        </edges>
    </graph>
</gexf>
```

Listing 4.3.1: GEXF representation of a simple dynamical network

```
<gexf ...>
    ...
    <graph mode="dynamic" timeformat="date">
        <nodes>
            <node id="0">
                <spells>
                    <spell start="2009-01-01" end="2009-01-15" />
                    <spell start="2009-01-30" end="2009-02-01" />
                </spells>
            </node>
        </nodes>
    </graph>
</gexf>
```

Listing 4.3.2: Extract of a GEXF graph definition showing the representation of a dynamical network with `<spells>` elements

- *Networkbench*[3] is a tool to model and visualize networked datasets from different fields, in support of cross-disciplinary research.

[3] `nwb.cns.iu.edu`

- *Walrus*[4] [142] is a tool that visualizes graphs using a 3D engine, based on their spanning tree representation. Thus, in practice, Walrus is best suited to visualizing graphs that are nearly trees.
- *GUESS*[5] [1] is a visualization and analysis tool based on Gython, a domain-specific language that supports operators that can deal directly with graph structures in an efficient and intuitive way.
- *GleamViz*[6] is specifically designed to simulate and visualize spreading of infectious diseases across the globe.

Furthermore, many graph analysis packages like *iGraph*[7], *networkx*[8], and *R*[9] provide network visualization tools or plug-ins.

Tools for analysis and visualization of dynamical networks, in contrast, are essentially statistical tools that admit the analysis of multiple networks or multiple states of the same network simultaneously. Interval-based filters are the basic ingredient for selecting states of a dynamical network in different stages of evolution.

Amongst the many research issues for graph visualization, visualizing temporal data is one of the most complex. In [141], for example, dynamical networks are called as longitudinal networks; and by recognizing that network visualization fosters theoretical insights, they state about the importance of creating dynamic network visualizations, or network "movies". This work also open some technical questions about meaningful ways to link network changes with changes in the graphical representation. The authors divide representations in two types: static flip books, where node position remains constant but edges cumulate over time, and dynamic movies, where nodes change position in function of changes in relations.

Following the increasing interest on dynamical graphs, *Gephi*[10] [20] emerged as a framework for graph analysis, manipulation and visualization which includes tools to study not only static complex networks but also large dynamical networks. Gephi is an open source software based on the Netbeans platform, specialized in graph analysis and visualization. For visualization of large networks, it uses OpenGL through Java™ JOGL, as shown in Figure 4.2, that speeds up the exploration and realtime rendering. The key features allow spatializing, filtering, navigating, manipulating and clustering graphs.

In the Gephi framework, the *Timeline* component is a simple interface that allows users to select pertinent time intervals and display and explore the corresponding graph.

[4] www.caida.org/tools/visualization/walrus/
[5] graphexploration.cond.org
[6] www.gleamviz.org
[7] igraph.sourceforge.net
[8] networkx.lanl.gov
[9] www.r-project.org
[10] http://gephi.org

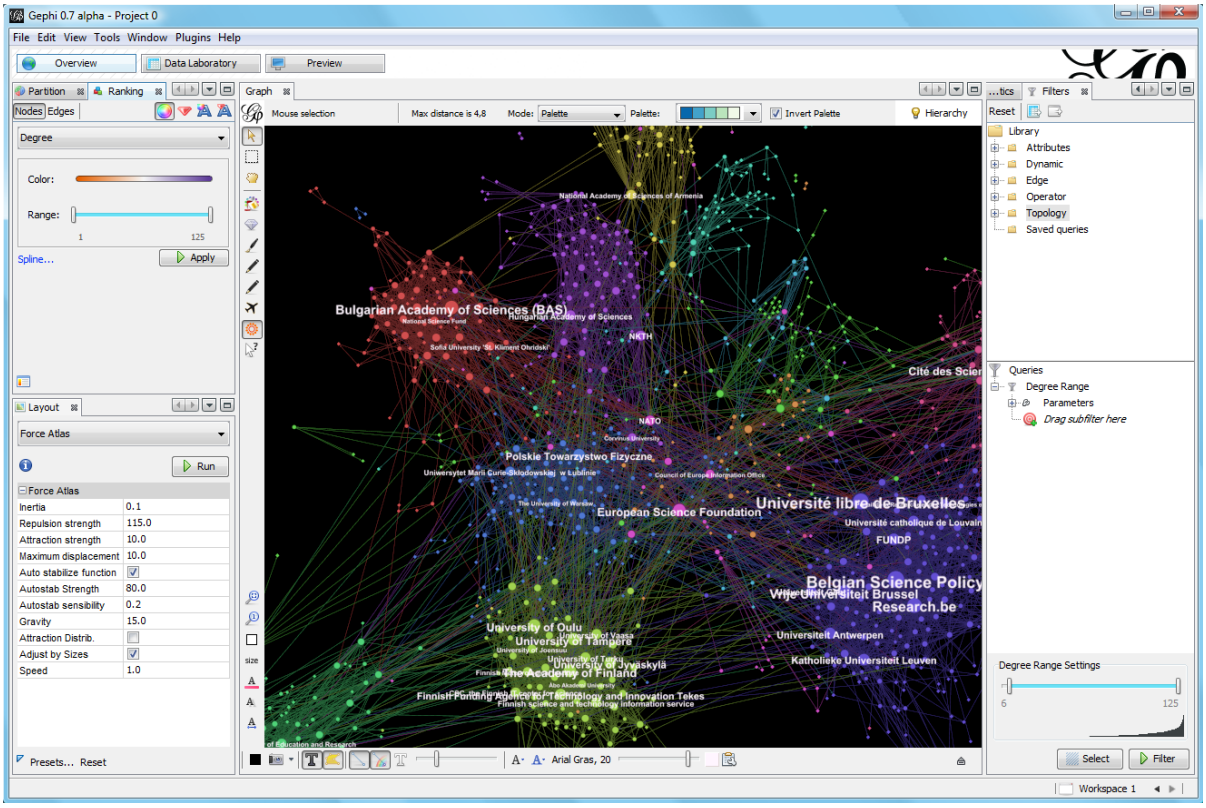


Figure 4.2: Screenshot of Gephi showing a large graph rendered with OpenGL.

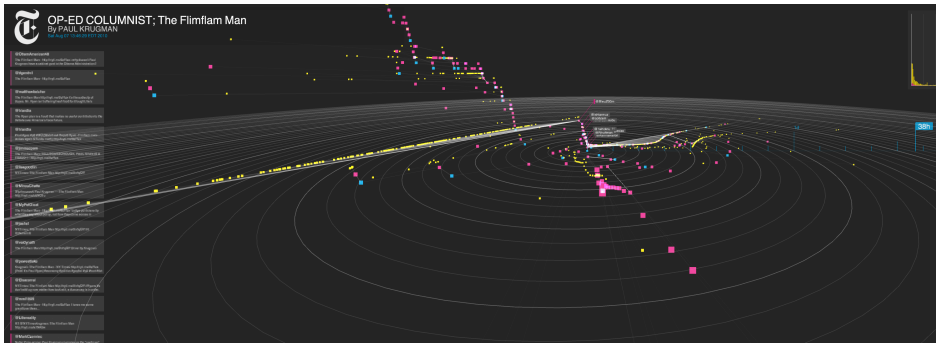


Figure 4.3: Screenshot of a three-dimensional dynamic network visualization using Cascade, a project developed in The New York Times' Research and Development Lab. The dots represent users' status updates in Twitter. The spatial representation obeys the temporal ordering of statuses, where the distance from the center express the temporal dimension of the process.

Gephi supports the GEXF format to represent the underlying dynamical networks.

Cascade[11] is a project developed in the The New York Times' Research and Development Lab using open source tools. This project focus just on Twitter data to be processed, and try to show the entire chain of reactions that results from an isolated social-media event. It tries to be dynamic and exceedingly detailed in comparison with most Twitter visualizations. The project is the result of a collaboration among UCLA professor Mark Hansen, who spent a spring 2010 sabbatical working at the Times, along with Jer Thorp, official data artist in residence, and Jake Porway, the staff data scientist. A screenshot of a three-dimensional dynamic network visualization using Cascades is shown in Figure 4.3

4.5 Graph Streaming

A lot of well-established web systems and services offer streamed data to its users using a streaming API. Twitter[12], for example, defined a Streaming API to allow real-time access to its data. They are using two different formats: XML and JSON, but JSON is strongly encouraged over XML, as JSON is more compact and parsing is greatly simplified for streaming. The Digg Streaming API[13], that also uses the JSON format to retrieve a full steam of submissions, comments and *Diggs*, is another example of how social applications tend to be more and more real-time oriented.

[11] `http://nytlabs.com/projects/cascade.html`
[12] `http://www.twitter.com/`
[13] `http://digg.com/`

An interesting proposal to stream graph entities into an application is the *GraphStream Java™ Library* [65]. It is composed of an API that gives a way to add edges and nodes in a graph and make it evolve. Nodes and edges that can appear, disappear or be modified through a stream of event-driven operations. While GraphStream can help researchers and developers to analyze dynamic graphs in their daily tasks of dynamic problem modeling and of classical graph management, its initial focus was not the real-time processing of data streams.

In the following, we describe an application independent event-based JSON format for graph exchange, where clients can interact with a server by retrieving and pushing graph data to it, in a REST architecture. It is composed of three different types of operations - (a)dd, (c)hange, (d)elete - performed in two types of entities - (n)odes and (e)dges - for a total of six types of events: `an`, `cn`, `dn`, `ae`, `ce`, `de`:

Event Type	Description	Required Attributes
an	Add node	Node ID
cn	Change node	Node ID
dn	Delete node	Node ID
ae	Add edge	Edge ID Source ID Target ID
ce	Change edge	Edge ID
de	Delete edge	Edge ID

```
json_object ::= '{' events '}'
events      ::= event | event ',' events
event       ::= event_type ':{' entity_defs '}'
event_type  ::= 'an'|'cn'|'dn'|'ae'|'ce'|'de'
entity_defs ::= entity_def |
    entity_def ',' entity_defs
entity_def  ::= entity_id ': {' attributes '}'
attributes  ::= attribute |
    attribute  ',' attributes | ''
attribute   ::= attribute_name ':' attribute_value
```

Listing 4.5.1: Simplified grammar definition for the Graph Streaming JSON format. In this definition, `entity_id`, `attribute_name` and `attribute_value` are literals representing entity (node or edge) identifiers, attribute names and attribute values respectively.

Each *event* is composed by an event type and a list of graph entities (nodes or edges, depending on the event type). Node and edge entities are similar, and composed of an identifier and a list of attributes. *Add Edge* (`ae`) is the only operation in which there are two mandatory attributes: source and target, the node identifiers representing the

source and target of the edge. The events are represented in JSON format according to the simplified grammar definition at Listing 4.5.1. The Listing 4.5.2 shows a list of events represented in JSON format, with some examples for each type of event.

```
{"an":{"A":{"label":"Node A","size":2}}} // add node A
{"an":{"B":{"label":"Node B","size":1}}} // add node B
{"an":{"C":{"label":"Node C","size":1}}} // add node C

// add edge A->B
{"ae":{"AB":{"source":"A","target":"B","weight":2}}}
// add edge B->C
{"ae":{"BC":{"source":"B","target":"C","weight":1}}}
// add edge C->A
{"ae":{"CA":{"source":"C","target":"A","weight":2}}}
// changes the size attribute to 2
{"cn":{"C":{"size":2}}}
// removes the label attribute
{"cn":{"B":{"label":null}}}
// add the label attribute
{"ce":{"AB":{"label":"From A to B"}}}

{"de":{"BC":{}}} // delete edge BC
{"de":{"CA":{}}} // delete edge CA
{"dn":{"C":{}}}  // delete node C
```

Listing 4.5.2: A list of events with some examples for each type of event, represented in JSON format

Each `json_object` must be separated by a carriage return control character (‘`\r`’) in order to allow the client to process it.

With this format it is possible to put more than one object in each event, as in the Listing 4.5.3. Although this format is very concise and allows for grouping of graph entities into one event and for grouping of events into one JSON object, it is recommended to send events immediately when they are produced, as this is more suitable for a streaming approach. Data should be read as soon as possible by the client, and the approach using multiple objects by event slows down the client reading, because it can’t parse the JSON event object until a ‘`\r`’ appears.

```
{"an":{
    "A":{"label":"Streaming Node A","size":2},
    "B":{"label":"Streaming Node B","size":1},
    "C":{"label":"Streaming Node C","size":1}
  }
}
```

Listing 4.5.3: A list of events with more than one object in each event, represented in JSON format

4.5.1 Coupling Tools with Graph Streams

Following the increasing interest in dynamical graphs, *Gephi*[14] [20] emerged as a framework for graph analysis, manipulation and visualization which includes tools to study not only static complex networks but also large dynamical networks. Gephi is an open source software based on the NetBeans platform, specialized in graph analysis and visualization. For visualization of large networks, it uses a Java™ 3D render engine, that speeds up the exploration and real-time rendering. The key features allow spatializing, filtering, navigating, manipulating and clustering graphs.

In order to build a unified framework for streaming graph entities into Gephi, we proposed the Graph Streaming plug-in[15]. The plug-in was developed in the scope of this work, as a project within the Google Summer of Code program in May-August 2010. The main purpose of this plug-in is to build a unified framework for streaming graph objects.

Gephi's data structure and visualization engine has been built with the idea that a graph is not static and might change continuously. By connecting Gephi with external data-sources, it is possible to leverage its power to visualize and monitor complex systems or enterprise data in real-time. Moreover, the idea of streaming graph data goes beyond Gephi, and a unified and standardized Application Programming Interface (API) could bring interoperability with other available tools for graph and network analysis, as they could start to interoperate with other tools in a distributed and cooperative fashion.

The Gephi Graph Streaming plug-in allows to stream in real-time the changes that a graph is undergoing, allowing an application to receive these changes and process them as they happen. Twitter (`http://twitter.com`), for example, has a Streaming API interface, through which it is possible to access a sample of the real-time flow of *tweets*. An application can connect to the Twitter server, query for a keyword and from that moment onward it will receive the new tweets that contain that keyword. Similarly, Gephi can connect to an external graph stream and get the data to construct and visualize a graph, in real-time.

The plug-in was implemented in such a way that Gephi can work both as server and as client. In client mode, Gephi connects to an external graph stream and gets data to construct and visualize the evolution of a graph. In server mode, an interface is provided to connect to Gephi and get generated graph data as a stream.

[14] `http://gephi.org/`
[15] `http://gephi.org/plugins/graph-streaming/`

4.5.2 Real-Time Visualizations with Graph Streaming

We used the Gephi Graph Streaming plug-in to collect data from Twitter in real-time and to represent it as a dynamic network. By filtering all Twitter messages (*tweets*) with a given token (*hashtag*) and taking only *retweets* (messages originated from another user than the user who posted the message), we created a real-time dynamic visualization of online information flowing during an important event in the social media, the Egypt uprising. The event was the resignation of Egypt's ex-president Hosni Mubarak, and we collected data starting slightly before the announcement during one hour.

Figure 4.4 shows a static representation of about one hour of data collected from Twitter, during the announcement of the resignation of Egypt's ex-president. In this representation, which we call the *#jan25 Network*, nodes are Twitter users, and links appear between the nodes A and B when user B retweeted a message from user A containing the hashtag #jan25. The dynamic representation can be viewed in a video[16] with the flow of retweets, and Figure 4.5 shows three different snapshots of this representation.

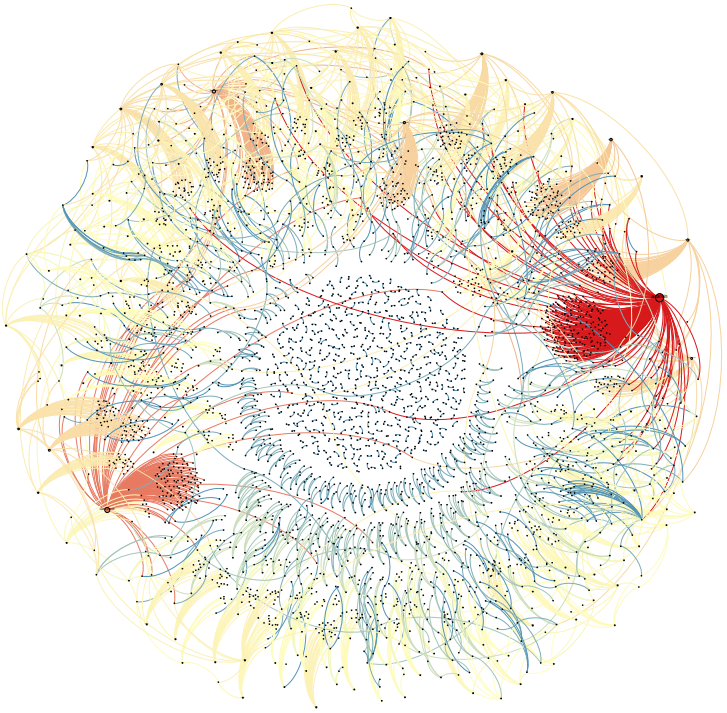


Figure 4.4: Static representation of data collected from Twitter, filtered by hashtag #jan25 at February 11, 2011, during the announcement of the resignation of Egypt's ex-president Hosni Mubarak. Nodes are Twitter users, and links appear between the nodes A and B when user B retweeted a message from user A containing the hashtag #jan25.

[16] Available at `http://www.youtube.com/watch?v=2guKJfvq4uI`

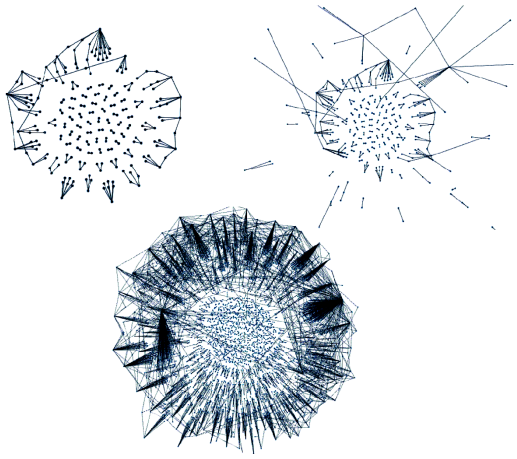


Figure 4.5: Snapshots of three different stages in the evolution of the dynamic network with data collected from Twitter.

This dynamic network visualization can unveil a lot of information about the underlying network structure, even using only simple representation techniques. Some clues about what is happening during the process of information spreading can be seen in real-time, while data is being received. We can see for example that the nodes in the center are pairs of users in which one user retweeted the other, but they have no other connection to the other users. On the other hand, there are other users who show an important role in the information spreading process, with lots of other users retweeting from them. It is possible to visually identify the long-tailed distribution of node degrees, with lots of nodes with few connections, and few nodes with a large number of connections. We can also identify the exact moment of the announcement, when a burst in the activity is perceived.

One of the problems in visualizing the complex network structure that arises is how to represent it on the screen. To this end, it is customary to use layout algorithms, that take a graph as an input and build a two-dimensional representation of it. There are many different layout techniques, that generate very different visual representation for the same input graph. In order to spatially represent the nodes, we used a force-directed layout algorithm called "Force Atlas", which is available in Gephi. The specific advantage of this layout is that it can be executed continuously over an incoming stream of data, grouping together clusters of nodes. This makes it suitable to visualize real-time streams of graph events, but other force-directed layouts should be appropriate as well.

Information spreading is one of the several aspects of Twitter that have been extensively investigated in the literature [84]. The *retweet network* unveils information about user credibility and their potential as indicators of the state of mind of a population. With the representation produced by the Graph Streaming plug-in, some clues about the process of information spreading over the Twitter network can be seen in real-time. It

is possible to identify the long-tailed distribution of node degrees and bursts in the user activity. The nodes with few connections concentrated in the center of the visualization are pairs of users in which one user retweeted the other, but have no additional connections with other users. Some users, however, show an important role in the information spreading process, with a large number of connections, meaning lots of users retweeting from them.

It is important to note that the collected data represent approximately only 10% of all tweets with the given hashtag. In fact, data sampling tends to break spreading cascades into smaller cascades, so the spreading pattern could be different when analyzing all data. In this case, it would be expected that there would be more connections between nodes in the graph (i.e., less missed retweets) and a different layout may be more effective. But most of the features as activity bursts and important nodes can be identified even with data sampling.

In order to collect data from Twitter, we developed a Python server that connects to the Twitter Streaming API, filter the data and serve it in JSON format. The source code for the server, along with other Python examples on how to use the Gephi Graph Sreaming plug-in, are freely available online[17].

4.5.3 Other Real-time Visualizations

Other projects are available to analyze the dynamics of message spreading over online networks. For example, dynamic representations of information dissemination over the Twitter network can be found on the *Truthy Project*[18] [166]. Truthy is a web service that tracks political *memes* in Twitter and helps detect *astroturfing*, smear campaigns, and other misinformation. This web service is based on an extensible framework that enables the real-time analysis of meme diffusion in social media by mining, visualizing, mapping, classifying, and modeling massive streams of public microblogging events.

There are also examples of information spreading visualizations in other contexts outside Twitter. For example, in Google's recently launched social network service Google+. In October 28 2011, Google launched a service in which it is possible to visualize the diffusion progress of posts published in Google+ by using *Google+ Ripples*.

Ripples is a visualization that charts the chain reaction that occurs when a post is shared on Google+. It follows the trajectory of a public post as it is shared from person to person. Arrows indicate a post's progression, while circles represents users who shared it. Larger circles indicate shared posts that influenced more people to repost, and zooming into these circles reveals the sequence of shares. An animation also allows to watch the content spread from the initial post until the present moment. A graph at the bottom of

[17] `https://github.com/panisson/pygephi_graphstreaming`
[18] `http://truthy.indiana.edu/`

the page charts the frequency of shares over time. This visualization shows the spreading evolution over time and the names the major diffusers of the post. This is a tool still being tested, but with which anyone can see the evolution from the perspective of "visual analytics". Figure 4.6 shows a Google+ Ripples diagram, in which it is possible to analyze how a post spreads as users share it on Google+.

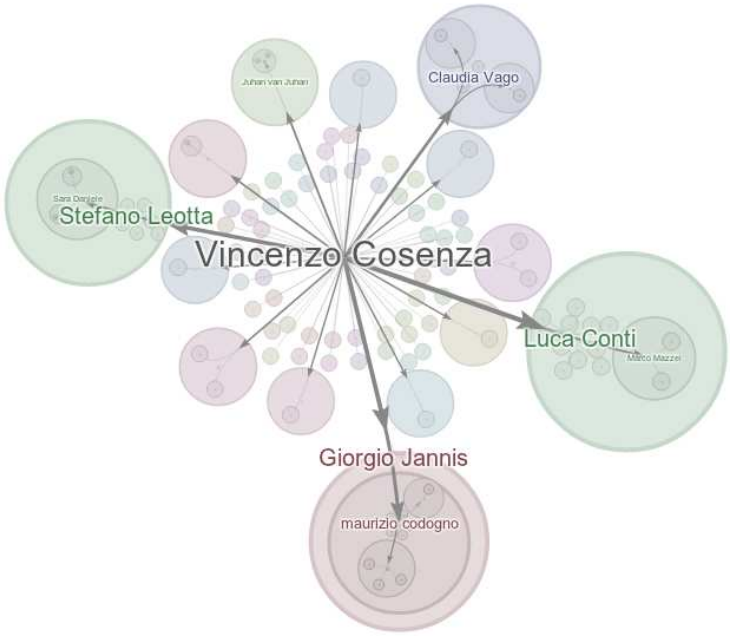


Figure 4.6: Google+ Ripples diagram, that shows a post spreading as users share it on Google+. Arrows indicate the direction of resharing. Circles within circles represent a resharing sequence, so large circles indicate heavy resharing.

4.6 Conclusion and Future Work

In this chapter, we discussed techniques to explore and visualize dynamical networks. We presented a generalized event-driven framework for graph streaming, and a set of tools to visualize streams of network data in a wide range of domains. The potential of such combination of tools for graph analysis with the possibility of streaming graph data in real-time is vast. The extensibility of Gephi enables the development of other plug-ins that leverage the capacities of the already existing ones, by creating a rich ecosystem of vital importance to network scientists.

However, the use of such framework is not limited to its application in Gephi, and we hope that a unified and standardized framework could bring interoperability with other available tools for graph and network analysis and visualization. In order to experiment in this direction, we have been testing the use of this framework in the context of the Data Interfaces laboratory[19], where it has been used to connect streaming data sources as diverse as social streams, graph databases or network traffic to visualizations built with technologies such as ActionScript, JavaScript or Java™.

The projects, aimed at exploring the expressive and communication possibilities of streaming data networks, provided a test bed for the framework, verifying its adaptability to a variety of cases, and providing a flexible communication middleware between the data sources, the data analysis layer, and the visualization engines.

[19] `http://www.datainterfaces.org/`

Chapter 5

Impact of User Mobility in Opportunistic Data Dissemination

In this chapter, we use some of the scientific tools presented in the previous chapter to analyze and characterize data presented in Chapter 2. We propose a novel type of analysis to understand and statistically quantify the process of message spreading in real world contact networks. This analysis takes into account user behavior heterogeneity, as it is visible and quantifiable in real world collected data. The proposed analysis process show results that are universal across different experiments, and are independent of the distribution of the contacts during time. Finally, we will show that some of the synthetic models widely used to generate contact data exhibit characteristics that differ from real world data.

The understanding of dynamics of human interactions is a difficult task and it is not yet explored in great depth. The analysis of epidemic models by computer simulation have been useful in different knowledge areas, from communication and networks by evaluating opportunistic and delay-tolerant protocols, to health care by understanding the dynamics of disease spread and how epidemic processes occurs. However, despite of the large number of models and datasets available and listed in the previous chapter, little analysis is reported that try to identify causal effects in the process of message diffusion based on these models and datasets.

Most frameworks concentrate in the analysis of message delay and distribution of inter-meeting times. But the study of the inter-meeting times distribution is not sufficient in a dynamic process as a message forwarding protocol. In the area of Complex Networks, it is long known that a social networks have scale-free distribution of node contacts. This means there are nodes that have a very limited number of contacts during their life cycle, while other nodes have a much larger number of contacts, and this distribution obeys a power law. If the networks have scale-free distribution of node contacts, as is the case in social and complex networks, then we need to find a way to identify the causal effects

taking place in such networks.

In order to better understand the constraints of opportunistic networking, evaluating the distribution of contacts is of great importance. The metrics used to evaluate these characteristics are the distribution of the contact time between two devices and the inter-contact time, i. e., the time gap separating two contacts between the same pair of devices [49, 40, 107]. The contact time (CT) of any contact is defined as the time difference between the starting of the contact and the end of the contact. The inter-contact time (ICT) between two nodes n_i and n_j is defined as the time difference between the end of a contact between n_i and n_j and the starting of the following contact between n_i and n_j. These metrics are used to evaluate protocols using both experimental data [49] and synthetic data [40].

Chaintreau et al [48] where perhaps the first to report empirical evidence suggesting that the ICT complementary cumulative distribution function (CCDF) follows a power-law. Based on these findings, they derived some hypothesis on the viability and performance of opportunistic algorithms. In particular, their hypothesis imply that, for any forwarding scheme, the mean packet delay is infinite, if the power-law exponent of the ICT CCDF is smaller than or equal to 1. These results are in sharp contrast with previously known results obtained under a hypothesis of exponentially decaying ICT CCDF [87]. Furthermore, as exponential decay is implied by most mobility models, the authors suggested a need for new models to support power-law distributions.

Understanding interaction patterns among individuals is a key point to the study of information dissemination in mobile and ad hoc environments. The current approaches that analyses human trajectories through phone mobility have some limitations, since they prevent us to understand contact phenomena in a short range. Another point that lacks of study in human mobility is the study of collective dynamics as human agglomerations. In this chapter we also analyze data gathered by the experiments in the SocioPatterns project. The analysis of the data set created by the short range human mobility information can shed light on hidden patterns in social mobility dynamics and improve the performance of information dissemination algorithms.

The collection of contact traces and node proximity is usually sufficient for the study of epidemic models or opportunistic and delay-tolerant protocols. Additional information, as node localization and path tracing, can be useful to identify risk areas, but it is not required to study the dynamics of the information/viral spreading.

For this purpose, some datasets are available; for example, the Haggle-Unitrans Mobility Trace [124], which includes several traces about the available Bluetooth connectivity during a typical day on the Unitrans bus system at University of California, and the Reality Mining project dataset [98], which includes information about communication, proximity, location, and activity from 100 subjects at MIT over the course of two academic years. Some of these datasets include more than only phone mobility information, but also information about proximity and short-range connectivity. But the works about

human mobility currently focus on observation and interpretation of results, and most of them with phone mobility data.

To overcome the limitations of mobile phone datasets, we aim to analyze data gathered by the experiments in the SocioPatterns platform, already presented in Chapter 2. SocioPatterns (`www.sociopatterns.org`) [47, 191, 5] is an experimental framework aimed to gather data on face-to-face social interactions between individuals. In these experiments, a group of volunteers wear small tags with integrated active RFID technology that broadcast small data packets to a number of stations and are relayed through a local network to a server for further processing. We are then able to locate, with a fine-grained granularity, reciprocal proximity information and interactions between participants of three social events. Moreover, we were able to collect temporal data on such contacts, allowing to simulate all the possible ways of spreading a given message, setting different hypotheses on the source node and the time when the packet is originated. The analysis of the dataset created by the short range human mobility information can shed light on hidden patterns in social dynamics, as on other open questions as sparsity distribution and short range trajectory patterns.

After data are collected and filtered accordingly our scope, we define a simple framework to represent all the given information. The contact graph that we introduce, is suitable for running simulations of different routing strategies over it, under very general store-carry-forward assumptions. By simulating the spreading process along the data collected, we try to answer to a very fundamental question: what should we expect from social behaviors for better defining routing strategies? We are highly motivated to find some universal patterns, invariants to all the events we observed, useful to limit the boundaries of a generic process for data dissemination in terms of coverage and reachability.

Finally, once such common findings have eventually emerged from the raw collected data, we can use them to validate existing models that are adopted to produce synthetic behaviors, assumed to be similar to real patterns. The proposal of underlying models that explain human mobility patterns is already an active research field, and such models should be validated by data in different levels of granularity. We aim to validate, adapt and correct these models using our findings. By shedding light on the processes supporting opportunistic and delay-tolerant ad-hoc networking strategies, our purpose is to evaluate and improve the performance of information diffusion algorithms, reduce the cold start of recommendation applications that can take advantage of opportunistic information diffusion, and understand how to reduce the impact of undesirable epidemic phenomena.

5.1 Related Work

Most analytical frameworks for message diffusion, such as [86], are stochastic models used to compute message delay distributions based on parameters describing transmission range and inter-contact time distributions, with no special characterization of the causal structure of message propagation. Other works such as [39] and [108] also focus on the analysis of the distributions of inter-meeting intervals.

In [30] Boldrini et al. reveal some clues about modeling data dissemination in opportunistic networks. Their goal is to understand if the data-dissemination system reaches stationary regimes and to characterize their properties. They propose a Markovian model of the data distribution process resulting from the dissemination system. However, their modeling is based on synthetic data, and it uses a Markovian representation of the data distribution process.

Other works focus on mobile content delivery [124] and delay-tolerant networks [93]. In [138] the authors argue that validating mobility models is challenging because little experimental data is available, and propose to examine just people's encounters, as opposed to analyzing their mobility patterns. In [125], there are interesting insights on the influence of contact dynamics over routing strategies in delay tolerant networks. Furthermore, in the DTNs domain, [198] proposes a new paradigm for fault tolerant pervasive information gathering. This framework is analyzed, by means of two basic approaches, namely, direct transmission and flooding. They propose two novel data delivery schemes, whose aim is to minimize transmission overhead in flooding. The framework is studied by means of queuing theory and statistics, and various assumptions that still need to be validated are done along the paper.

Epidemic routing [190] is particularly relevant to this work. In fact, Epidemic routing follows a store-carry-forward approach originally proposed for sparse and/or highly mobile networks in which there may not be, in a given moment, a path from source to destination. Analogies with the spread of infectious diseases are quite straightforward: the carrier infects another node by sending a data packet when it is in the proximity of the receiver. Many epidemic routing strategies have been proposed and performance evaluation analyses have been carried on (e.g., [207], [186], and [129]).

Some effort has also been devoted to characterizing forwarding paths. Chaintreau et al. [50] state that the structure of mobility networks is in general characterized by a small diameter; i.e., a device can be reached using a small number of relays. This is shown analytically for random graphs, and empirically based on data from conference deployments. Based on this observation, the authors introduce an efficient algorithm to compute the delay-optimal path between nodes that exploits the small-world character of the underlying mobility network. Erramilli et al. [72] investigate message forwarding in conference settings and characterize optimal paths in time and space. They find that these paths, while optimal, may take a very long time to reach the destination (thousands

of seconds), and report a so-called "path explosion phenomenon", i.e., that shortly after the optimal path reaches the destination, a large number of nearly-optimal paths does the same.

Since portable devices carried by humans are becoming more and more pervasive, several solutions have been proposed that exploit the interplay between the structural properties of social networks, mobility aspects, and information diffusion. Daly and Haahr [60] propose an algorithm (SimBet) that uses social network properties such as betweenness centrality and social similarity to inform the routing strategy. Simulations based on real traces show a performance comparable to Epidemic Routing, without the associated overhead, and without a complete knowledge of the network topology. Hui et al. [99] aim at using social structures to better understand human mobility and inform forwarding algorithms. Based on real-world traces, the authors observe high heterogeneity in human interactions both at the level of individuals and of communities. The socially-aware forwarding scheme they devise (BUBBLE Rap) exploits such heterogeneity by targeting nodes with high centrality as well as members of the communities, yielding delivery ratios similar to flooding approaches with lower resource utilization. Pietilainen et al. [163] propose a middleware (MobiClique) that exploits ad-hoc social interactions to disseminate information using a store-carry-forward mechanism. Data collected from the deployment of the MobiClique system at two conference gatherings demonstrates its ability to create and maintain ad-hoc social networks and communities based on physical proximity.

5.2 Dynamics of information spreading

In this section, we report on a data-driven investigation aimed at understanding the dynamics of information spreading in a real-world dynamical network of human proximity. We use data collected during three different social gatherings in which the SocioPatterns plataform was deployed, simultaneously involving several hundred individuals. We simulate a information spreading process over the recorded proximity network, focusing on both the topological and the temporal properties. We show that by using an appropriate technique to deal with the temporal heterogeneity of proximity events, a universal statistical pattern emerges for the delivery times of messages, robust across all the data sets. Our results are useful to set constraints for generic processes of data dissemination, as well as to validate established models of human mobility and proximity that are frequently used to simulate realistic behaviors.

By focusing on wireless short range communications only, and assuming that a unit of information (i.e., a packet) is transmitted when two nodes are in proximity of each other (given some definition of *proximity*), we try to understand the spatio-temporal dynamics of the network of human contacts. For the pursuit of this goal, we intend to start from a collection of real world data.

In order to accomplish the data collection, we used the SocioPatterns platform to locate, with a fine-grained granularity, reciprocal proximity information between participants of three social events: *25C3*, *SFHH* and *HT09* (see Section 2.3.1 of Chapter 2 for more details about the events and datasets). Moreover, we were able to collect temporal data on such proximity patterns. As we will see later, this is very relevant to the purpose of our investigation, because it allows us to simulate all the possible ways of spreading a given message, setting different hypotheses on the source node and the time when the packet is originated. Let us observe that even if we are collecting contact information by means of RFID devices, the results coming from our experimental settings are independent from the transmitting technology. Moreover, we used different ranges for proximity sensing in the three scenarios used for collecting data. This allows us to compare our data with other deployments using Bluetooth or other short range wireless systems.

We will use the framework defined in Chapter 3 to represent the collected information as *Dynamical Networks*. We use a time-dependent contact graph that is suitable for running simulations of different routing strategies, under very general store-carry-forward assumptions. More precisely, our approach is based on building the so called *Fastest Route Trees*, generated by simulations of spreading process along the measured dynamical proximity network. Our analysis is based on both a topological and a temporal point of view in order to give as much generality as possible to our findings. This is done because we need to answer a very fundamental question: what should we expect from social behaviors for better defining routing strategies? We were highly motivated to find some universal patterns, invariant in all the events we observed, useful to define the boundaries of a generic process for data dissemination in terms of coverage and reachability. Once such robust patterns have eventually emerged from the collected data, we can use them to validate existing models that are commonly used to generate synthetic behaviors, assumed to be representative of real behaviors. Of course, this is a really challenging task and we do not aim here at exhaustively covering this aspect.

5.2.1 Information spreading process

The information spreading process is simulated by supposing that any entity which could be subject to spreading over the contact network can be modeled as a message. In order to obtain results that could be generically applied, we used a theoretical scenario, where nodes have an infinite amount of resources, and message exchanging delays are not considered. The result is a discrete Susceptible/Infected (SI) process simulation, which is one of the most basic compartmental models known in epidemiology [110].

We define a message exchanging protocol that specifies the behavior of any pair of nodes when they are in contact. In the case of a simple flooding protocol, if two nodes i and j are in contact, i sends all its known messages to j and vice-versa. If node i receives a message that it has not yet received, it keeps the message in its memory, and the same

for j. In our theoretical scenario, both i and j have an infinite amount of resources, so the local storage is unlimited and no message is discarded.

In order to model the message spreading process, we define a message M_{n_0,t_0} generated by a node n_0 at time t_0. We choose n_0 among all $i \in N$, where N represents the ensemble of all nodes, and t_0 as some moment during the experiment timeline. We will use the previously defined message flooding protocol, since it serves as the best case for message spreading. Theoretically, this corresponds to an epidemic process on top of the dynamical contact network, allowing us to probe the causal structure of the network and the interplay of topology and activity burstiness. It is important to remark that, for the case of the collected data, if we choose any arbitrary node n_0 and an initial time t_0, the first message exchanging could occur a long time after t_0, as the first opportunity for transmission depends on the time of the first contact involving n_0 (after t_0). For example, if we choose t_0 in the middle of the night, it could take hours to n_0 to forward the message to the first node it interacts with.

Once a spreading process starts, whenever node i makes contact with node j at time t and propagates a message that j has not yet received, we count this contact $\langle i, j, t\rangle$ as relevant to this specific spreading process. Each initial pair $\langle n_0, t_0\rangle$ yields a different spreading history, with different relevant contacts. These relevant contacts form a tree where n_0 is the root node and all relevant contacts are edges. We call it the Fastest Route Tree $\mathrm{FRT}(n_0, t_0)$, as each path between n_0 and $j \in \mathrm{FRT}(n_0, t_0)$ represents the *fastest route* along which a message generated by n_0 at time t_0 would arrive at j using the message flooding process. The initial time t_r of $\mathrm{FRT}(n_0, t_0)$ is the first time n_0 propagates the message, that is, the earliest t of all contacts $\langle i, j, t\rangle \in \mathrm{FRT}(n_0, t_0)$.

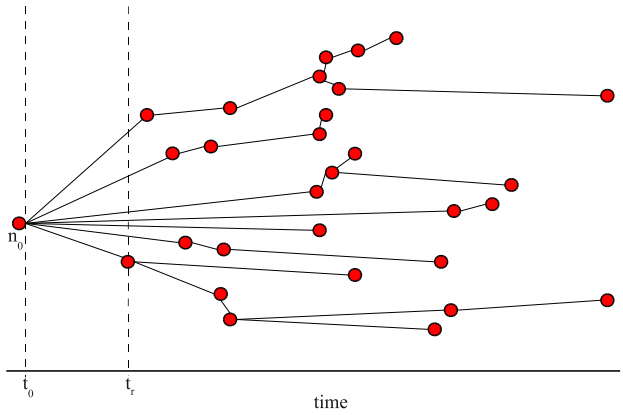


Figure 5.1: A typical Fastest Route Tree $\mathrm{FRT}(n_0, t_0)$. The position of each node along the x axis represents the time when the node received the spreading message. Node n_0 injects the message at t_0. The first node that receives the message from n_0 is represented at time t_r.

A way to graphically represent $\mathrm{FRT}(n_0, t_0)$ is shown in figure 5.1. It is a schematic

visualization of the spreading history, represented as a tree, where each node is horizontally placed according to the time of the message reception, with edges representing the transmission events.

5.2.2 Analysis methodology

Many works on mobility networks have focused on general characteristics such as the distribution of inter-meeting times between nodes. Inter-meeting times represent one of the key metrics in forwarding algorithms, and are typically assumed to be exponentially distributed, although some studies found power-law distributions in some circumstances [108]. In the present work, we do not consider the distribution of all inter-meeting times, but focus instead on those contacts that are relevant to the spreading process, i.e., through which messages are propagated. In other words, we only consider the times between contacts represented in the FRT.

To this aim, we propose an analysis based on building Fastest Route Trees generated by simulations of spreading process along data collected in the three above-mentioned SocioPatterns deployments. For each node n_0, and for several starting times t_0, we build the FRT(n_0, t_0). We analyze these FRTs both from a topological and from a temporal point of view.

5.2.3 Fastest Route Tree structure

The topological analysis of the FRTs can be used to unveil information about the importance of each node in the spreading process. In particular, the spreading activity of a node is quantified by the number of nodes to which it has sent a message. For each node n_i, we therefore measure its average out-degree (i.e., the average number of direct children) in FRT(n, t_0). Figure 5.2 shows the probability density of this quantity in a semi-log plot, computed for each dataset for 50 different values of t_0 and for all possible choices of the root node at t_0. The distributions exhibits an exponential decay for the *SFHH* and *HT09* deployments, in which the contact detection range was short, and a broader shape for the *25C3* case, which had a broader detection range.

From its definition, the FRT consists of successive topological levels. The root node, from which the message was initially sent, is at level 0. Level 1 is formed by the nodes who received the message directly from the root. More generally, level ℓ consists of all nodes who received the message from a node at level $\ell - 1$. Nodes at level ℓ receive therefore a message which has been transmitted ℓ times from the root. For each n_0, the number of nodes at level ℓ is $N(n_0, \ell)$, and we compute the distribution $P_\ell(N)$ of these numbers, computed for all possible root nodes n_0. Figure 5.3 displays the corresponding box plots. The number of nodes at a given FRT level typically grows for small values of

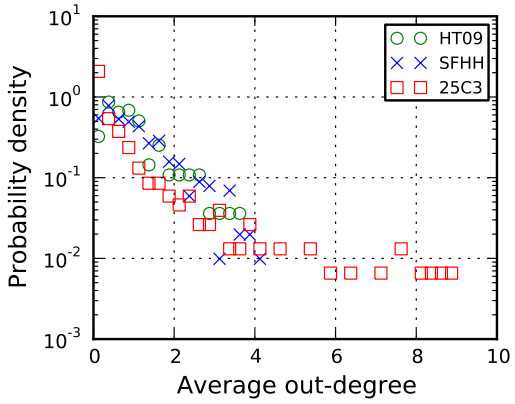


Figure 5.2: Probability density of the average node out-degree (number of direct children) for all nodes in the FRTs in the three different SocioPatterns deployments. The probability densities are binned on intervals of width 0.25 and are computed for 50 different message injection times and for all choices of the root node at a given initial time. For deployments with short contact detection range (*HT09* and *SFHH*), the distribution appears approximately exponential, while it is broader for the deployment where a longer range was used (*25C3*).

ℓ, reaches a maximum, and then decreases. Figure 5.3 is in fact similar to usual shortest paths distributions found in networks. The strong distinction in this case is that we are dealing with *fastest* paths between nodes, in a dynamically evolving network, which are known to be different from the shortest paths in the corresponding static aggregated networks [118, 103].

5.2.4 Arrival times

Messages may reach at very different times the nodes belonging to the same level of a FRT. It is therefore important to study, for each tree level, the distribution of arrival times. To this aim, we record, at given initial time t_0, for each root node n_0 and each level ℓ, the arrival times of the message at node i $\{t_i(n_0, \ell)|\ell = 1, 2, 3...; n_0 \in N\}$. Figure 5.4 displays the histograms $P_\ell(t|t_0)$, computed over all choices of n_0, for several levels ℓ and two starting times t_0, together with the global distribution of contact times. More precisely, in Fig. 5.4, the x-axis shows the time t, and the y-axis gives the probability that an arrival time (or a contact, for the top row) falls in the interval $[t-\Delta, t+\Delta]$, with $\Delta = 30mn$. In Figure 5.4(a) the spreading starts at $t_0 = 35$ hours, when the contact density is low, while for Figure 5.4(b) $t_0 = 45$ hours, when the contact density is high. The comparison of Figs. 5.4(a) and 5.4(b) illustrates how the global temporal patterns of the contacts between nodes impacts the arrival times. When the spreading starts during a period in which the contacts are rare, very large delays are observed. On the contrary,

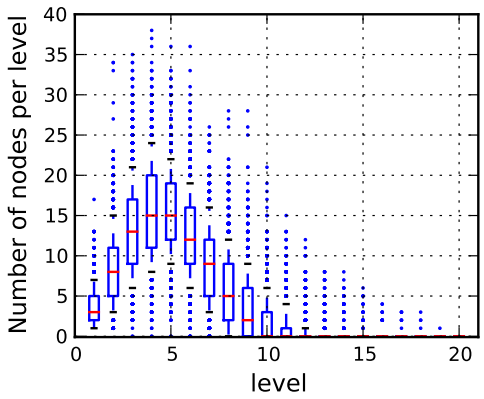


Figure 5.3: Box plot of the number of nodes reached at each level of the FRTs. In each box, the red dash represents the median, the bottom and top of the box are the 25^{th} and 75^{th} percentile, and the ends of the whiskers are the 10^{th} and 90^{th} percentile. Dots represent outliers.

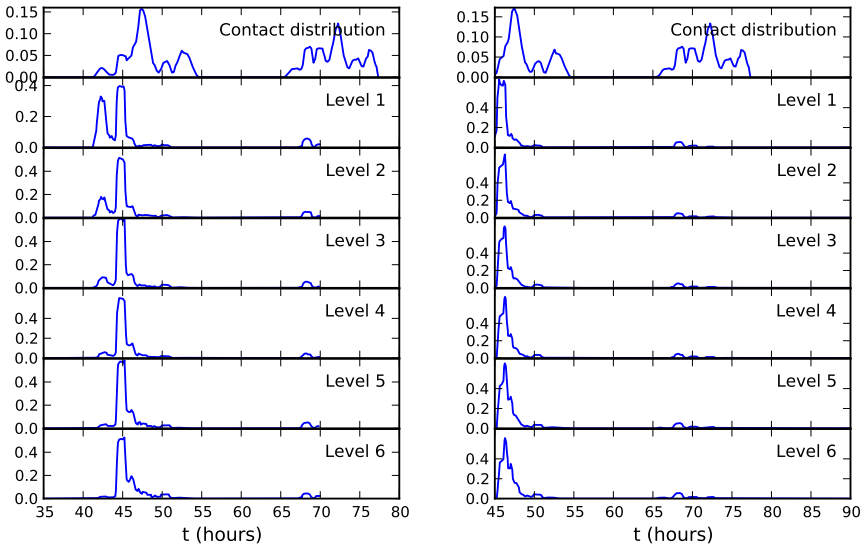
a message starting during a period of strong interaction is spread very fast, with most of the nodes receiving the message after less than 2 hours.

5.2.5 Delivery time metrics

The previous analysis has shown how the analysis of message arrival delays in a real-world scenario is affected by the heterogeneity of the contact density in different periods. The SocioPatterns deployments show indeed how the social behavior of individuals tends to be characterized by bursty periods of intense activity, separated by "quiet" periods in which very few contacts are observed. This pattern clearly affects our ability to compare the delivery delays of messages in different spreading processes, which may have started during periods of very different levels of activity. In the following we focus on defining a new approach to the measure of time delays in a message spreading process.

The most straightforward approach to calculate the message delay time in a FRT started at t_0 at node n_0 consists in measuring, as in the previous subsection, the elapsed time between the message generation at t_0 and the delivery time t_i at each node i. Figure 5.5(a) shows the distribution of elapsed times $t_i - t_0$ for two different starting times t_0. For each starting time, the distribution is computed over all root nodes n_0 and over all arrival nodes i. The first starting time is chosen to lie in a period of low contact density, while the second falls in a period of high contact density. As already pointed out above for the delivery times at the various levels of the FRTs, different starting times can lead to very different delivery time distributions.

A first effect of contact density at the time of message injection comes from the fact

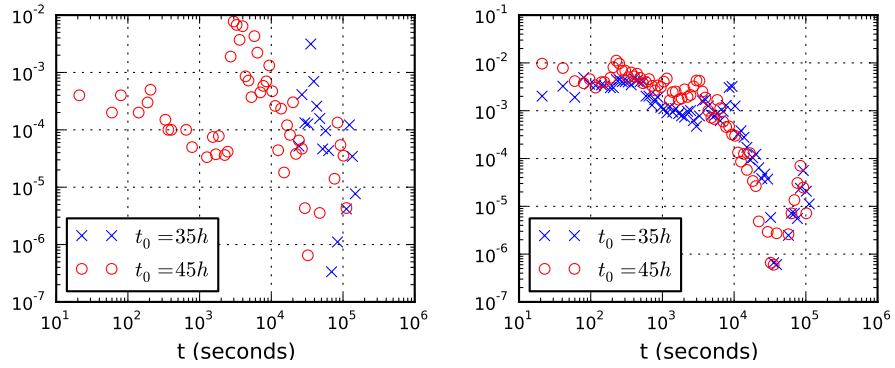


(a) $t_0 = 35$ h (b) $t_0 = 45$ h

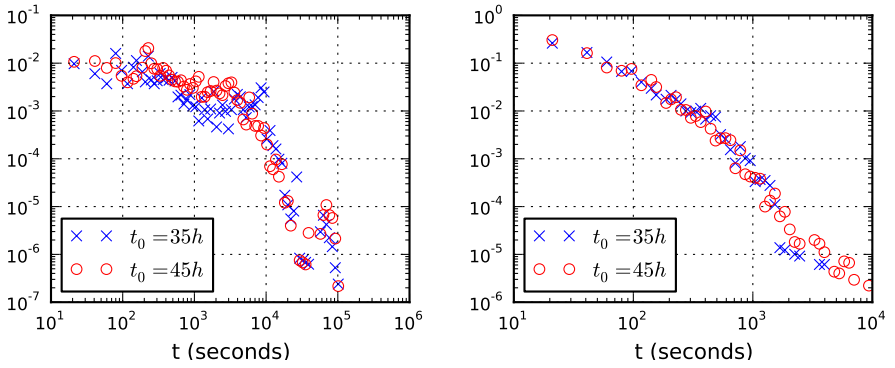
Figure 5.4: Histograms of the arrival times at various levels of the Fastest Route Trees, for two different initial times of the spreading process. The first row shows the distribution of all contacts in time, while the subsequent rows show only the contacts used to spread messages in each level. Each histogram shows where the contacts responsible for the spreading of the messages in each level are concentrated in time, if the message is created at time t_0. In (a), the messages are generated at time $t_0 = 35$ h and in (b) the messages are generated at $t_0 = 45$ h. In the histograms, the y-axis shows the probability that a contact is in the interval $[t - \Delta, t + \Delta]$, with $\Delta = 0.5$ hours.

that the first contact of n_0 with another node can occur at a (much) later time t_r. In particular, the message may be generated during a period in which n_0 is isolated, blocking the propagation of the message until a contact involving n_0 occurs. A way to take this into account consists in choosing the time t_r of the first contact as the starting time for the computation of delays. The delivery time for node i is thus computed as $t_i - t_r$. The corresponding distributions of elapsed times are shown in Fig. 5.5(b). There is much less difference in the distributions than in the previous case. However, the distributions do not exhibit any clear functional form, and are still strongly impacted by the time variation of the contact density. For instance, a certain number of nodes receive the message only during the second day of the conference, simply because they were not present during the first day.

This last point suggests to consider, for each node i, the time t_i^0 at which it appears for the first time in the system. Figure 5.5(c) therefore shows the distribution of the time difference between the arrival time at node i, t_i, and t_i^0, defined as the first time after t_r in which node i has a contact. This difference represents how much time the



(a) $t_i - t_0$ (b) $t_i - t_r$



(c) $t_i - t_i^0$ (d) Elapsed Contact Time

Figure 5.5: Log-binned distribution of time delays. Each distribution represents a different way of quantifying the time delay. (a) the delay is defined as the difference of time between the arrival of the message at node i and the generation time of the message. (b) the time origin is taken as the first time t_r at which the root node n_0 has a contact, after the message has been generated. (c) the time origin for a node i is t_i^0, the time at which it first had a contact since the message generation. (d) for each node, the time increases only when it is in contact with other nodes: one counts only the time which can be used for propagation purposes, i. e., the total time the node was effectively in contact with other nodes.

message took to reach i, once i was able to receive it. As for Fig. 5.5(b), the shape of the distribution depends on the distribution of contacts: there are more points in periods where the contact density is higher.

The fact that the distribution of delivery times strongly depends on the temporal heterogeneity of contacts hints at an alternate way to define time, which is intrinsically more robust with respect to contact density fluctuations. The idea is to turn to a nonuniform time frame in which we use the time a node spends in contact as a clock for the

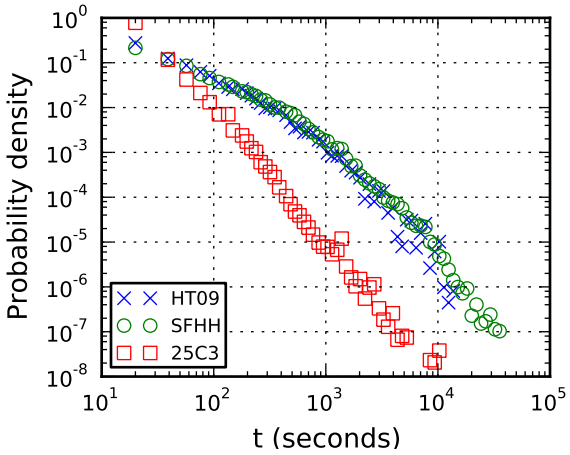


Figure 5.6: Probability density distributions of elapsed contact times for three different deployments.

process under investigation, viewed from the perspective of that given node. To this end, we trade a globally defined time for a node-specific clock, which only ticks forward when the node is involved in a contact. The clock of node i is then defined as the total number of frames in which i has been present and in contact with any other node, starting from zero at the moment the spreading process starts. That is, the clock of node i starts ticking the first time node i participates in a contact occurring after the starting time t_r. Using these node-dependent clocks, the message delivery delay for node i is defined as the cumulated time node i has spent in contact, from the time t_r, when the message diffusion starts, to the moment when i receives the message. The clock of a given node does not advance during the time intervals in which that node is isolated or not present, and therefore cannot receive any message. The efficiency of a given protocol is quantified by using a measure grounded in the contact activity of each node. The corresponding distributions of message delivery times, measured in terms of *elapsed contact time* are shown in Fig. 5.5(d), and are very robust with respect to a change in the injection time of the message.

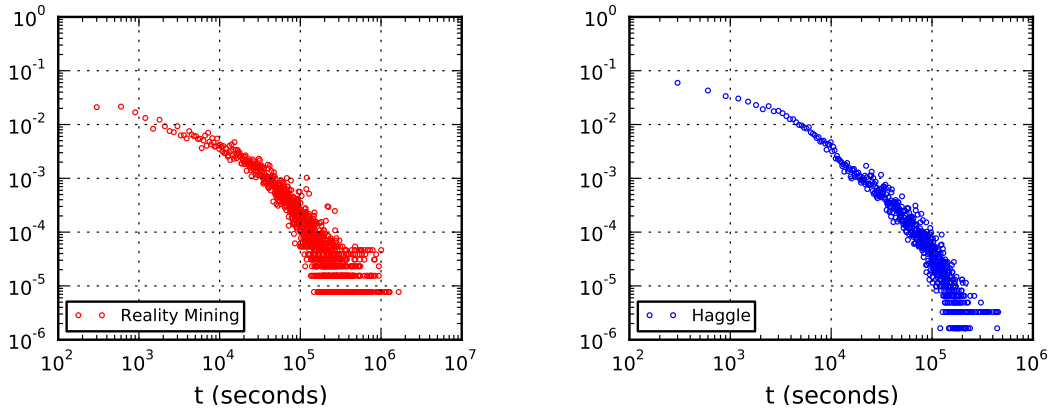
Figure 5.6 displays the distributions of elapsed contact times for the different deployments, computed for 10 choices of the injection time. Strikingly, the distributions are superimposed for the two deployments in which the same contact detection range was used, namely *HT09* and *SFHH*, although the time sequences of contacts was clearly very different (with sessions, lunches and coffee breaks taking place at different times). For the *25C3* deployment, in which the contact detection range was more extended, the distribution is different.

In all cases, the distribution is maximal at short delays: the probability that a node receives a message at its first contact event is large. Moreover, the distributions are

broad, extending over a large range of possible delays.

In summary, at a given detection range, the distribution of message delay, using as a clock for each user the time in which it is in contact, does not depend on the deployment, at fixed contact detection range, nor on the timeline of contacts and of their densities, nor on the starting time of the spreading process.

We applied the same temporal analysis in other experiments and datasets, like the MIT/Reality Mining dataset and the Haggle dataset. The results are shown in figure 5.2.5. We see that the same pattern of long-tailed distributions apply to other datasets, independently of the technology or the transmission range used in the experiment.



(a) Reality Mining dataset (b) Haggle dataset

Figure 5.7: Probability density distributions of elapsed contact times for (a) the MIT/Reality Mining dataset and (b) the Haggle dataset.

5.3 Comparison with data generated by synthetic models

In the previous paragraphs, we have shown how to measure message delays in a way that yields robust distributions across different real-world sequences of contact events. We now turn to a comparison with the outcome of contact sequences generated by models. Protocols are indeed most often validated against data generated by synthetic models of contact networks, and it is important for these models to accurately reproduce the phenomenology of real-world data sets.

An extensive analysis of all synthetic models used by the research community is beyond the scope of this work. We therefore focus on two models widely used when dealing with opportunistic and delay-tolerant protocols: the Random Waypoint model

and the Truncated Lévy Walk model [169]. Using the analysis described above, it is possible to see how much the models' generated data is close to or differs from real world data, with respect to the characteristics involved in the dynamics of information spreading.

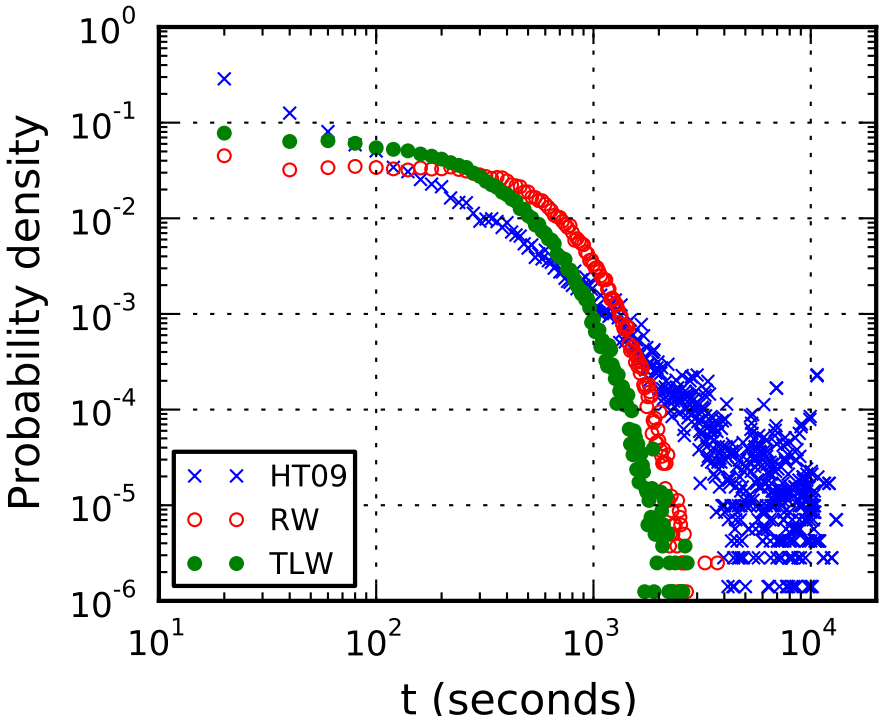


Figure 5.8: Comparison of the distribution of elapsed contact times with three data sets, one collected in the *HT09* deployment and the other two generated by mobility models, both with 100 nodes and contact detection range of 2 meters: the Random Waypoint model with node speed distributed uniformly from 0.01 to 0.1 m/s, and the Truncated Lévy Walk model where flight lengths and pause times follow truncated power laws with $\alpha = 1.6$ and $\beta = 0.8$. No binning was used to represent the data.

In Figure 5.8 we compare the distribution of elapsed contact times, as defined in the previous subsection, between the injection of the message and its reception by all nodes, for real-world data (*HT09*) and for contact sequences generated by the two chosen mobility models. In simulating the models, we use 100 nodes with a contact detection range of 2 meters in a square area of $40m \times 40m$, parameters that are close to the ones of the real-world data sets, and we adapted the parameters of path lengths, node speed and pause times to produce data sets with contact time distributions close to the real-world distributions. For the Random Waypoint model, we used node speeds uniformly distributed between 0.01 and 0.1 m/s. For the Truncated Lévy Walk model, flight lengths (l) and pause times (t) follow truncated power laws $p(l) \sim l^{-(1+\alpha)}, l < l_{max}$

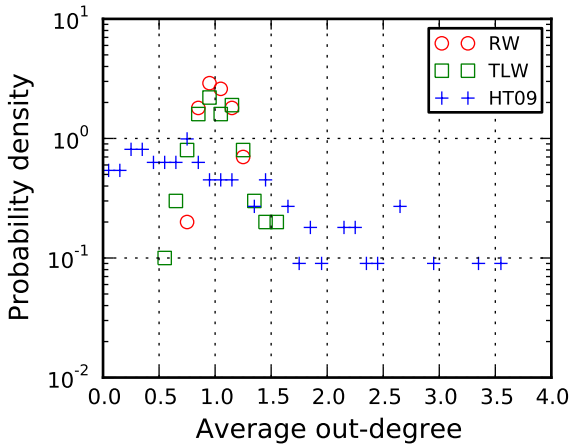


Figure 5.9: Probability density of the average node out-degree (number of direct children) in the FRTs for the Random Waypoint model, the Truncated Lévy Walk model, as well as the empirical data for the *HT09* deployment. The probability densities are binned on intervals of width 0.1 and are computed for 50 different message injection times and for all choices of the root node at a given initial time. The empirical data appear to exhibit higher heterogeneity than the simulated data, where the average out-degree is peaked at approximately $(n-1)/n$ for all nodes.

and $p(t) \sim t^{-(1+\beta)}, t < t_{max}$ with $\alpha = 1.6$, $\beta = 0.8$, $l_{max} = 40m$ and $t_{max} = 1h$, with turning angles taken from a uniform distribution and node speed increasing with the flight length. In order to simulate the data, we used the ONE simulator for DTN protocol evaluation [109] with a customized report that produces proximity data every 20s.

It is important to notice that in order to make the comparison between data and models more meaningful and simpler to interpret, we decided to compare the synthetic data against real-world data from the smallest deployment of this study (*HT09*). For this conference dataset we know that the cumulative contact network exhibits no significant modular structure: thus we do not incur in the difficulty of comparing real-world data against models that cannot produce any modular structure, nor do we incur in the additional complexity of defining models that respect a known modular structure in the real-world data.

A strong difference is observed between the distributions generated by the two types of data, with a much narrower distribution for the model data than for the real-world ones, as shown in Table 5.1 by the comparison of the ratios between variance and average of the distributions. This is particularly striking as the two models considered here correspond to very different mobility patterns, with respectively homogeneous (for the Random Waypoint) and heterogeneous (for the Truncated Lévy Walk) distributions of

Dataset	Average	Standard Deviation	Std. Dev. / Average
25C3	31.6175	61.6218	1.9489
SFHH	323.3640	790.2398	2.4438
HT09	262.2559	692.2294	2.6395
RW	377.0415	294.8501	0.7820
TLW	236.9179	210.4378	0.8882

Table 5.1: Statistical properties of the delivery time distributions: 25C3, SFHH and HT09 refer to the experimental data sets, while RW and TLW are synthetic data sets generated by simulating the Random Waypoint and the Truncated Lévy Walk models. Notice how the high dispersion of experimental data, characterized by long-tailed distributions, contrasts the low dispersion of the simulated data sets.

flight and pause times. To further probe this point, we show in Figure 5.9 the probability density of the average node out-degree in the Fastest Route Trees corresponding to both the real-world and the simulated data: also in this case the simulated models, including the Lévy process, fail to reproduce the empirical behavior.

Although more extensive research is needed to extend the comparison to data sets created by other models, this preliminary analysis shows that the introduction of realistic individual mobility patterns (through power law distributions for instance) is not enough to fit real-world propagation patterns. Taking into account the information about contact patterns when measuring the properties of spreading dynamics is crucial to unveil characteristics that can differ strongly between a model's outcome and the real-world dynamics.

Finally, we add a word of caution about the analysis of data generated by synthetic models. Very often, opportunistic and delay-tolerant protocols are evaluated through measures of average delay times and standard deviations. These quantities may not be very representative in the case of broad distributions of delays, such as the ones observed here. In these cases, the whole distribution should be considered instead.

5.4 Conclusion

In this chapter we studied the process of data diffusion in a real-world dynamic networks of human proximity. We analyzed the topological and temporal dynamics of the networks, focusing on the interactions between participants in three large-scale social gatherings. We highlighted the temporal heterogeneity that arises from a number of social activities.

To investigate the general properties of information propagation, we focused on a simple flooding routing protocol that allows us to expose the interplay between network topology and the bursty nature of human activity. The dynamics of message diffusion

is captured by the so called *Fastest Route Trees* that represent the fastest route along which a piece of information can flow from the origin to the proximal nodes. The activity of a node is quantified by the number of nodes to which it has propagated a message. The activity distribution displays an exponential decay for the two deployments with short-range proximity sensing (*SFHH* and *HT09*), and a broader tail for the case with a longer detection range (*25C3*).

We showed that the distribution of message delivery times is strongly affected by the temporal heterogeneity of proximity events. When the spreading starts during a period of low social activity, very large delays are observed. On the contrary, a message originated during a period of high interaction tends to spread fast. We studied the effect that different definitions of "delivery time" have over the delivery time distribution. In particular, we introduced here a new notion of "intrinsic" time, specific to every node, that measures the cumulated time that node has been in proximity with any other node of the system. In other words, we trade a globally defined time for a user-specific clock, which only advances when the corresponding user is engaged in a proximity relation. Strikingly, we find that by using this definition of time, the delivery time distributions assume a generic form. That is, the distribution is the same for distinct deployments with the same contact detection range, and does not depend on the detailed timeline, nor on the initial time of the spreading process.

Moreover, we made a first step at comparing the measured sequences of proximity events with sequences generated by using commonly accepted models of human mobility, such as the Random Waypoint model and the Truncated Lévy Walk model, which are widely used in the domain of opportunistic and delay-tolerant networks. Even though an extensive comparison of the models used in the literature against data is outside the scope of this work, we report a strong difference between the propagation processes on model-based and real-world proximity networks. This points to the importance of taking into account realistic contact patterns, and not only individual mobility patterns, for studying dynamical processes on dynamical proximity networks. In fact, the dynamics of information diffusion depends on non-trivial properties of contacts and inter-contact time intervals, at least as much as on the topological and temporal heterogeneity of human mobility. Our results call for future work in the direction of defining fine observables that can capture those properties of the proximity networks that bear relevance to a variety of general processes occurring over them. Such observables could be used to compare the synthetic proximity networks generated by established models of human mobility with the proximity networks recorded in experimental settings. This will allow to expose the limits of current mobility models, and to devise more realistic modeling schemes.

Chapter 6

Collaborative Filtering for Selective Information Dissemination

In this chapter we discuss some collaborative filtering techniques used to obtain selective information dissemination. Most of the basic techniques for information dissemination through recommendation systems can be found on Adomavicius et al. [2], so in this chapter we focus on the improvements given to collaborative filtering algorithms since the survey was compiled.

Originally, *selective dissemination of information* (SDI) was first described by Hans Peter Luhn of IBM in the 1950's [131]. As described by Luhn Selective Dissemination of Information was part of the business intelligence system, involving the use of the computer to select from a flow of new documents, those of interest to each of a number of users. This process could be thought of as a complement to information retrieval. It was a topic related to library and information science, referring to tools and resources used to inform users about new resources on selected topics. The term itself is somewhat dated, but contemporary analogous systems include alerts, awareness tools or trackers. SDI systems provide automated tools to inform users about the availability of new resources meeting certain keywords or search parameters.

Recommender Systems (RS) refers to a broad spectrum of mathematical modeling techniques and software tools providing suggestions for items to be of use to a user [170]. In this context, a series of data mining techniques are used to infer recommendation rules or build recommendation models from large data sets. Recommender systems that incorporate data mining techniques make their recommendations using knowledge learned from the actions and attributes of users. Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices [168].

Accordingly to the nature of data-driven applications that produce information over-

load, users need support to make choices, even without sufficient personal experience of the alternatives. In real life, "Word-of-Mouth" is a very common process for the user to cope with this issue, and social networking is the natural framework for finding affinities and exploiting collaborations between users and for filtering relevant and personalized information. Many popular RS rely on user profiling and/or a given object domain-dependent taxonomy.

Suggestions provided by RS are aimed at supporting their users in various decision-making processes, such as what items to buy, what music to listen, or what news to read. Recommender systems have proven to be valuable means for online users to cope with the information overload and have become one of the most powerful and popular tool in electronic commerce. Correspondingly, various techniques for recommendation generation have been proposed and during the last decade, many of them have also been successfully deployed in commercial environments.

Collaborative filtering is one of the most successful technologies for RS. In the last decade, it has been developed and improved, and a wide variety of algorithms exist for generating recommendations. There has been much work done both in the industry and academia on developing new approaches to RS. The interest in this area still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users to deal with information overload and provide personalized recommendations, content, and services to them.

Since the survey compiled by Adomavicius et al. [2] published in 2005, large improvements have been made in Recommendation Systems and, mainly, collaborative filtering techniques. A lot of attention has been given to this area from the industry, with focus on the *NetFlix Prize*. In October, 2006 Netflix released a large movie rating dataset and challenged the data mining, machine learning and computer science communities to develop systems that could beat the accuracy of Cinematch by certain amounts. Winners of the various Prizes were required to document and publish their approaches, enabling everyone to understand and benefit from the insights and techniques required to achieve the enhanced levels of predictive accuracy.

The attention given by the industry to RS pushed this area to the center of interest, not only in academia, but also different groups, most of them multidisciplinary groups involving mathematicians, computer scientists and even psychologists. In 2009 the Netflix Prize contest was won by a team comprised of software and electrical engineers, statisticians and machine learning researchers from Austria, Canada, Israel and the United States. But the great contributions given to the area of recommendation systems were the improvements in the collaborative filtering algorithms.

In this chapter we will focus on the improvements given to collaborative filtering algorithms since the survey compiled in 2005. These improvements can be summarized in the following techniques used to improve ratings prediction: jointly derived interpolation [23], matrix factorization [206][116] and restricted Boltzmann machines [172].

6.1 Background

More formally, the recommendation problem can be formulated as follows: Let U be the set of all users and let S be the set of all possible items that can be recommended. The space S of possible items can be very large, ranging in hundreds of thousands or even millions of items in some applications. Similarly, the user space can also be very large. Let r be a function that measures the preference of user u to item s, i.e., $r : U \times S \rightarrow R$, where R is a totally ordered set, ordered by preference value. Then, for each user $u \in U$, we want to choose such item $s' \in S$ that maximizes the user's preference estimation. In RS, the preference of an item is usually represented by a rating, which indicates how a particular user liked a particular item.

Each element of the user space U can be defined with a profile that includes various user characteristics: age, gender, income, marital status, etc. In the simplest case, the profile contains only a user identifier. Similarly, each element of the item space S is defined with a set of characteristics. In a movie recommendation application for example, each movie can be represented not only by its identifier, but also by its title, genre, director, year of release, leading actors, etc. Finally, the preference r can also be represented as more than one characteristic, for example the rating given to the item s by user u and the date when the rating was given by the user.

The central problem of RS lies in that ratings is usually not defined on the whole $U \rightarrow S$ space, but only on some subset of it. This means r needs to be extrapolated to the whole space $U \rightarrow S$. In RS, the ratings are initially defined only on the items previously rated by the users. For example, in a movie recommendation application, users initially rate some subset of movies that they have already seen. For the non-rated movie/user combinations, the recommendation engine should be able to estimate (predict) them and issue appropriate recommendations based on these predictions.

Extrapolations from known to unknown ratings are usually done by 1) specifying heuristics that try to estimate user preferences and empirically validating its performance and 2) estimating preferences that optimizes certain performance criterion, such as the mean square error. Once the unknown ratings are estimated, actual recommendations of an item to a user are made by selecting the highest rating among all the estimated ratings for that user, according to (1). Alternatively, we can recommend the N best items to a user or a set of users to an item.

In order to implement its core functions by identifying items that are useful for a user, a RS must estimate the preference of an item for a user. The system must be able to estimate the utility of some item to a user, or compare its utility with other items in order to decide which items are worth of recommendation. Ratings of not-yet-rated items can be estimated in many different ways using methods from machine learning, approximation theory, and various heuristics. Recommender systems are usually classified according to their approach to rating estimation. Moreover, RS are usually classified into the following

categories, based on how recommendations are made:

Content-based: The system recommends to a user items that are similar to the ones the user preferred in the past. Similarity between items is calculated based on features like genre, price, color, etc.

Collaborative Filtering: The simplest implementation of this approach is by recommending to a user the items that other users with similar tastes and preferences liked in the past. Similarity between users, or between items in case of item-based approaches, are calculated based in the rating history. This is the most popular and widely implemented technique in RS.

Hybrid approaches: These methods combine collaborative and content-based methods.

In addition to RS that predict the absolute values of ratings that individual users would give to the yet unseen items, there has been work done on preference-based filtering, i.e., predicting the relative preferences of users. For example, in a movie recommendation application, preference-based filtering techniques would focus on predicting the correct relative order of the movies, rather than their individual ratings.

6.2 Related work on Recommender Systems

While the Internet contains a huge volume of digital information, it is often difficult for users to find the information they really want without having a direct experience about the alternatives [168]. To reduce data overloading, several techniques such as data indexing, retrieving, searching and filtering has been developed in the past. The semantic of these approaches is simply to find an exact match between a query string and content without any consideration of user's preferences. To provide a sort of *personalized advice*, a *recommender system* is commonly adopted to support users in finding useful information according to their likings. The most relevant proposals in the recommendation area are the *content-based* and the *collaborative filtering* techniques.

In the *content-based* approach, the system suggests the items that are similar to the items previously liked by the user. Resources' similarities are evaluated by way of an objective and automatic analysis of the content, identifying some distinguishing features and comparing them to the user's profile. Some examples are the *Syskill & Webert* recommender system [156] that proposes a mechanism to suggest Web documents based on textual analysis; or *NewsWeeder* [123] that assists users to filter and select interesting netnews.

In contrast with the previous approach, *collaborative filtering* focuses on the comparison between user's profiles. Generally, a profile is identified by an array of items where each directly experienced item is associated to a feedback rating that denotes the user approval. By means of these ratings the system is able to estimate a similarity value be-

tween users in order to recommend the resources that are much liked by kindred people. Collaborative filtering recommenders have been implemented in a wide range of domains: for example, *Tapestry* [82] is able to filter out off topic posts and select interesting documents in newsgroups; *Ringo* generates suggestions on music albums and artists based on a social filtering mechanism that automates the process of *"word of mouth"* [184]; *GroupLens* [114] helps users of *Usenet* to find fascinating articles; and *MEMOIR* [63] that focuses on finding people with similar tastes and interests.

While the content-based approach is based on an objective analysis of the content, the collaborative filtering technique introduces a *subjective* evaluation of items without forcing the system to represent data in a machine-parsable form. Accordingly, they are completely independent to content representation problems, working well for both complex objects like music songs and movies, and for ordinary textual documents or Web pages.

Since the previous approaches show both benefits and shortcomings, there is not a widely accepted solution that overcomes the other systems. For all that, many proposals try to combine the different perspectives in a *hybrid* technique. Generally, the *content-based* and the *collaborative filtering* approaches are integrated. For example, *Fab* [11] maintains user profiles based on content analysis, and directly compare them to define similar users for collaborative recommendation. Other hybrid proposals can be found in [175, 155, 164, 54].

Furthermore, RS have an inherently *social* bias to bring people together by way of the power of human relationships. Even though social dynamics have been strongly underemphasized in literature, in our work we adopt a *connection-centric* [162] vision to design RS. We address the task of exploiting spontaneous partnerships between users for pushing suggestions to them. Like in the real world, even in virtual communities people can meet each other for conversing about their favorite topics. Hence, a user would trust another user if they have many interests in common: they create a de facto "Word of Mouth" mechanism that helps both to select an item amongst the huge volumes of data that are available on the domain.

6.3 Evaluating Recommendation Systems and algorithms

Evaluating RS and their algorithms is inherently difficult for several reasons. Identifying the best algorithm for a given purpose is not very easy because it is not very clear which attributes should be measured and which metrics should be used for each attribute. The metrics most used to this evaluation are the Root Mean Square Error (RMSE) and Mean Average Error (MAE) when trying to predict users' preferences, and Precision and Recall when trying to recommend items to users.

MAE and RMSE metrics are normally used to measure the accuracy of predicted user ratings. MAE measures the average absolute deviation between a predicted rating and the user's true rating. RMSE, that squares the error before summing it, emphasizes on large errors, that is, it amplifies the contributions of egregious errors, both false positives ("*trust busters*") and false negatives ("*missed opportunities*").

Precision and Recall are definitions originated in the Information Retrieval context, and are defined in terms of a set of retrieved items (e.g. the list of documents produced by a web search engine for a query) and a set of relevant items (e.g. the list of all documents on the Internet that are relevant for a certain topic). Precision can be seen as a measure of exactness or fidelity, whereas Recall is a measure of completeness. In a more formal way, Precision is the fraction of retrieved instances that are relevant:

$$\text{precision} = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{retrieved items}\}|} \tag{6.1}$$

while Recall is the fraction of relevant instances that are successfully retrieved:

$$\text{recall} = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{relevant items}\}|} \tag{6.2}$$

Both precision and recall takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called *precision at n* or *recall at n*.

When dealing with classification tasks, the terms *true positives*, *true negatives*, *false positives*, and *false negatives* compare the results of the classifier under test with trusted external judgments. The terms positive and negative refer to the classifier's prediction (sometimes known as the observation), and the terms true and false refer to whether that prediction corresponds to the external judgment (sometimes known as the expectation). Recall in this context is also referred to as the True Positive Rate, other related measures used in classification include True Negative Rate and Accuracy. True Negative Rate is also called Specificity.

The F-measure is a measure that combines precision and recall as the harmonic mean of both measures:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.3}$$

This measure is also known as the $F1$ measure, because recall and precision are evenly weighted. It is a special case of the general $F\beta$ measure (for non-negative real values of β):

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{6.4}$$

Two other commonly used F measures are the F_2 measure, which weights recall higher than precision, and the $F_{0.5}$ measure, which puts more emphasis on precision than recall.

Different algorithms may be better or worse on different data sets, and this is another element that makes difficult to evaluate recommender systems. Many collaborative filtering algorithms have been designed specifically for data sets where there are many more users than items (e.g., the MovieLens data set has 65,000 users and 5,000 movies). Such algorithms may be entirely inappropriate in a domain where there are many more items than users (e.g., a research paper recommender with thousands of users but tens or hundreds of thousands of articles to recommend). Similar differences exist for ratings density, ratings scale, and other properties of data sets.

6.4 Content-based Recommendation

In content-based recommendation methods, the preference of item s_i for user u_k is estimated based on the preferences r_j assigned by user u_k to items s_j that are "similar" to item s_i. For example, in a movie recommendation application, in order to recommend movies to user u_k, the content-based recommender system tries to understand the commonalities among the movies user u_k has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Then, only the movies that have a high degree of similarity to whatever the user's preferences are would be recommended.

The content-based approach to recommendation has its roots in information retrieval and information filtering research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user profiles that contain information about users' tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly — learned from their transactional behavior over time.

6.5 Classical Collaborative Filtering Algorithms

The survey compiled by Adomavicius et al. [2] presented some of the classical collaborative filtering algorithms. Algorithms for collaborative recommendations can be grouped into two general classes: memory-based(or heuristic-based) and model-based.

6.5.1 Memory-based algorithms

Memory-based algorithms essentially are heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating r for user u and item i is usually computed as an aggregate of the ratings of some other (usually, the N most similar) users for the same item i. To find the most similar users of item i, the method uses a similarity function. The similarity function is essentially a distance measure and is used as a weight, i.e., the more similar users u and u' are, the more weight rating $r_{(u',i)}$ will carry in the prediction of $r_{(u,i)}$. Note that the similarity function is a heuristic artifact that is introduced in order to be able to differentiate between levels of user/item similarity and, at the same time, simplify the rating estimation procedure. Different recommendation applications can use their own user similarity measure.

User-based collaborative filtering

The most common form of collaborative filtering is the neighborhood-based approach (also known as "k Nearest Neighbors" or kNN, for short). The kNN methods identify pairs of items that tend to be rated similarly or like-minded users with similar histories of rating, in order to predict ratings for unobserved users-item pairs.

User-based collaborative filtering predicts a test user's interest in a test item based on rating information from similar user profiles. Each user profile is sorted by its similarity towards the test user's profile. Ratings by more similar users contribute more to predicting the test item rating. The set of similar users can be identified by employing a threshold or selecting top-N. In the top-N case, a set of top-N similar users $N(u_k)$ also called neighbors of user k can be generated according to:

$$N(u_k) = \{u_a \in U | sim(u_k, u_a) > \theta\} \tag{6.5}$$

where $|N(u_k)| = N$. $sim(u_k, u_a)$ is the similarity between users k and a. Cosine similarity and Pearson correlation are popular similarity measures in collaborative filtering. Consequently, the predicted rating $\mathrm{r}(u_k, s_i)$ of test item s_i by test user u_k is computed as

$$\mathrm{r}(u_k, s_i) = \frac{\sum\limits_{u_a \in N(u_k)} \mathrm{r}(u_a, s_i) \cdot \mathrm{sim}(u_k, u_a)}{\sum\limits_{u_a \in N(u_k)} \mathrm{sim}(u_k, u_a)}. \tag{6.6}$$

where $\mathrm{r}(u_a, s_i)$ denote the rating made to item s_i by user u_a.

Existing methods differ in their treatment of unknown ratings from similar users.

Missing ratings can be replaced by a 0 score, which lowers the prediction, or the average rating of that similar user could be used.

Item-based collaborative filtering

Item-based approaches apply the same idea, but use similarity between items instead of users. The unknown rating of a test item by a test user can be predicted by averaging the ratings of other similar items rated by this test user. Again, each item is sorted and re-indexed according to its similarity towards the test item in the user-item matrix, and, ratings from more similar items are weighted stronger.Formally,

$$\mathrm{r}(u_k, s_i) = \frac{\sum_{s_j \in N(s_i)} \mathrm{r}(u_k, s_j) \cdot \mathrm{sim}(s_i, s_j)}{\sum_{s_j \in N(s_i)} \mathrm{sim}(s_i, s_j)}. \tag{6.7}$$

Where item similarity $\mathrm{sim}(s_i, s_j)$ can be approximated by the cosine measure or Pearson correlation. To remove the difference in rating scale between users when computing the similarity, it is used to to adjust the cosine similarity by subtracting the user's average rating from each co-rated pair beforehand. Like the top-N similar users, the neighbors of item s_i, denoted as $N(s_i)$, can be generated according to:

$$N(s_i) = \{s_a \in S | sim(s_i, s_a) > \theta\} \tag{6.8}$$

and where $|N(s_i)| = N$.

6.5.2 Model-based algorithms

As in the case of content-based techniques, the main difference between collaborative model-based techniques and heuristic-based approaches is that the model-based techniques calculate rating predictions based not on some ad hoc heuristic rules, but, rather, based on a model learned from the underlying data using statistical and machine learning techniques.

In contrast to memory-based methods, model-based algorithms use the collection of ratings to learn a model, which is then used to make rating predictions. In order to estimate preferences, there are probabilistic models like cluster models and Bayesian networks. In the first model, like-minded users are clustered into classes. The number of classes and the parameters of the model are learned from the data. The second model represents each item in the domain as a node in a Bayesian network, where the states of each node correspond to the possible rating values for each item. Both the structure of

the network and the conditional probabilities are learned from the data. One limitation of this approach is that each user can be clustered into a single cluster, whereas some recommendation applications may benefit from the ability to cluster users into several categories at once.

6.6 Improved Collaborative Filtering Algorithms

This section will focus on the improvements given to collaborative filtering algorithms since the survey compiled in 2005. These improvements can be summarized in the following techniques used to improve ratings prediction: jointly derived interpolation, matrix factorization and restricted Boltzmann machines.

6.6.1 Jointly Derived Interpolation

The Jointly Derived Neighborhood Interpolation [23] is basically an improvement for the user-based and item-based (kNN) collaborative filtering. Past kNN methods relate items (or users) by various heuristic variants of correlation coefficients, and the correlation coefficients are used directly as interpolation weights when calculating rating scores. Jointly Derived Neighborhood Interpolation is a rigorous alternative to these interpolation weights. It is based on global optimization of a cost function pertaining to all weights simultaneously. This results in an improvement of estimation quality with a minor increase in running time.

Standard neighborhood-based methods have some problems in relation to the way the rating predictions are calculated. First of all, the similarity function which directly defines the interpolation weights, is arbitrary. Various algorithms use different similarity measures, trying to quantify the elusive notion of similarity. These similarity measures represents a previous chosen heuristic, and not necessarily implies in minimizing prediction error. Also, previous neighborhood-based methods do not account for interactions among neighbors. Each similarity between an user u_k and a neighbor $u_a \in N(u_k)$ is computed independently of the content of $N(u_k)$ and the other similarities. Another problem is that by definition, the interpolation weights sum to one, which may cause overfitting. Finally, neighborhood methods may not work well if variability differs substantially among neighbors.

Jointly Derived Neighborhood Interpolation does not use the similarity measure directly as the weight to calculate the prediction, but calculate the best weights to be used using jointly derived interpolation. Given a set of neighbors $N(u_k)$, this method learns the interpolation weights by modeling the relationships between user u_k and its neighbors through a least squares problem. The interpolation weights can then be computed by solving a least squares problem using standard linear equations solvers.

6.6.2 Matrix Factorization

Latent factor models are an approach that tries to explain the ratings by characterizing both items and users on N factors inferred from the ratings patterns. Some of the most successful realizations of latent factor models are based on matrix factorization [206, 116]. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. These methods have become popular by combining good scalability with high prediction accuracy. In addition, they offer good flexibility for modeling various real-life situations.

Matrix factorization is applied in recommender systems by placing in a matrix with one dimension representing users and the other dimension representing the items of interest. The most convenient data to training the model is high-quality explicit feedback, which includes explicit input by users regarding their interest in items. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items.

Such a model is closely related to singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values caused by sparseness in the user-item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting.

Earlier systems relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, inaccurate imputation might distort the data considerably. Hence, more recent works suggested modeling directly the observed ratings only, while avoiding overfitting through a regularized model. To learn the factor vectors, the system minimizes the regularized squared error on the set of known ratings. Two approaches to minimizing the regularized squared are stochastic gradient descent and alternating least squares (ALS).

One strength of matrix factorization is that it allows incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix.

So far, the presented models have been static. In reality, product perception and popularity constantly change as new selections emerge, and we can consider the whole model as not being static. Similarly, customers' inclinations evolve, leading them to

redefine their taste. Thus, the system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions.

The matrix factorization approach lends itself well to modeling temporal effects, which can significantly improve accuracy. Decomposing ratings into distinct terms allows the system to treat different temporal aspects separately. Specifically, the following terms vary over time: item biases, user biases, and user preferences. The more complex factor models, whose descriptions involve more distinct sets of parameters, are more accurate. In fact, the temporal components are particularly important to model as there are significant temporal effects in the data.

Finally, matrix factorization techniques have become a dominant methodology within collaborative filtering recommenders. Experience with datasets such as the Netflix Prize data has shown that they deliver accuracy superior to classical nearest-neighbor techniques. At the same time, they offer a compact memory-efficient model that systems can learn relatively easily. What makes these techniques even more convenient is that models can integrate naturally many crucial aspects of the data, such as multiple forms of feedback, temporal dynamics, and confidence levels.

6.6.3 Restricted Boltzmann Machines

Restricted Boltzmann Machines [172] are stochastic neural networks (networks of neurons where each neuron have some random behavior when activated). It consists of one layer of visible units (neurons) and one layer of hidden units. Units in each layer have no connections between them and are connected to all other units in other layer. Connections between neurons are bidirectional and symmetric. This means that information flows in both directions during the training and during the usage of the network and that weights are the same in both directions.

RBM Network works in the following way: First the network is trained by using some data set and setting the neurons on visible layer to match data points in this data set. After the network is trained we can use it on new unknown data to make classification of the data (this is known as unsupervised learning).

To predict ratings using RBMs, a conditional multinomial distribution is used for modeling each column of the observed "visible" binary rating matrix V and a conditional Bernoulli distribution for modeling "hidden" user features h.

Comparing RBMs with an SVD model, is possible to see that conditional factored RBM slightly outperforms SVD, but not by much. Both models could potentially be improved by more careful tuning of learning rates, batch sizes, and weight-decay. More importantly, the errors made by various versions of the RBM are significantly different from the errors made by various versions of SVD, so linearly combining the predictions of several different versions of each method, using coefficients tuned on the validation

data, produces an error rate significantly lower.

6.6.4 Blending it all together

Each of the previous algorithms produce rating estimations. Linear blending refers to combining multiple estimations. Each estimation is associated to a weight, where each weight is in the range 0 to 1 and the sum of all weights is 1. To find the optimal linear blending consists in finding the optimal weights by letting them range through all valid permutations. Serious attempts at the Netflix challenge required blending results from many algorithms. Blending is an operation that transforms multiple estimates into a single higher accuracy estimate.

Arguably, one of the the Netflix Prize's most convincing lesson is that a disparity of approaches drawn from a diverse crowd is more effective than a smaller number of more powerful techniques. Joining forces allowed teams to incorporate small, outlying techniques that are relatively inconsequential in the big picture, but crucial during the final stages where tweaking matters most.

6.7 Conclusion and Discussion

This chapter was concentrated in the collaborative filtering algorithms to predict user preferences. It was mainly based on papers published during the ongoing of the Netflix Prize, with methods aimed to minimize the root mean square error in a very large dataset with explicit ratings available.

A big part of these works were concentrated in algorithms and strategies trying to predict ratings and to reduce the root mean square error (RMSE) of the predicted ratings, as the competition was concentrated in this metric of evaluation. Some of the algorithms, as jointly derived interpolation and matrix factorization, are specifically designed to work with explicit ratings and to reduce the RMSE. But it was shown also that a better performance with RMSE have also a big impact in precision and recall, which are the metrics used by the information retrieval community.

Finally, it has been shown that predictive accuracy is substantially improved when blending multiple strategies. The experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. Consequently, the best practical solution is an ensemble of many methods.

Chapter 7

A Practical Experience on Collaborative Filtering for Digital Recorders

In order to understand the peculiarities of the collaborative filtering domain, it is of extreme importance to implement and test recommendation strategies that use collaborative filtering approaches in an experimental field. In one of the projects that took place during the PhD course, we had the opportunity to have a practical experience in the complete lifecycle of a recommendation application for personal video recording. In this chapter, we show these experiences, starting from the context analysis and data extraction, to the implementation of different recommendation strategies and their evaluation using a dataset of thousands of active users.

In the wide context of IPTV services, *Digital Video Recorders (DVR)* (a.k.a. *Personal Video Recorder (PVR)*), are hardware or software devices that record digital video to a memory medium, for a further access (e.g., stand-alone set-top-boxes (STB), portable media players, and PC based DVRs). One of the most important exploitations of the DVRs is the *media sharing*: users may want to access to all their personal media resources (e.g., pictures, podcasts, TV programs) from any of their own devices (e.g., laptops, smartphones and so on). For example, Orb[1] is a freeware streaming software that enables users to access remotely their media via Internet. In this chapter, while focusing on the recording of radio and TV broadcasts, we keep in mind the general media sharing domain.

A common marketing view differentiates users in two categories, namely couch potatoes and PC enthusiasts. Although STBs tend to be oriented to couch potatoes and solutions such as Elgato EyeTV, MythTV, InterVideo WinDVR and others are popular amongst PC users, there are a number of common issues and exploitations. One of the

[1] http://www.orb.com

most important exploitations is *media sharing*: users may want to access to all their personal media resources (e.g., pictures, movies, music, Internet podcasts, favorite TV programs) from any of their own devices, including laptops, smartphones, video game consoles, and so on. Of course, they may want to record TV and radio broadcasts or other live events for successive views. For example, Orb[2] is a freeware streaming software that enables users to access remotely their media via Internet. Even if this chapter focuses on the process of recording radio and TV broadcasts, the presented analysis has been conducted keeping in mind the general media sharing domain.

Recording and personalization is changing the way providers insert advertisements during content delivery. The emergence of new behaviors and content consuming trends is forcing advertisers to look for new ways (e.g., a pull model) for spreading their commercials, trying to avoid the fast-forward of pre-recorded videos to skip commercials. A pull model is rapidly attracting the interest of advertisers, because they have to understand which kind of information and content the user wants to search for and receive. In this domain, enthusiasm for *recommender systems* [2] is rapidly growing. In fact, recommender systems can be very useful for selling more targeted items, but also they are powerful retention tools for both old and new customers. When a user is satisfied with suggestions, she dedicates more attention to links and references that are proposed in her personal (web) area. Even if some recommendation engine has been tailored for DVRs (e.g., Neptuny's Content Wise, ReignSoft's Impress), a comprehensive analysis of the peculiarities of this domain is still missing.

First of all, usage data collection is subject to serious privacy concerns. Users are not willing to loose control of data they produce while taking advantage of recommendations and personalized information. Moreover, when a shared device is used (e.g., the television) also family control issues arise, and it can be difficult to provide personalized information. Even for such reasons, costumers protect themselves behind fictitious on line identities, and this is an important challenge for many recommendation algorithms, that need *user profiles* to increase their accuracy.

Another important problem in the IPTV domain as pointed out also in [59, 12] is that, differently from other domains where recommenders can learn from explicit user ratings over content, in DVR based systems we need to operate according to *implicit feedbacks*, such as recordings of a given event, downloading of pre-recorded videos, or - whenever it is possible to collect this information - monitoring if the video has been effectively watched by the user.

A third relevant problem, that has been largely underestimated by previous relevant works on this subject, is the difficulty of discriminating between different items. DVRs usually provide Electronic Program Guides (EPGs) to help the user to record their favorite programs. Unfortunately, in the wider context, EPGs are not reliable to identify elements to recommend, due to many reasons: many TV channels do not respect pro-

[2]http://www.orb.com

grammed schedules, different media and podcasts available within the Web do not make use of a common schedule's format, users are sometimes interested only to single parts of a programmed event, and they will prefer to set up their own timings over a given channel. In general, content description is difficult in the linear TV domain, in opposition with VoD. Broadcasts are announced, but they often lack of structured meta-information. In fact collaborative filtering is usually preferred to content-based recommendation algorithms, even if their execution over collected data is not straightforward.

Recommendation is usually reduced to a prediction problem over the function $r(u_a, e_i)$ that returns the expected rating of element e_i for user u_a. As observed above, we are dealing with an environment where the definitions of all the parameters involved in this function (i.e., user profiles, feedback ratings and elements) are controversial. To our knowledge, this is the first attempt to run collaborative filtering algorithms without inner assumptions.

For this purpose, we start our analysis from an unstructured set of recordings, before performing a data pre-processing phase in order to extract useful information. Hence, as described in Section 7.1, we experiment with a real system where EPGs have not been provided to the user for selecting event timings and where explicit feedbacks were not collected. Then, we describe the procedure for extracting meaningful information from the large and unstructured amount of data provided by the PVR system (Section 7.2) and the recommendation algorithms analyzed in our tests (Section 7.3). Finally, the evaluation of the chosen algorithms in terms of precision and recall is presented in Section 7.4, before drawing conclusions and proposing future research questions in Section 7.5.

7.1 The Experimental Environment

Our analysis is based on real data generated by the *Faucet PVR* system, integrated in a web-based podcasting service named *VCast*. Faucet allows users to record their favorite (Italian) TV and Radio programs, and to further download them into their devices (e.g., iPod, PC, notebook) [44]. The user can set up her own programming and see or download her recordings by the use of a simple web interface. Bringing the ability to record and group into a single feed public and private channels (such as radio and TV recorded programs), Faucet PVR offers a single framework for creating and aggregating personal podcast compilations.

The Faucet PVR produces a very rich and dynamic dataset[3], populated by real users expressing their preferences through the recorded programs. Such a context, however, is characterized by a number of constraints which we had to deal with, in order to be able to perform the analysis on the recommendation algorithms. In particular, the intrinsic dynamism and variability of the recordings, as well as the lack of any permanent event,

[3]The Vcast dataset is publicly available at: `http://secnet.di.unito.it/vcast`

require that a series of pre-processing steps have to be undertaken prior to be able to apply any recommender.

A noticeable property characterizing the context in which we operate is the lack of a well defined programming for recorded contents. Despite several EPG sources do exist, we can not consider them reliable enough to be used for extracting the input information of the recommender engine. Therefore, we decided to opt for a different approach. Exploiting a bottom up approach, we rely on users' knowledge to define the most relevant properties of the events transmitted on the major TVs and radios.
More precisely, the task of defining the parameters related to every recorded transmission is therefore demanded to single users. Since it is their primary interest to make sure that the information inserted in the Faucet PVR are as much precise as possible, we can consider such data reliable enough to be used in the recommendation process. Furthermore, a number of inferences can be deduced from the user activity, and considering also the good popularity of the system, also numerical processing is statistically reliable.

The Faucet PVR involves three different steps to be taken by an user when she is interested in recording an event: the *parameters setting*, the *execution* and the *downloading* of the recorded item. All three steps are performed in different moments, in the aforementioned order. In the parameters setting step, the user chooses a channel, periodicity, name, starting and ending times. This step must be done before the beginning of the program. The execution phase starts at the starting time and finishes at ending time. The downloading step is available only after the recording finishes.

As mentioned above, an event can be periodic: if the user wishes, the system records the desired event in regular intervals. These intervals can be of a week or a day, and in the case of a daily event, the user can choose to skip weekends. Events classified as non-periodic have absolute starting and ending times. However, in the case of periodic events, starting and ending do not represent absolute times, but rather a weekday and daytime (in the case of weekly events) or just a daytime (in the case of daily events). Also, in the case of periodic events, there is one parameter setting step, but the execution step can occur an undefined number of times. After each execution step, a download referring to it is made available. The system limits the number of accumulated recordings to 3 in order to save resources (only the last 3 executions are available to download).

The fact that the dataset includes information about periodicity implies some issues in properly determining the events broadcasted on TV and radio from the amount of recordings made by users. On the other side, it decreases the complexity of calculating recommendations, resulting in an overall improvement in their novelty. In fact, without taking into account the periodicity, as in [95], the recommender has to explicitly ignore periodic elements recently seen by the user, in order to provide a more valuable and accurate recommendation. In our domain, as the periodicity is an intrinsic feature of the recommendable items, we do not have such a constraint, being these elements automatically excluded.

The goal of a recommendation system in the PVR context is to suggest a personalized set of transmissions to the users. However, data coming from the Faucet PVR are not immediately usable to identify events such as the transmissions, but assume the form of unstructured information, which have to be properly processed. In particular, let T be the set of transmissions during a day and t_i be a specific transmission broadcasted on channel c_{t_i}, starting at time b_{t_i} and ending at time e_{t_i}. Then, t_i can be directly used in the recommendation engine, as well as $\forall t \in T$. On the contrary, data collected by the Faucet system (i.e., users' recordings) differ from t_i in the sense that they define a set R of several events r_i with a temporal validity, each referring to a specific event. However, recordings with different timings may refer to the same broadcast: given the pair $r_i, r_j \in R$, they may refer to the same transmission even if $r_i \neq r_j$. Clearly, this property does not hold with discrete and well defined events such as the transmissions.

As well as we can not exploit any EPG to identify the recorded contents, we can not even rely on any information about the specific content of each recording. Indeed, the Faucet PVR does not provide any reference to the type of transmission recorded (e.g., sport program, news, or comedy-movie), nor we can rely on information inserted by users in *title* field, as the insertion of titles and annotations is completely free, and this results in very diversified and, possibly, incorrect descriptions.

As a final observation, broadcasting is characterized by the *expiration* of some events: we can suggest the user to record only future broadcasts, and even if some shows are serialized, the recording of the single episode should be programmed in advance. This phenomenon is (partially) due to copyright management, since the content provider are not willing to authorize service providers to store previously recorded event for further distribution. Nevertheless, recording of a broadcast is still allowed, because it is seen as a single user activity. As a consequence, we have to deal (also) with volatile content, and this differs very much with the VoD domain, that has been exhaustively explored in the context of recommendation systems.

7.2 Data Extraction

Due to the specific domain, we are required to perform a pre-process of the data obtained from the Faucet PVR prior to be able to use such information as input for a recommendation algorithm. This is needed because the Faucet system generates a set of recordings in the continuous domain of *timings*, while a recommender system requires to operate in the discrete domain of *events*.

As a consequence, the first goal that we have to accomplish is the identification of the broadcasted transmissions from the amount of unstructured data resulting from the recording process. This is a multi-step procedure, whose aim is to identify a set of *discrete elements* as the representatives of the broadcasted *events*. Basically, a discrete element

is obtained as the result of the aggregation of several different recordings. A preliminary investigation on the extraction of events from recordings is given in [19].

Let $U = \{u_1, u_2, ..., u_k\}$ be the set of distinct users in the Faucet platform. Each user in set U has recorded some programs in the past and scheduled some for the future. To schedule a program, a user must choose a channel $c \in C$, representing a list of predefined channels, and a periodicity $p \in \{no-repeat, weekly, daily, mon-fri, mon-sat\}$, representing all the possible periodicities allowed in the PVR system. Besides, the user is required to annotate his/her recording with a (possibly) meaningful title.

Let $R = \{r_1, r_2, ..., r_m\}$ be the set of the broadcasted recorded programs. Each element in R (a recording) is a tuple $r_i =< u_i, c_i, p_i, t_i, b_i, f_i >$ set by a user $u_i \in U$ who recorded on the channel $c_i \in C$ with periodicity $p_i \in P$ a program titled t_i with start time b_i and end time f_i. Thus, we can assume that there exists a function mapping every user to her recordings.

The set R is first processed by means of clustering; then, aggregation and collapsing are carried out in sequence on the output of the clustering. The three phases are described in the following.

Clustering Due to the lack of information about the content of each recording, they are clustered wrt the channel, the periodicity and the difference between timings. Specifically, $\forall r_i, r_j \in R | c_{r_i} = c_{r_j} \wedge p_{r_i} = p_{r_j}$ we have that

$$r_i \biguplus r_j \text{ iff } |b_{r_i} - b_{r_j}| < \delta_b \wedge |f_{r_i} - f_{r_j}| < \delta_f,$$

where $\biguplus$ is the clustering function and δ_b, δ_f determine the maximum clustering distance for the start and end times, respectively. The identified clusters contain recordings equal in the channel and periodicity, and similar on the timing. The recording that minimizes the intra-cluster timing distances is elected as the centroid of the cluster. At the end of the clustering, each cluster identifies an event.

Aggregation As the Faucet platform produces new recordings with a hourly frequency, we perform the clustering once a hour obtaining a set of newly generated events. A further step is then required to possibly aggregate similar events, i.e., the new one with those previously created. Such an operation is performed as follows: (1) we compare each element generated with the clustering with the existing events wrt channel, periodicity and timings; (2) if the timings are similar, we correct the properties of existing events with the values of the newly created ones. The list of the users associated to the event is updated accordingly.

Collapsing Similar discrete elements, i.e. with the same channel and periodicity but timings within a fixed range, are merged into a single event. All features of the new

events are computed by means of the values of the collapsed discrete elements. This operation is required basically because events can be created in subsequent moments, by aggregating recordings referring to the same broadcasted transmissions. Due to the high variability of the timings, especially when a new transmission appears, such events slowly and independently converge to more stable timeframes, determining the need of collapsing them into single events.

As a result of the whole process, we obtain a number of events, each being a tuple defined as follows:

$$e_j =< \{u_{e_j}\}, c_j, tl_j, b_j, f_j, p_j >$$

where:

- $\{u_{e_j}\}$ is the list of users who set a recording referring to that event;
- c_j is the channel;
- tl_j is the title chosen among those given by users;
- b_j and f_j are the the starting and ending times respectively;
- $p_j \in \{no-repeat, weekly, daily, mon-fri, mon-sat\}$ is the periodicity.

Let $E = \{e_1, e_2, ..., e_n\}$ the set of the discrete elements, result of the aggregation process. We can define a function $f : U \to E$, that associates each user to a subset of events, i.e., that defines $E_u \subseteq E$ containing the events which user u is associated to.

In Figure 7.1, we can observe the behavior of the system in a one year timeframe, i.e., from June 2008 to June 2009, wrt the number of users, events and recordings. As the number of active recordings and events (b) tends to increase over time, the number of users follows a different, less constant, trend. Specifically, we can notice a considerable increase in the number of users in the system between November 2008 and March 2009. Such a happening implies a consequent raise in the number of recordings, due to the augmented activity in the system. Analogously, the number of events generated by the aggregations of the recordings grows up, although less noticeably if compared to the recordings.

7.3 Recommendation

In our context, we can identify two different approaches to recommendation, depending on the specific target which is considered. As a first attempt, we define a set of events which can be of interest to the majority of the users. In such a case, we are trying to identify the *most frequent* events in the systems, i.e., those programs which have been

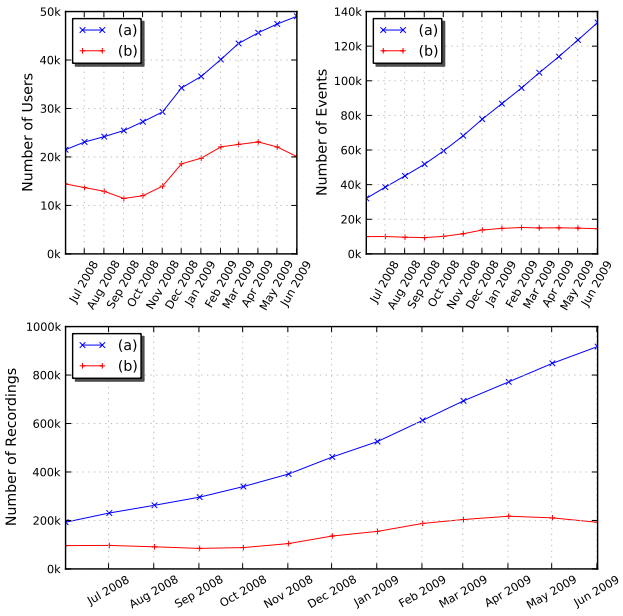


Figure 7.1: Number of users, events and recordings (a) Total and (b) Active in the considered period

recorded by the largest subset of users. This is done by means of a recommendation algorithm which we name *MostPopular*.
A second approach focuses on identifying those events which can be of any interest for a single user of the system. In this case, which we can refer to as user-oriented, the aim of the recommendation algorithm is to suggest items specifically tailored to the user's preferences.

7.3.1 Algorithms Overview

Two well-known recommendation techniques are considered in this work: (1) the memory based *collaborative filtering* approach named k-Nearest Neighbors (kNN) [173]; (2) the model based approach based on the *SVD transform* [174].
Exploiting the basic idea of the *nearest neighbors* approach, we apply both variants of the kNN algorithm: the user-based one [89], by identifying users interested in similar contents; and the item-based approach [64], by focusing on items shared by two or more users. In addition, we also analyze the performance of a variant of the SVD technique based on implicit ratings, presented in [95].

User-based kNN In the *user-based* kNN algorithm, the weight of an element e_i for an user u_k can be defined as:

$$\mathrm{w}(u_k, e_i) = \sum_{u_a \in N(u_k)} \mathrm{r}(u_a, e_i) \cdot \mathrm{c}(u_k, u_a), \tag{7.1}$$

$$\text{where} \quad \mathrm{r}(u_a, e_i) = \begin{cases} 1 & \text{if} \quad e_i \in E_{u_a} \\ 0 & \text{if} \quad e_i \notin E_{u_a} \end{cases}$$

E_{u_a} is the set of elements recorded by user u_a, whilst $N(u_k)$ is the neighborhood of user u_k, limited by considering only the top-N neighbors ordered by user similarity. The different similarity functions are discussed in section 7.3.2. In case the number of neighbors is limited by the chosen similarity function to a number lower than k, we also consider the 2nd-level neighbors, i.e., for each user u_a belonging to $N(u_k)$ we compute $N(u_a)$. The overall set of 1st-level and 2nd-level users is then used to define the users similar to u_k, as previously described.

The coefficient $\mathrm{c}(u_k, u_a)$ represents the neighbor's information weight for user u_k. In most of the kNN-based algorithms [89], the coefficient used is the similarity between u_k and u_a. In other cases [23] the coefficients are calculated using derived interpolation weights. It is worth noting that, in case of considering 2nd-level neighbors, the coefficient $\mathrm{c}(u_k, u_a)$ in eq. (7.1) has to be computed taking into account the similarity between the considered neighbor and further ones. For example, considering user u_k, her neighbor u_a and her 2nd-level neighbor u_b, we have:

$$\mathrm{c}(u_k, u_b) = \mathrm{c}(u_k, u_a) * \mathrm{c}(u_a, u_b),$$

that is a combination of the similarities computed between the neighbors pairs for the considered user.

MostPopular The *MostPopular* algorithm can be also defined by means of eq. (7.1), assuming the number of neighbors unbounded, which implies $N(u_k) = U,\ \forall u_k \in U$; and $\mathrm{c}(u_a, u_b) = 1,\ \forall u_a, u_b \in U$.
The weight of an element e_i to an user u_k is therefore defined as:

$$\mathrm{w}(u_k, e_i) = \sum_{u_a \in U} \mathrm{r}(u_a, e_i) \tag{7.2}$$

After calculating the weight of all elements, they are sorted in descendant order. In the *MostPopular* algorithm, as the set of neighbors is independent of the user, all users receive the same recommended elements, i.e., the most popular elements.

Item-based kNN In the *item-based* kNN algorithm, the weight of an element e_i for an user u_k is defined as:

$$\mathrm{w}(u_k, e_i) = \sum_{e_j \in N(e_i)} \mathrm{r}(u_k, e_j) \cdot \mathrm{c}(e_i, e_j), \tag{7.3}$$

$N(e_i)$ is the set of n items most similar to e_i and recorder by u_k, and $\mathrm{c}(e_i, e_j)$ is the neighbor's information weight wrt item e_i.

Differently from the user-based case, using $k = \infty$ in the item-based approach does not lead to the *Most Popular* set of elements. In fact, the algorithm simply takes all items $e_j \in E_{u_k}$ as neighbors of e_i, making $N(e_i)$ user-dependent.

SVD The Singular Value Decomposition technique analyzed in this work makes use of implicit feedbacks and implements the method proposed in [95]. Specifically, given the observations of the behavior of user u wrt item i, r_{ui}, we can define the user's preference as:

$$p_{ui} = \begin{cases} 1 & \text{if} \quad r_{ui} > 0 \\ 0 & \text{if} \quad r_{ui} = 0 \end{cases}$$

where r_{ui} is set to 1 when u records item i, 0 otherwise.

After associating each user u with a user-factors vector $x_u \in \mathbb{R}^f$ and each item i with an item-factors vector $y_i \in \mathbb{R}^f$, we can predict the unobserved value by user u for item i through the inner product: $x_u^T y_i$. Factors are computed by minimizing the following function [95]:

$$\min_{x*y*} \sum_{u,i} (p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

7.3.2 Computing Neighborhood

In order to provide recommendation on the discrete elements, we have to define a similarity function for grouping similar users/items from which choosing the appropriate elements to recommend. The definition of the similarity is based only on implicit ratings resulting from observing the behavior of users: if she records something, then we assume that she is interested in it; otherwise, we can not infer anything about the interest of the user for that element. We are therefore considering binary feedbacks.

User-to-user Let u and v be two users and E_u, E_v the sets of recorded elements associated to them; we can choose the similarity metric to be used considering several well known measures [132], defined as follows:

- the *Jaccard*'s coefficient: $S(u,v) = \frac{|E_u \cap E_v|}{|E_u \cup E_v|}$;
- the *Dice*'s coefficient: $S(u,v) = \frac{2|E_u \cap E_v|}{|E_u| + |E_v|}$;
- the *Cosine* similarity: $S(u,v) = \frac{E_u^T E_v}{||E_u|| \cdot ||E_v||}$;
- the *Matching* similarity: $S(u,v) = |E_u \cap E_v|$.

Note that the Jaccard, Dice and Cosine similarities have values in the range $[0,1]$ while the Matching similarity has values in the range $[0,\infty]$. In addition, since both Jaccard and Dice are monotonic functions, we expect a similar behavior in the computed neighborhood, i.e., N_u are the same in both cases.

Then, $\forall u$, we can then compute the subset $N_u \subseteq U$ of *neighbors* of user u. A user v such that $E_v \cap E_u \neq \emptyset$ is thus defined as a neighbor of u. Starting from the neighborhood of u, similarity with u is computed for each pair $< u, v >$ such that $v \in N_u$.

Finally, if $S(u,v) > 0$, we consider u similar to v, i.e., there is an arc connecting them in the similarity network. The value $S(u,v)$ is used to weight such a relation, therefore determining a similarity order among the neighborhood of u.

Item-to-item The similarity among items is based on the same measures already mentioned before, yet redefined as follows by considering two items e, f and their sets of users U_e, U_f who recorded them:

- the *Jaccard*'s coefficient: $S(e,f) = \frac{|U_e \cap U_f|}{|U_e \cup U_f|}$;
- the *Dice*'s coefficient: $S(e,f) = \frac{2|U_e \cap U_f|}{|U_e| + |U_f|}$;
- the *Cosine* similarity: $S(e,f) = \frac{U_e^T U_f}{||U_e|| \cdot ||U_f||}$;
- the *Matching* similarity: $S(e,f) = |U_e \cap U_f|$.

$\forall e \in E$ we can compute the subset $N_e \subseteq E$ of *neighbors* of item e. An item f such that $U_e \cap U_f \neq \emptyset$ is thus defined as a neighbor of e. Starting from the neighborhood of e, similarity with e is computed for each pair $< e, f >$ such that $f \in N_e$.

We can then decide whether a couple of items is similar or not. Items e is considered similar to f, i.e., there is an arc connecting them in the similarity network, if $S(e,f) > 0$. A similarity order among the neighbors of e is thus determined.

7.4 Experimental Results

Our evaluation is based on trying to measure how accurate is each recommendation algorithm in predicting the elements that users would program. This is achieved by computing precision and recall on the predicted items. The more accurate is this prediction, the more valuable elements are recommended. It is important to underline that we do not consider any feedback related to the user's interest in the recommended items, but we only focus on the prediction ability of the algorithms analyzed.

To start evaluating a recommendation algorithm, we fix an arbitrary time t after the data collection started and before the data collection stopped. The value of t should be carefully chosen not to be too close to the data collection start, since we do not have sufficient data to make good predictions. Also, the time t should not be close to the end of data collection, because we need a good amount of data to make the verification if the algorithm was able to predict it. As the data collection started January 23rd 2008 and ended November 19th 2009, we choose values of t varying from June 1st 2008 to June 1st 2009.

7.4.1 Metrics

Given the set E of events in our framework, we define the following subsets:

- $A(t) \subset E$, active events at time t ($b_j > t$);
- $R(u,t) \subset E$, events recorded by user u before time t;
- $V(u,t) \subset A(t)$, events recorded by user u after time t;
- $Rec(u,t) \subset A(t)$, events recommended to user u at time t.

It is important to notice that $A(t)$ is also the set of all elements suitable for recommendation at time t. The aim of our recommendation algorithms is to predict which events are in $V(u,t)$. For that, for each user, the algorithms associate a weight $w(u,s)$ to each element $s \in A(t)$ which represents, from the recommender's point of view, how much reliable is the fact that $s \in V(u,t)$. Furthermore, a recommendation algorithm use only the information in

$$\bigcup_{u \in U} R(u,t), \quad \text{with} \quad R(u,t) \cap V(u,t) = \emptyset.$$

To recommend items to users, we use only the top n elements in $Rec(u,t)$, ordered by weight. This set is represented as $Rec(n,u,t)$: it is the set of top n items recommended to user u at time t. The *precision* and *recall* at time t are computed as the average of all users' precision and recall values computed using the top n recommended elements [174]. Finally, we compute the system precision and recall at different times, and calculate the system overall precision and recall as the average of it.

As in [95], also in our context precision measures are not very meaningful, because we do not have feedbacks regarding the user's interest in those items which have not been considered (i.e., not programmed, nor downloaded). On the contrary, recall-oriented measures are more suitable. Infact, we can assume that e_i is of any interest for user u only if $e_i \in V(u,t)$, otherwise no assumption on user's interests can be made. Anyway, for sake of completeness, we also report the analysis of precision values.

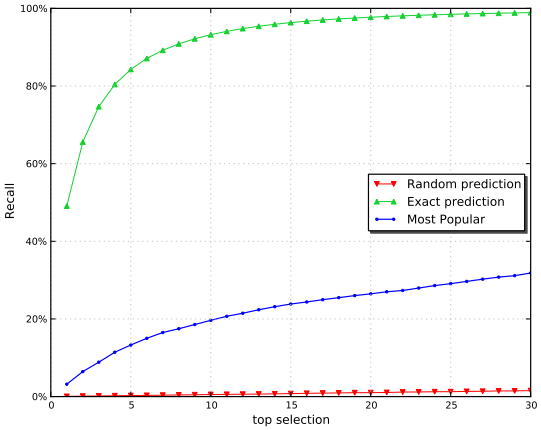
7.4.2 Evaluation

As a first step in the evaluation, we attempt to define the specific upper and lower bounds which characterize the recommendations in the PVR domain. In particular, we compare the *MostPopular recommender*, which identifies the most frequent elements among all users (Section 7.3.1), with the following two algorithms: (1) a *random recommender*, which simply chooses n random elements among those of $A(t)$, defining the lower bound to our experiment; (2) an *exact predictor recommender*, which has knowledge about the elements in $V(u,t)$, thus yielding to the best possible results and defining the upper bound.
The results are depicted in Figure 7.2(a), which clearly shows that, as expected, even the *MostPopular* algorithm can easily outperform a random predictor. However, it is still far from being able to make a complete prediction of all the elements, especially when the considered top n are just few items.

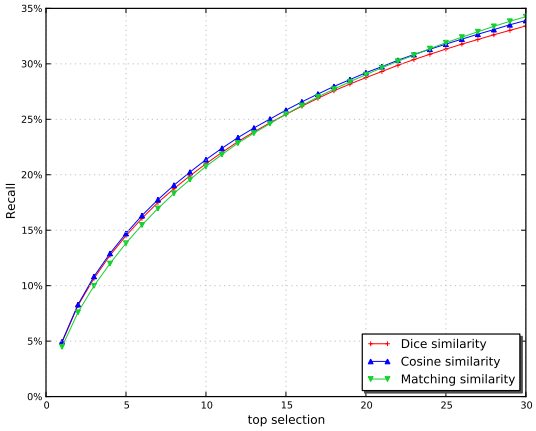
The second step in our evaluation is to study how different similarity functions affect the results of user-based kNN recommendation algorithms. We can observe from Figure 7.2(b) that, in case of the user-based algorithm, all chosen similarities show nearly the same performances.
On the contrary, the Matching similarity considerably outperforms the other measures when it comes to the item-based algorithm, as displayed in Figure 7.2(c). Again, both Dice and Jaccard show a very similar behavior, being superior to the Cosine metric already when more than 5 elements are recommended. In both Figures 7.2(b) and 7.2(c), the Jaccard similarity is not shown being almost identical to the Dice.

Another step in our evaluation is to find the consequences of adding second-level neighbors in the neighborhood of user-based kNN recommendation algorithms. In Figure 7.3(a), we can observe that increasing the number of first level neighbors (when it is lower than k) by adding the second level ones implies a better performance of the algorithms. In this example, we used Dice similarity and $k = 300$, however the results are similar when applying second-level neighbors to other similarities.

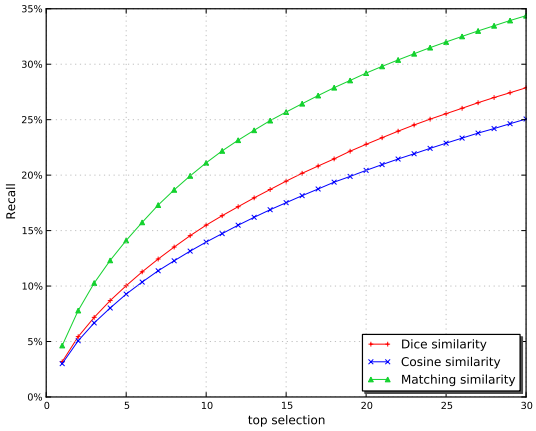
In the next tests, we try to find an optimal value for k in the user-based kNN algorithm. Figure 7.3(b) shows the results of kNN user-based with $k \in \{100, 300, 500, 700, 2000\}$, and the *MostPopular* recommender. We used Dice similarity, but the results are similar with other similarity functions. In addition, in Figure 7.3(b), as well as in Figure 7.3(c),



(a) Upper and lower bounds for recall



(b) Comparison between similarity functions in user-based kNN



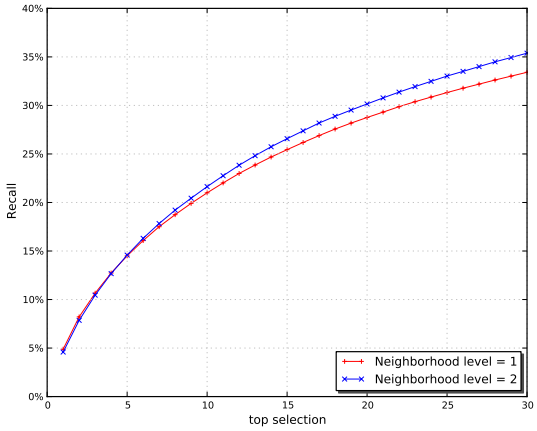
(c) Comparison between similarity functions in item-based kNN

Figure 7.2: Comparison between recommenders and similarity functions.

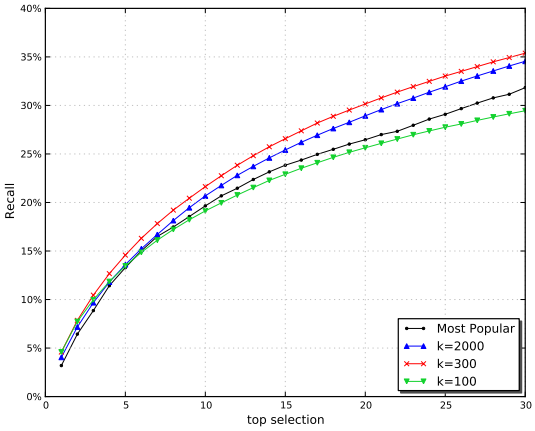
we omit the values of $k = \{500, 700\}$ since the results are very similar to the case of $k = 300$.

We can observe that a value $k = 100$ is not sufficient to outperform the *MostPopular* algorithm, due to the lower value of the recall. On the other side, a very high number of neighbors allows to perform better than the *MostPopular*. However, we could notice that, already with $k = 2000$, the algorithm starts to converge to the *MostPopular*, characterized by an unbounded number of neighbors by definition. Therefore, we can consider the range $[100, 2000]$ as suitable to identify the optimal value for k.

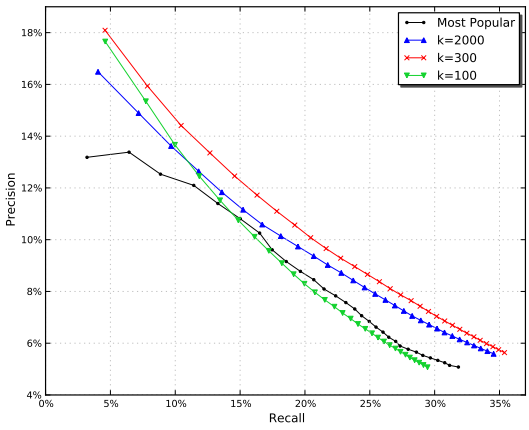
For this purpose, we test the values $k = \{300, 500, 700\}$, obtaining very similar performance. Considering the top 10 recommended elements, we can achieve better results for $k = 300$, whilst $k = 500$ is more suitable when taking the top 11 to 30 elements. As in most cases 10 elements are sufficient for a recommendation, $k = 300$ is a good



(a) Comparison of one-level and two-level neighborhoods for user-based kNN



(b) Recall for user-based kNN



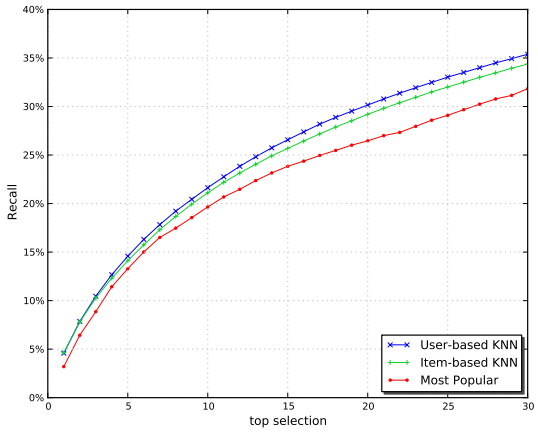
(c) Precision vs Recall for user-based kNN

Figure 7.3: Neighborhoods comparison, precision and recall for user-based kNN.

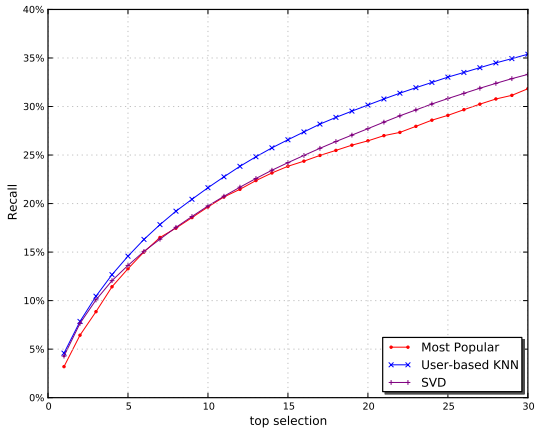
compromise between the ability of providing valuable recommendations and the resource consumption in calculating the neighborhood.

To better observe the trend of both recall and precision, Figure 7.3(c) shows the two values combined. Again, $k = 300$ performs better if we take only the top 10 recommended elements, as it also yields to good results in terms of precision. Considering more than 10 recommendations, it would seem appropriate to increase the number of neighbors to 500, as the results for precision and recall are slightly better. However, the overall behavior of the algorithm is almost identical with k in the range $\{300, 700\}$. Nevertheless, the above mentioned considerations regarding the superior performance of the kNN algorithm with $k = 300$ in terms of computation requirements still apply when we take into account the precision metric.

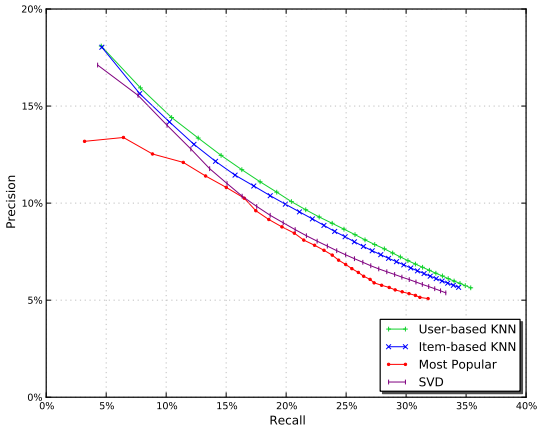
An interesting comparison among the three kNN algorithms analyzed, i.e., user-based, item-based and *MostPopular*, is depicted in Figure 7.4(a). We can observe that the latter is clearly outperformed by the other two algorithms in terms of recall, especially when more than 7 recommended items are considered. Between the item-based and the user-based version of the kNN, the latter performs slightly better, although the gap is mostly noticeable when more than 15 items are recommended. In general, item-based algorithms tend to perform better because usually the number of items is considerably lower than the users [173]. Such a property does not hold in our domain, hence making the user-based version superior in terms of recall, as we initially expected.



(a) Recall for kNN ($k = 300$) wrt MostPopular



(b) Recall for SVD wrt kNN ($k = 300$) User-based and MostPopular



(c) Precision vs Recall for kNN ($k = 300$), SVD and MostPopular

Figure 7.4: Precision and recall for the analyzed algorithms.

A final experiment is attempted in order to measure the behavior of the SVD approach wrt the performance of the kNN method. The implementation of the SVD algorithms

described in Section 7.3.1 is tested with different parameters, with the purpose of identifying the more suitable ones in our context. In particular, we try different sizes for user-factors and item-factors vectors, values for the λ parameter and number of training steps.
Results are depicted in Figure 7.4(b). The best prediction is obtained with 100 features, $\lambda = 500$, $\alpha = 40$ and 15 training steps. However, the behavior of the latent factor model based on SVD in the analyzed context is worse if compared to a neighborhood model such as kNN. As the reader can notice, the kNN user-based is able to substantially outperform the SVD technique, whose results in terms of recall are slightly better than those of the MostPopular algorithm only when a considerable number of items are recommended.

Similarly, results related to the precision of the recommendations (Figure 7.4(c)) show an analogous behavior of the kNN algorithms wrt SVD, with the *Most Popular* being considerably less precise than others. Also, the user-based algorithm shows to be more precise than the item-based in determining the recommendable items, for the same reason previously mentioned considering recall.

It could appear surprising that the prediction performance of the SVD recommender is worse than other techniques, as this algorithm normally performs better in several other contexts [174, 115, 95]. We believe that the motivations for such an unusual behavior reside in the dataset characteristics. In particular, a reason might be identified in the so called *cold start problem*, whose effects involve users, items and communities [178].
In our context, the cold start problem is particularly noticeable with items and is due to the lack of relevant feedbacks when a new event first appears in the system. Such an issue is made worse by the fact that items to recommend are generally new ones, i.e. those events having a starting time in the future. This property holds for no-repeat events as well as for repetive ones (the starting time is updated according to their periodicity). So, events whose starting time has passed are no longer elegible for recommendation.

The fact that recommendations are affected by the cold start problem is one key factor that may influence SVD performance, as this algorithm needs support of user's preferences to perform well. On the contrary, a neighborhood-based approach such as kNN appears to better deal with newly introduced items, as also reported in [59].

7.5 Conclusion

Many services are converging towards a media sharing model, that let the user access different media files from any device he/she owns. This brings to a growing difficulty for a system to discriminate between different media type and content consuming trends. In fact, we can expect that users will set up starting and ending times of live events (that can be TV broadcasts, as well as Internet streaming events) without caring of datetimes announced in EPGs or advertisement messages. Moreover, it can be difficult to access to

a useful description of the consumed content, and as a consequence collaborative filtering tools are important means for providing useful suggestions. Furthermore, users are often unwilling to rate explicitly the content they consumed, and other kind of trivial implicit feedbacks (e.g., the suggested item has been purchased) are not applicable to any domain.

We experimented with a real digital recording service, and, accordingly to the above mentioned restrictions, we decided to run our analysis under the strongest assumptions: no EPGs are available, users can set up timings as well as channels, explicit feedbacks are not collected, and so on. In addition, the intrinsically dynamic nature of the analyzed PVR domain, which determines a continuous process of creation and deletion of events and a consequent amplification of the cold start problem, makes such a context sensibly different in terms of recommendation if compared to those where items have no time validity (i.e., netflix, movielens, etc.).

Despite these constraints, our results showed that neighborhood based strategies, such as kNN, can return in good prediction accuracy and, if correctly tuned, they can outperform SVD-based techniques as well as *most popular* strategies, that dangerously leverage the phenomenon of many users concentrated on very few relevant events.
Finally, there is evidence that digital recorders differ from other interest based services, because factors other than personal tastes might influence the user's behavior and the success of a recommendation engine. In fact, the direct social influence of friends and the volatile nature of events are supposed to be relevant factors in causing a user to schedule a recording.
In our opinion, a possibile future research direction could be indeed the identification and study of those social factors which affect user's behavior in systems characterized by high dynamism and short lifetime of items.

Chapter 8

Epidemic Collaborative Filtering in Opportunistic Networks

In this Chapter, we present an *epidemic collaborative filtering* approach that allows a mobile device to identify similar neighbors from *opportunistic communications* and exchange information in a *selective way*. Collected information is used to incrementally refine locally calculated recommendations, without the needing of interacting with a remote server or accessing the Internet. Using a simulated environment, we show how recommendation accuracies experimented in the mobile domain using *experimental datasets* converge to values that are comparable to the best ones of the centralized scenario; moreover, we empirically prove how selective spreading strategies significantly reduce the cold start problem and show performance similar to epidemic strategies.

The increasing proliferation of wireless communications in various devices (i.e., smart phones, gaming consoles, music players, devices within a vehicle, and so on) has led to a growing interest in *Ad Hoc Networks* as well as *Opportunistic Networks*. This interest is also motivated by the consideration that nowadays information is available (almost) everywhere and anyway, coming from different sources, such as television broadcasters, radio stations, and the Web; as a consequence, users produce, consume, and share digital resources, creating new self-feeding cycles.

As computing is becoming mobile and pervasive, devices with such wireless communication capabilities are going to be embedded in everyday objects and in the physical environment. Not surprisingly, users are increasingly more demanding in terms of services even when a fixed architecture is not available in the proximity [126]. This is the case of a *delay-tolerant network* (DTN) consisting of isolated devices and networks that can be occasionally linked. Such communications are realized following the mobility behavior of the entities that carry these devices. Additionally, consider the many cases when a portable device carried by a mobile user cannot access a fixed infrastructure, e.g., in a suburban area, in the subway, in a car. Even if a hotspot is in the proximity, it can deny

the access because it is protected against unauthorized, undesired or free accesses.

In such a scenario, *opportunistic networks* [159] have gathered much attention from the scientific community, since they are an evolution of MANETs where mobile nodes can exchange information even if there is not a route connecting them. A particular emphasis has been given to routing optimization problems, because the underlying assumption is that a device will be willing to cooperate in the distribution of information that is "useful" according to some definition of utility. For example, if a message has a specific destination, the sender can opportunistically select a given device in the proximity as relaying node, and the utility is defined as the likeliness to bring the message as closer as possible to the destination.

Such proposals do not target the replacement of infrastructure-based solutions, or, even, the Internet. Instead, they are meant to provide an alternative when an infrastructure is not available, so that discontinuously connected devices can keep working and forwarding information. In this context, the understanding of human mobility patterns can be used to implement new approaches to optimize data diffusion schemes. This motivated part of our planned scientific activity, that involved the study of human mobility models, and validation of such models with real-world data collected during experimental deployments, using the SocioPatterns framework [47].

Opportunistic networking can be used in the sense previously described, in order to deliver messages from point to point, but it can also be used to disseminate information that is useful not only to a single user, but to communities of people with common interests. The dissemination of information that fits in the preference of communities of interest belongs to the *Recommendation Systems* context, and more specifically to the *Collaborative Filtering* domain.

If opportunistic communications are used instead of broadcasting or random walks, then the problem turns to understand which information should be exchanged when a device is in the proximity. Interest-driven data diffusion schemes [45] and selective exchanges between "neighbors" in a similarity network [180], can epidemically and efficiently spread ratings and preferences information. Of course, evaluation metrics must be considered as well. For instance, [45] introduces a *network infection* ratio, an *utility* function and an *efficiency* estimation, measuring respectively the average fraction of "infected" nodes per messages, the average number of interested nodes infected by a message, and the trade-off between message delivery and resource consumption.

In order to simulate and evaluate such algorithms and protocols, we use empirical data collected from the SocioPatterns project, augmented by data collected at the same time by the Live Social Semantics (LSS) experiment [5], as described in Chapter 2. This experiment is focused in the integration of heterogeneous data sources, such as real-world face-to-face contacts, on-line friendships and shared interests (both explicitly stated or implicitly inferred from metadata). Such data is used to create different definitions of similarities between users in the SocioPatterns experiments, and at the same time define

the interest of each user in the information being disseminated. Based on such similarities and interests, we propose *spreading strategies that use collaborative filtering techniques to selectively disseminate information*. Such algorithms are evaluated in terms of *utility* and *efficiency*, by measuring parameters commonly used in the collaborative filtering domain.

To evaluate the proposed opportunistic information spreading strategies, we use an hypothetical application for tag recommendation, with tags extracted from the given empirical datasets. In fact, the dynamics of user behavior stemming from social annotation systems follows a collective exploration of the semantic space, and this exploration unveils a complex network structure [46]. By exploiting the social structure created by such social annotation systems, it was shown that tag recommendation systems can effectively support users of social bookmarking systems in assigning tags to their bookmarks [100]. The same approach can be used even for movie recommendation [188].

In the forthcoming sections, we focus on collaborative filtering dealing with self-organizing communities, host mobility, wireless access, and opportunistic communications. In such a domain, knowledge representation and users profiling can be hard; remote servers can be often unreachable due to client mobility; and feedback ratings collected during random connections to other users' ad-hoc devices can be useless, because of natural differences between human beings. Our approach is based on so called *Similarity Networks*, and on a system called *MobHinter*, that epidemically spreads recommendations through spontaneous similarities between users.

Main results of this study are two fold: firstly we show how to reach comparable recommendation accuracies in the mobile domain as well as in the centralized scenario; secondly, we propose epidemic collaborative strategies that can reduce rapidly and realistically the cold start problem just by exchanging information filtered accordingly to user preferences.

8.1 Related Work

One common characteristic of the recommender systems described in Chapter 6 is the use of a *centralized* architecture in which the information about items and ratings is stored in a central database that has complete knowledge of the domain. In the last years, the wide adoption of fully decentralized platforms such as peer-to-peer content distribution systems or the availability of mobile devices with innovative multimedia features and adequate quality of service have completely subverted these preconditions. In fact, a mobile scenario is characterized by the lack of a complete knowledge of the application domain or user's profiles, along with the absence of a central repository in which ratings are stored and from which they are retrieved. However, to authors knowledge, there has been few experience in designing collaborative filtering systems in

distributed environments such as mobile ad-hoc networks.

A first attempt to deal with a decentralized environment is proposed in [189] where products and services are suggested in a marketplace populated by mobile customers. In the environment assistant agents act as peers serving the mobile customers. When a neighbor is looking for suggestions it broadcasts a query containing a vector with its votes on products and recommendations. When a peer receives the voting vector it calculates the proximity with the cached previous messages: if the proximity is higher than a threshold, then the peer send back the cached voting vector. If the proximity measure is lower, the query voting vector is broadcasted further to other peers.

The effectiveness of fully decentralized collaborative filtering techniques has been of particular interest for the opportunistic networking community. For example, in [22], RFID devices are used as passive storing devices to aggregate information about the visiting patterns of users, and no transmission is done between RFID tags or between users. In this work, the performance of the decentralized approach is evaluated by comparing with centralized strategies.

Protocols used for interest dissemination in wireless sensor networks are discussed in [135]. This approach propagates data through broadcast communication by estimating the number of neighbors, and packets are routed towards a sink node by exploiting hop count topologies. Algorithms for interest dissemination are used in this approach, and data propagation involves one-to-all communication which is initiated and governed by the sink. In fact, they observe that simply broadcasting data may be expensive, and node connectivity is not ensured at all times, so a probabilistic approach is integrated with the forward data dissemination scheme.

Solutions using fully decentralized opportunistic approaches are also available for recommending new contacts to mobile phone users [10]. This approach does not assume any centralized coordination, and it does not take advantage of physical presence detection to eventually exchange information. Instead of using physical proximity to transfer data, they propose an approach that opportunistically uses residual space in *Short Message Service* (SMS) messages occasionally exchanged between users. This solution collects and processes information that is available in the mobile phones and that is exchanged transparently and opportunistically, preserving user's privacy.

The idea of exploiting information regarding social ties between network nodes was also explored in [137]. They validate the intuition that people with common interests tend to meet more frequently, and that our movements are guided by our interests, by analyzing the experimental traces collected during the Infocom 2006 conference. They found that the correlation of meeting frequency and similarity of interest profiles is considerably high when focusing on longer meetings. These results support the conclusion that this intuition can be used as the basic mechanism of social-aware, stateless forwarding protocols, as well as opportunistic information spreading.

Finally the work of Splinder et al. [62] deals specifically with collaborative mobile ad-hoc networks. They introduce the notion of *shared social context* in order to help their distributed collaborative filtering system to establish similarity relationship between co-present users. The assumption behind their proposal is that if two users attend the same event, as arts festival events, it is likely that they have similar interests. Thus, without computing prior similarity, predictions are based on the set of users sharing at least one item, that is an event/location they have consumed during the same period of time. However this approach does not work in context where users are copresent by chance, such as on the subway, or on the bus, etc. and they are not supposed to share similar interests. What our proposal shares with the work of Splinder et al. [62] is the notion of collaborative filtering protocol for mobile environments. According to them, the protocol must respect the following requirements: (1) computation and storage must work in decentralised settings, therefore (2) local computation must be kept simple and the required data small, (3) the protocol must rely on ad-hoc peer-to-peer connections only and must be delay tolerant, (4) data exchange has to be short and to consume little bandwidth, (5) user interaction should be minimal.

8.2 Similarity Networks

In order to model our domain in a more formal way, we have a set of users $U = \{u_1, u_2, \ldots, u_n\}$, and a set of objects $S = \{s_1, s_2, \ldots, s_l\}$ (e.g., pictures, videos, songs, favorite restaurants, ...). In many domains, we may assume that a *preference* function is defined: $\text{pref} : U \times S \to P$, where P can be a set of allowed rating values (e.g., $R = \{1, \ldots, 5\}$), or simply $\{0, 1\}$ if binary ratings are considered. We assume a bijection between users and nodes in the system, hence the user u_i denotes both the i-th node and the i-th user. We define $\mathcal{P}(S)$ as the power set of S, i.e. the set of all subsets of S. The function $f : U \to \mathcal{P}(S)$ maps users to objects of the entire collection (i.e., $\bigcup_{i=1}^{n} f(u_i) = S$). In other words, $f(u_i)$ is the set of items user u_i is related to; for instance, in a social annotation system $f(u_i)$ is the set of *tags* used by u_i, as well as for a video streaming service $f(u_i)$ is the set of channels or podcasts u_i is subscribed to. In the peculiar domain that we investigate in our work, an object is mapped to a user iff such a user expressed, in any way, a preference to the given object; i.e., $\forall u_i \in U \wedge \forall s_k \in S : \text{rate}(u_i, s_k)$ is defined $\iff s_k \in f(u_i)$.

To take advantage of the power of social relationships, we need to evaluate the *similarity* among users. For this purpose, we first introduce the *similarity function* $sim : U^2 \to [0, 1]$, that returns a similarity value for any pair of users.

This function is subject to the following properties:

1. $sim(u_i, u_j)$ must be computable using information that u_i and u_j can exchange each other without interacting with a third party;

2. $sim(u_i, u_i) = 1$;
3. $sim(u_i, u_j) = 0$ means that, due to their knowledge of each other, u_i and u_j have nothing in common;
4. if $sim(u_i, u_j) > sim(u_i, u_k)$ then u_i is more similar to u_j than to u_k. This does not imply, in general, any relationships between u_j and u_k.

Similarities can be estimated taking into account many parameters: resources in common, similar behaviors, comparable ratings assigned to items, and so on.

Now we introduce the idea of *"Similarity Network"* that is represented by a graph $G^\theta = (U, E)$ such as:

$$e_{ij} \in E \Leftrightarrow sim(u_i, u_j) > 0. \tag{8.1}$$

In the remaining of the section, we will discuss how similarity graphs can be exploited for selective spreading of information for recommendation systems in the mobility domain.

8.3 Collaborative Filtering via Opportunistic Communications

The mobility scenario has its own peculiar issues. Even if we can assume an "omniscient" server that stores and maintains all the knowledge provided for in our community, we need to grant a high level of autonomy to each host. In fact, in order to make our approach as much general as possible, we allow the user to walk around, and to *meet* other users following different mobility patterns. Only devices in the proximity can exchange information, and by using empirical datasets, this approach provides a natural filtering based on the social contexts, even if we are not assuming anything about the reasons why they are there in that moment, in contrast with other authors (e.g., [62]).

By allowing only short-range opportunistic communication, we fix a lower bound for cases when the mobile devices are unable to connect to the remote server (e.g., it cannot access the broadband Internet), as well as cases when the user has a device with short-range wireless facilities to connect ad-hoc to other mobile devices in the proximity.

When a device is able to connect to the Internet, the user can access the rest of the community, and publish his/her own data, profile and preferences. Searches and lookups can be managed by a central directory service, or in a peer-to-peer style (e.g., by flooding or by means of a DHT based overlay network). If each node u accesses the status of its similarity network's neighbors, then it can calculate the predicted preference over an object s using one of the many user-based collaborative filtering algorithms known in the literature.

Given a similarity graph G^{θ}, let $N(u)$ be the set of neighbors of u. This means that if v is a neighbor of u, then they are similar according the chosen similarity function. Furthermore, node u may store all neighbors' referred items and ratings. Hence, if we define the list of preferences of u as a set of pairs $\langle s_{id}, p_{us} \rangle$, where s_{id} is s's object identifier and p_{us} is u's preference for s, then the user can keep the following information:

$$Fp_u = \{\forall s \in f(u) : \langle s_{id}, p_{us} \rangle)\} \tag{8.2}$$

In this scenario we can say that the state of node u is very limited, because many data can be retrieved run-time contacting on-line neighbors. Therefore, u needs to keep references to its neighbors $N(u)$, and their corresponding lists of preferences Fp_u.

In our experiments, in order to generate a set of recommended items to user u, we used a slightly modified version of the well known user based k-nearest neighbor algorithm [177] to calculate the weight w of an object s:

$$\mathrm{w}(u, s) = \sum_{v \in N(u)} \mathrm{pref}(v, s) \cdot \mathrm{sim}(u, v) \tag{8.3}$$

where $sim(u, v)$ is the similarity between users u and v, $N(u)$ is the set of top k users most similar to u and $\mathrm{pref}(v, s)$ is v's preference for item s.

Of course other approaches can be used to improve recommendation accuracy, but this is out of the scope of this work, since MobHinter is just an epidemic approach that can be applied even when only ad-hoc communications are available during a relatively big period of time.

Without opportunistic information exchanging, if mobile devices cannot access the Internet and they cannot contact directly their neighbors in the similarity network, then the recommendation simply cannot be calculated. Our hypothesis is that each node u can start with an empty set of neighbors, i.e., $N(u) = \emptyset$. We suppose that for each neighbor $v \in N(u)$, the node stores also its list of ratings Fp_v. Moreover, u maintains a list of *known hosts* Kn_u, which is a cache of nodes encountered during u's walks. Again, for each node $v \in Kn_u$, list Fp_v is stored as well.

When a device u finds another device v in the proximity, a handshake phase is started: u and v can exchange their identities, their preferences lists Fp_u and Fp_v, their neighbors lists $N(u)$ and $N(v)$, or their known hosts lists Kn_u and Kn_v. Then, both ad-hoc nodes can calculate their similarity in autonomy.

We assume that user u knows its similarity value with each neighbor $v \in N(u)$. If it is not stored, it can be calculated in time. Each time u receives information about a previously unknown user v, his neighbors lists $N(u)$ can be updated. If $|N(u)| < k$ (where k is the maximum number of neighbors), then v is added to $N(u)$. Otherwise, if $|N(u)| = k$, then we must find the user $x \in N(u)$ with the lowest similarity and, if $sim(u, v) > sim(u, x)$, then we remove x from $N(u)$ and add v on his place.

Trivially, we can foresee an approximately long *cold start* phase. Nevertheless, MobHinter exploits social networking style *Word-of-Mouth* strategies in order to opportunistically spread neighborhood information across the proximity network. In fact, if node u is not a neighbor of v, it can happen that a similarity is found with one of his/her neighbors in $N(v)$, or with one of known nodes in cache Kn_v. It is important to observe that, after the first ad-hoc interaction, few or no other ad-hoc interactions are needed. This is very important because a node can be out of sight in a few seconds after handshake. All the similarities and recommendations are calculated in autonomy with the available data. In the following, we list three different spreading strategies. For improving readability, only actions at u side are described, because v behaves symmetrically.

KNOWN u receive from v only their identity Fp_v. Additionally, $\mathrm{sim}(u, v)$ is calculated and $N(u)$ is updated accordingly to the value found for the similarity.

NEIGH u receive from v their identity Fp_v and neighbors list $N(v)$. Of course, $\mathrm{sim}(u, v)$ is calculated and $N(u)$ is updated accordingly to the value found for the similarity; moreover, $\forall z \in N(v)$, $\mathrm{sim}(u, z)$ is calculated, and $N(u)$ is updated accordingly. This is possible because we assumed that each node keeps each neighbor's ratings Fp_z.

EPI u receive from v their identity Fr_v and known hosts list Kn_v. It updates its Kn_u with all users found in Kn_v: $Kn_u = Kn_u \cup Kn_v$. Additionally, u looks for neighbors in Kn_v. When a new neighbor z is found, Kn_u is updated. It is possible to evaluate similarities because we assumed that each node keeps each known host's ratings Fp_z.

The Epidemic spreading strategy (EPI standing for *epidemic*) is the most aggressive spreading approach. It floods all information found to all contacted users, meaning that the list Kn_u can grow quickly and indefinitely. KNOWN is the least aggressive spreading approach, as no information is stored in Kn_u. However, in the KNOWN approach, the information is disseminated indiscriminately. The NEIGH exchange strategy uses selective data dissemination. Exchanged data is limited to $N(u)$, and the stored information is limited to the size of $N(u)$.

Recommendations are calculated again after each meeting using Equation 8.3, considering all the neighbors in $N(u)$ and their preferences. The three strategies are increasingly more expensive in terms of bandwidth consumption, state storage, and computational resources usage. We want to find if selective dissemination strategies are scalable in terms of the number of iterations, and if cold start mitigation worths the overhead. Such a simulative evaluation is provided in the next section, while a discussion on other practical issues is given in Section 8.5.

8.4 Experimental Analysis

In order to simulate and analyze the opportunistic recommendation strategies, we will assume a hypothetical application for tag recommendation. The main goal of this application is to recommend a set of tags not yet associated to a user, based on the tags already associated to that user. To simulate this application, we first need a set of associations $\langle user, tag \rangle$, that from now on is called *Tag Cloud*. Henceforth we will describe how the tag cloud was extracted from the available datasets to simulate our hypothetical application over the contact traces.

We will use the contact trace gathered using the SocioPatterns platform during two different experiments: the Hypertext 2009 Conference and the ESWC Conference. As we already mentioned in the description of the datasets in Chapter 2, the participants of these conferences are associated with metadata. We only use the part of the metadata that associates the name of the participant with the badge identifier worn by him. So, we first extract from the datasets the $\langle id, name \rangle$ associations, where *id* is the identifier of the badge, and *name* is the name of participant who worn the badge during the conference.

Having the name of each participant associated with the RFID badges, we create the tag cloud by extracting data from a public bibliography database, associated with each participant. The DBLP (Digital Bibliography and Library Project) [128] is a computer science bibliography website hosted at Universität Trier, in Germany. We found that most of the participants in the experiments have associated bibliography in DBLP. We select the publications associated with each participant, and from this list we extract the set of words from the publications' titles.

At this point, we have a set of words associated with each participant. This set of words contains also lots of non important words, like articles and prepositions. In order to remove the most common and non important words, we create a list of *stop words*, that contains the list of approximately 6 hundred most common words in English - among them articles, prepositions and pronouns. We filter out these words from the set of words associated with the participants.

The resulting set is, at this point, a good source for creating the Tag Cloud. There is still one problem to solve: the different word inflections. For example, one of the most used words is 'ontology', with more than 30 appearances; but the word 'ontologies' is also used a lot, and it contributes to the same idea of 'ontology'. So, we apply a *lemmatization* process, by grouping together the different inflected forms of a word so they can be analyzed as a single item. The *stemming* expression is also used in this context; it is the process for reducing inflected words to their *stem*. In order to achieve this, we used the *Wordnet lemmatizer*, a lemmatizer implementation from the Natural Language Toolkit [27] that uses WordNet's *morphy* function [139].

The resulting list of lemmatized words is used to construct the Tag Cloud. Using the

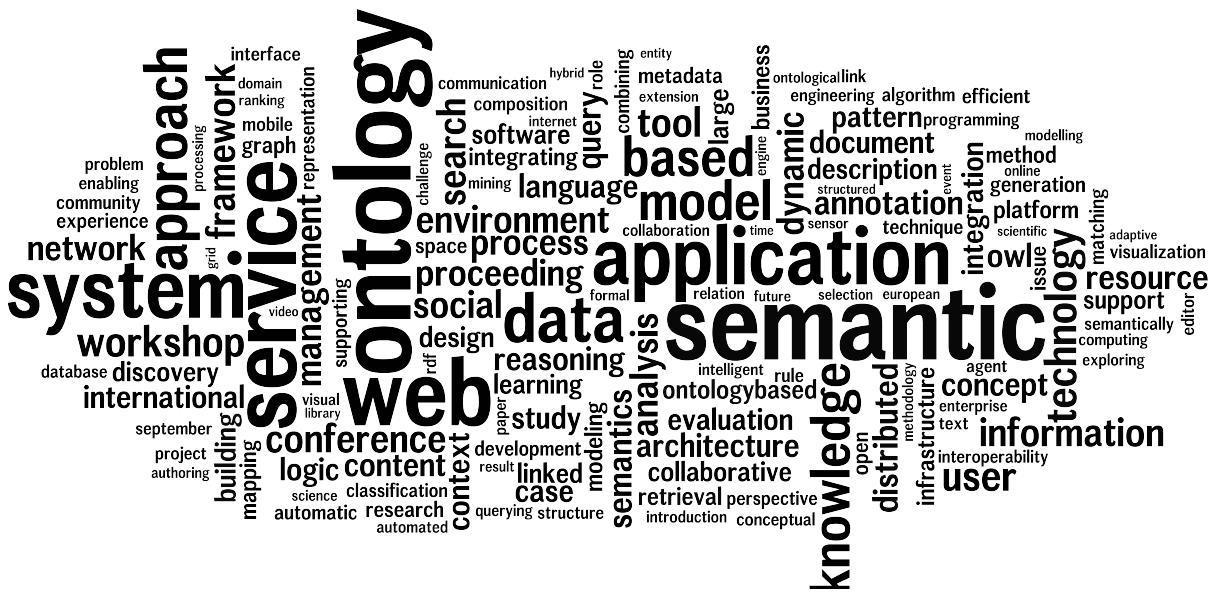


Figure 8.1: The ESWC 2009 tag cloud

HT2009 conference dataset, 799 tags were extracted from 57 users with valid metadata at DBLP. Using the ESWC conference dataset, 1285 tags were extracted from 94 users with valid metadata at DBLP. The most representative tags in the ESWC conference are shown in figure 8.1, where the word size is proportional to the number of times it appears in the extracted tag cloud. The tag cloud for the HT2009 conference is shown in figure 8.2. The final tag cloud is a set of tuples $\langle user, tag \rangle$ with all associations between users and tags. The tag cloud layout was produced using the online application *Wordle* (`http://www.wordle.net/`) [196].

Figure 8.3 shows that, as it has been observed in other tagging datasets, the extracted tag clouds have power law distribution of the number of users associated with each tag. The extracted dataset shows the same regularities that chan be observed in most datasets generated by written text [8], and the power law distribution of word frequencies is one of these regularities.

Our hypothetical application will recommend a set of tags based on the user's associated tags. We can define different recommendation strategies to be evaluated. For example, a naive strategy is accomplished by recommending tags that are popular amongst all users, given that the recommended tags are not yet associated with the user. A more complex strategy that uses collaborative filtering will get the k users most similar to the given user and create a recommendation list based on data gathered from these k neighbors, as defined in Equation 8.3.

Figure 8.2: The Hypertext 2009 conference tag cloud

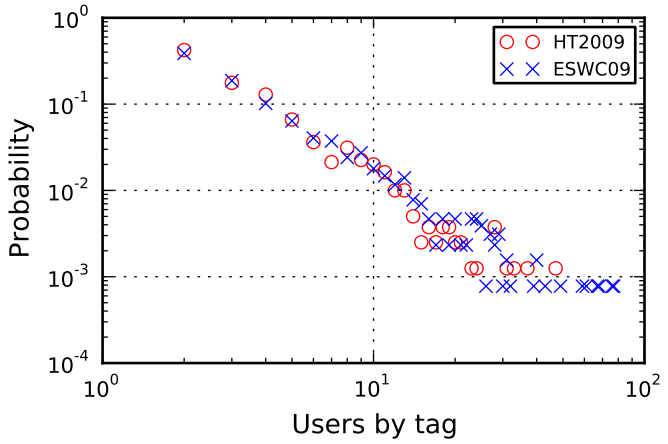


Figure 8.3: The distribution of the number of users associated with each tag

Most probably, the best results that could be acquired for a given recommendation strategy is reached when the system has complete knowledge about all objects, users and user's tags. We don't have users explicitly rating tags, so we will use an implicit binary rating: 1 if the tag is associated with the user, 0 otherwise. Devices using our application moves into a disconnected environment, where epidemic dissemination of information is critical before accurate recommendation of tags can be triggered to the user. This process is simulated through a set of iterations that use experimental data to mimic meetings between users.

The evaluation is made of two phases:

1. Run the adopted recommender strategies in a simulated fully connected scenario, in order to produce a reference model and to find optimal recommendation results;
2. Run the same recommendation strategies in a fully disconnected environment where only ad-hoc meetings and opportunistic message exchanging are allowed.

Aims of this evaluation is three-fold: (1) to estimate the average simulation time before converging to optimal values, accordingly to the different message exchanging strategies described in the previous section; (2) to estimate overhead w.r.t. the capacity of a modern mobile device; and (3) to analyze the scalability of the approach.

8.4.1 Evaluation Methodology

In order to evaluate the accuracy of the recommendation strategies, we exploit two well known accuracy metrics: *Precision* and *Recall*. We chose to not use the *Mean Absolute Error* (MAE) and the *Root Mean Square Error* (RMSE) because these metrics are normally used to measure the accuracy of predicted user ratings, and do not really measure top-N performance [58].

To make an evaluation that includes all the data set, we used the *k-fold cross-validation*, with $k = 10$. In this evaluation, the ratings set is partitioned in 10 samples. Of the 10 samples, a single sample is retained as the *testing set*, and the remaining 9 samples are used as *training set*. This validation is repeated 10 times, ensuring that all ratings are used for testing. The results are then averaged to produce a single estimation.

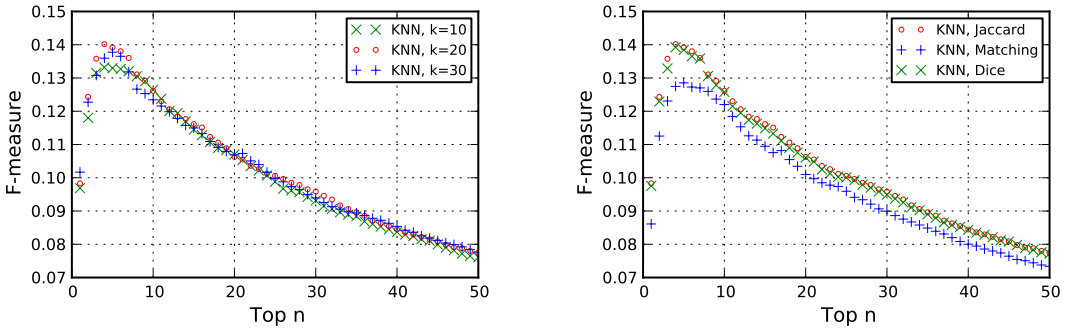
Our evaluation is based on measuring, in each simulation step, how accurate is each recommendation strategy in predicting the tags associated to users in the test set. We use as training set only information that is available to the users after contacting other users in each step. The accuracy evaluation is achieved by computing precision and recall on the test set. The more accurate is the recommender on predicting the tags in the test set, the more valuable is the set of recommended tags. It is important to underline that we do not consider any feedback related to the user's interest in the recommended tags, but we only focus on the prediction ability of the strategies analyzed.

8.4.2 Results in the Reference Scenario

In the reference scenario, we evaluated the precision and recall of tags for the entire dataset, with the assumption that all nodes could find its complete set of neighbors and run the tag recommendation using different approaches.

Our goal in this step is to compare the results of a collaborative filtering approach with other naïve approaches using all dataset. As we focus on the k-Nearest Neighbors (kNN) collaborative filtering approach, we must first search for the optimal parameters for the kNN. The parameters to optimize are the number of neighbors (k) and the similarity function.

Figure 8.4(a) shows the performance results for different values of k in the ESWC dataset. The number of neighbors is an important parameter of the algorithm, as it impacts in the processing time to calculate the list of recommended tags, and in the required space to store information about other users. It shows that $k = 10$, $k = 20$ and $k = 30$ have similar performances, with better results going to $k = 20$. $k = 30$ shows similar results to $k = 20$, but from now on we choose $k = 20$ in the following simulations as it is a good tradeoff between performance and processing load: it results in acceptable performance with a limited number of neighbors.

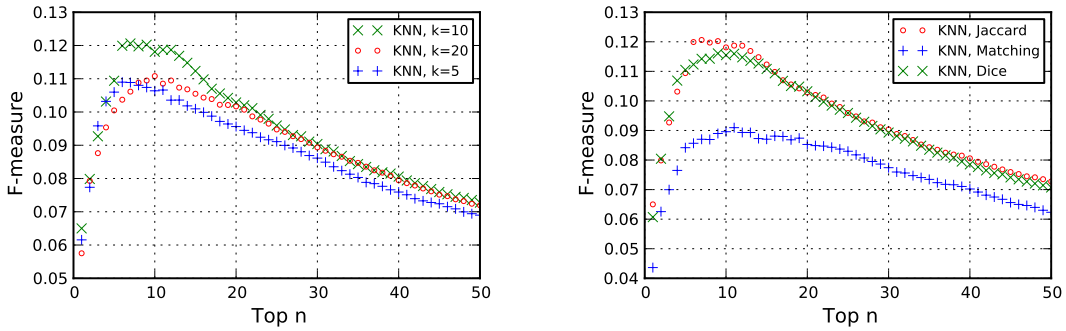


(a) Comparing the number of neighbors (b) Comparing different similarity functions

Figure 8.4: Comparing the F-measure performance for (a) different number of neighbors and (b) different similiarity functions in the ESWC dataset.

Figure 8.4(b) shows the performance of the kNN approach using $k = 20$ but with different similarity functions to calculate user similarity. As the best results were obtained using the Jaccard similarity, from now on we will use this similarity in all cases.

Figure 8.5(a) shows the performance results for different values of k using the HT2009 dataset. Differently from the ESWC dataset, it shows that $k = 5$, $k = 10$ and $k = 20$ have similar performances, with better results going to $k = 10$. Figure 8.5(b) shows the performance of the kNN approach using $k = 10$ but with different similarity functions to



(a) Comparing the number of neighbors

(b) Comparing different similarity functions

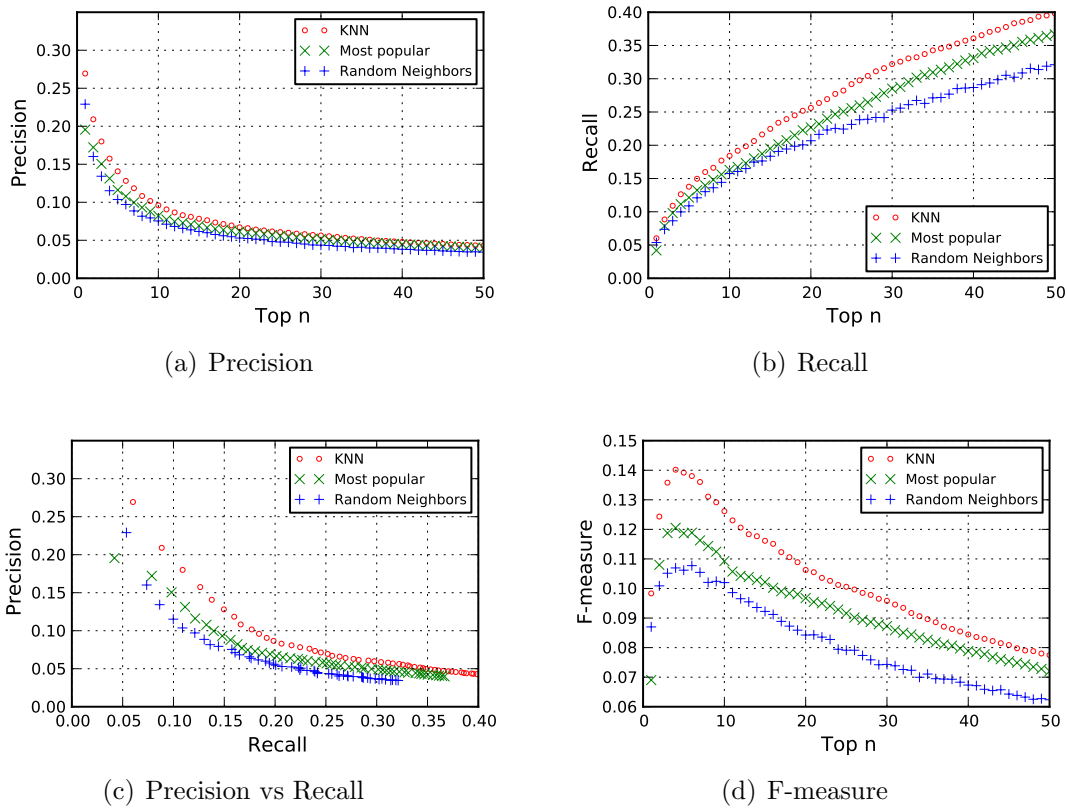
Figure 8.5: Comparing the F-measure performance for (a) different number of neighbors and (b) different similiarity functions in the HT2009 dataset.

calculate user similarity. As the best results were obtained using the Jaccard similarity, from now on we will also use this similarity in all HT2009 cases.

At this point, we individuated the best parameters for the kNN in a reference scenario, where all users have complete knowledge about other users. We will use the Jaccard similarity to calculate user similarity. The optimal value for the number of neighbors is different in the two datasets: for the ESWC dataset we will use $k = 20$ and for the HT2009 dataset we will use $k = 10$.

The next step is to compare the kNN approach with other approaches. The first approach to compare is a naïve approach, where the system recommends tags that are popular amongst all users, given that the recommended tags are not yet associated with the user. This corresponds to the Most Popular approach, as referred in Figures 8.6 and 8.7. The other approach is similar to the k-nearest neighbors collaborative filtering algorithm, but it takes k random users to create the user neighborhood. The evaluation of this approach is important to show that the selection of similar users as neighbors is important for a good recommendation performance, and just collecting information from randomly chosen nodes is not sufficient for a good performance.

Figure 8.6 and 8.7 compares the precision and recall values using all three different approaches, using both datasets. It is important to note that choosing random neighbors produce a bad performance comparing to the original kNN approach, and it shows the importance of choosing similar users to recommend tags. It also shows that the Most Popular strategy does not have a performance similar to kNN, even in a reference scenario where all users know everything about other users. These results reinforce the idea that using a collaborative filtering algorithm is important to reach a good recommendation. In this figure we show also the F-measure, because it considers both precision and recall to compute the score, and better shows the performance difference between each approach.



(a) Precision

(b) Recall

(c) Precision vs Recall

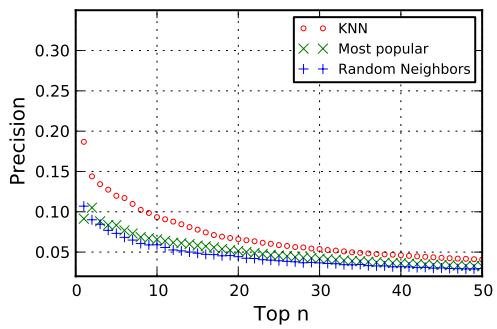
(d) F-measure

Figure 8.6: Comparing the performance of three different recommendation strategies: k-Nearest Neighbors (kNN), Most Popular and Random Neighbors. The dataset used for this comparison was extracted from the ESWC Conference.

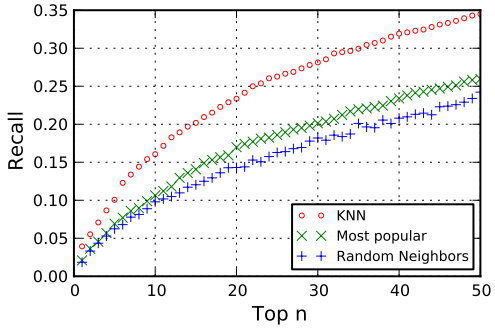
8.4.3 Simulation of Ad-Hoc Scenario

In the previous section we individuated the parameters used to arrive to the best recommendation performance in the reference scenario for a given user-based neighbor recommender system. In the following, we will present a simulative framework in order to model the dynamics of a population of mobile devices that interact each others through ad-hoc meetings. In this way, they share their local knowledge for producing personalized advices.

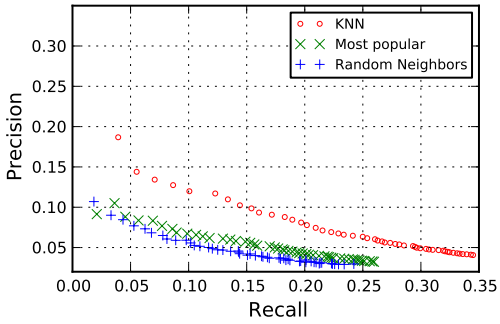
We assume that each device starts without any a priori knowledge about the domain or the population of users. Afterward, it starts a *discovery phase* during which the device explores the world looking for new nodes. During this exploratory phase we suppose that it does not have access to the Internet or to any other remote server. Its only knowledge comes from its contacts with other devices. The device is able to exchange data only through ad-hoc and opportunistic communications.



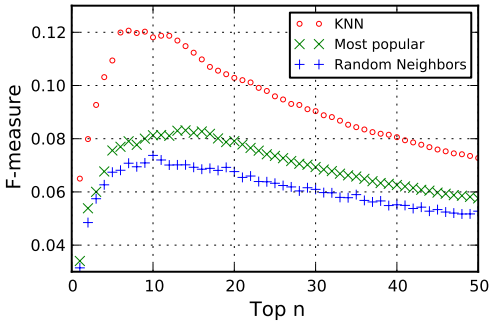
(a) Precision



(b) Recall



(c) Precision vs Recall



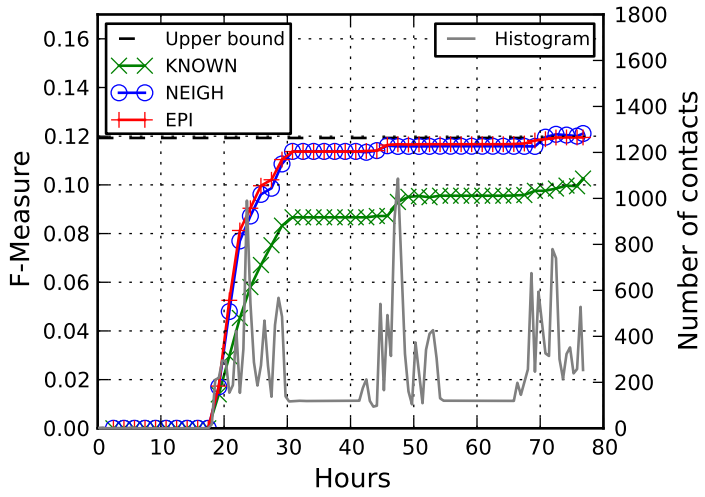
(d) F-measure

Figure 8.7: Comparing the performance of three different recommendation strategies: k-Nearest Neighbors (kNN), Most Popular and Random Neighbors. The dataset used for this comparison was extracted from the Hypertext 2009 Conference.

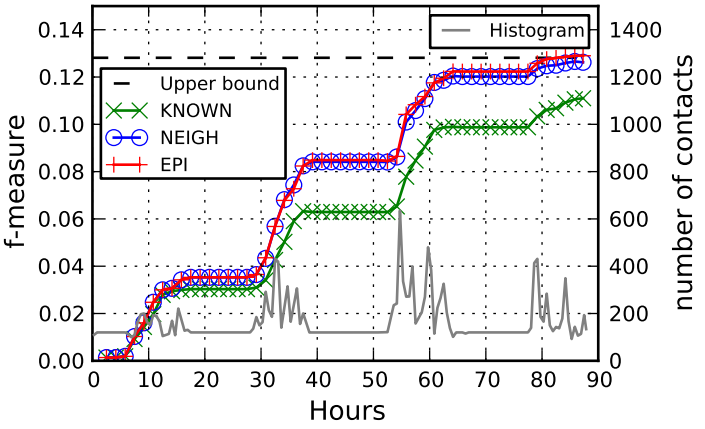
The simulation is composed by a sequence of *iterative steps* that model the contacts between users. The contacts are generated by the experimental contact traces collected in the Hypertext 2009 Conference and in the ESWC Conference, as described in Chapter 2.

The state of a user u_i, i.e., the amount of data the user has to store, is composed by two sets: (1) the set of tags associated with that user and the set of neighbors U_i. Of course, we have that $U_i = \emptyset$ at the beginning of the simulation. When users u_i and u_j meet each other, they can run one the three selected strategies: *KNOWN*, *PREF* and *EPI*.

In our simulation, we performed a certain number of iterative steps and, at each iteration, we computed for all *KNOWN*, *PREF* and *EPI* strategies the precision and recall accuracy metrics, under the evaluation framework discussed in Section 8.4.1. In all the simulations, we used the Jaccard similarity to calculate user similarity.



(a) HT2009



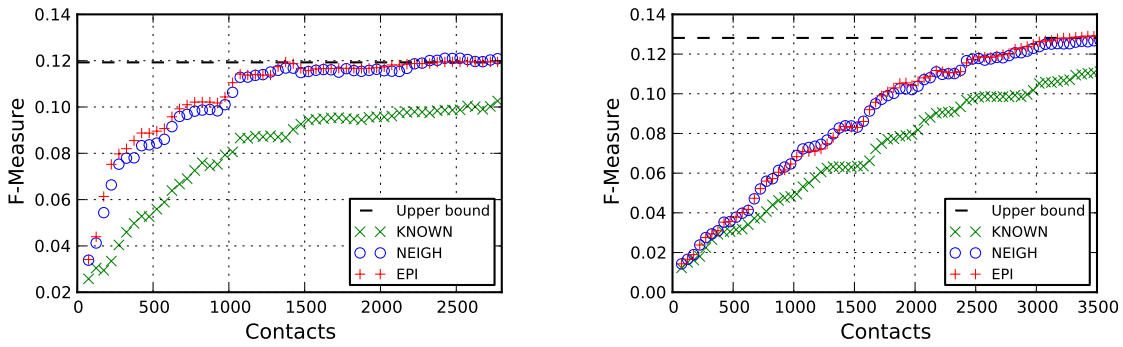
(b) ESWC

Figure 8.8: Evolution of F-Measure estimation over time in the Ad-Hoc scenario compared with F-Measure estimation in the reference scenario for top 10 recommended items. The gray line histogram shows the number of contacts in each hour. The dataset used for this comparison was extracted from the Hypertext 2009 Conference and from the ESWC Conference.

First of all, we make a parallel with the evolution of the recommendation performance and the the contact histogram. Figure 8.8(a) shows the evolution in the f-measure value for the *KNOWN*, *PREF* and *EPI* strategies. In the same graph we plot the the contact histogram. In this figure it is possible to see that the f-measure value boost its value when there are more contacts. As it turns out, the influence of contact density and burstiness is

high in both datasets; there are short intervals of fast increasing in the recommendation performance, and long intervals in which the measured performance is maintained.

It is possible to visualize the ongoing performance of the exchanging strategies from another perspective, by measuring the performance in function of the number of contacts. By focusing in the number of contacts which occurred in order to obtain a given recommendation performance, we are in fact changing the clock used to observe it. This changing of perspective is done in a similar way in Chapter 5, in order to observe the delivery time of messages. Consequently, we remove from the observation the intervals in which there are no contacts between users, since the recommendation performance cannot increase during these intervals. At the same time, peaks in which lots of contacts happen in a short period are flattened, allowing to observe what happens during these peaks in more detail.



(a) HT2009 (b) ESWC

Figure 8.9: Evolution of F-Measure estimation in the Ad-Hoc scenario for three different approaches of data spreadin: EPI, NEIGH and KNOWN. The approaches are compared with F-Measure estimation in the reference scenario for top 10 recommended items. The number of contacts is used as the observation clock. The dataset used for this comparison was extracted from the Hypertext 2009 Conference and from the ESWC Conference.

In Figure 8.9, the number of contacts is used as observation clock. This visualization is interesting as it shows out the number of contacts that should happen in order to reach a given level of dissemination. It can also be interpreted as the real energy spent by the nodes in order to reach the given recommendation performance. Both figures show a consistent better performance for both EPI and NEIGH approaches, compared to the KNOWN approach.

As it turns out, the influence of bursts are relatively mitigated in the evaluation when using the number of contacts as observation clock. In fact, Figures 8.9(a) and 8.9(b) show a smoother evolution in the recommendation performance. Intuitively, we can note that the recommendation performance is not a function of time, but it is a function of the number of contacts between users. As the number of contacts increases

in time, the recommendation performance also increases, but the real mechanism that boosts the recommendation performance is the number of contacts. Therefore, following this intuitive conclusion, in the next visualizations we will represent the recommendation performance in function of the number of contacts.

The process of neighbors discovery is shown in Figure 8.10, again by using the number of contacts as the observation clock. At each step, we computed the average number of users stored by each user, i.e., $avg(|N(u) \cup Kn_u|)$. While the required storage space grows slowly using the KNOWN strategy, the content of $N(u)$ is also slowly populated. Using the EPI strategy the content of $N(u)$ is quickly populated, but the required storage space grows quickly and is limited only by the number of users in the dataset. Using the NEIGH strategy, in contrast to the others, limits the storage space to the number of neighbors in $N(u)$ and allows quick population of $N(u)$.

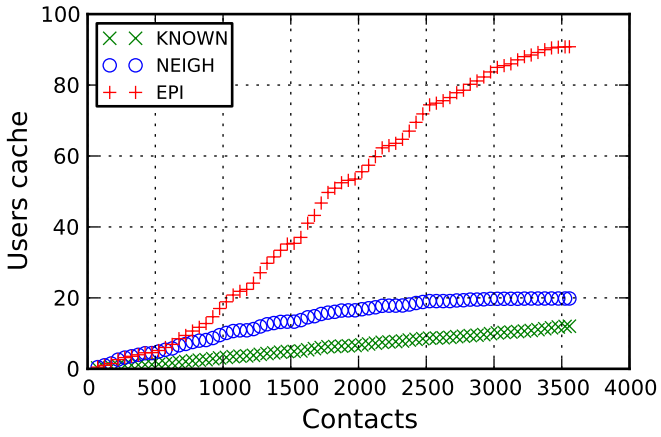


Figure 8.10: Evolution of the number of users in the cache in the Ad-Hoc scenario for three different approaches of data spreading: Epidemic spreading (EPI), Selective dissemination (NEIGH) and Unselective dissemination (KNOWN).

Very interestingly, even if there is a little difference between the recommendation performance of *EPI* and *NEIGH* strategies, this is not comparable to the difference observed between *NEIGH* and *KNOWN* strategies. Our interpretation is that, as observed in many social networking analysis, "the neighbors of my neighbors are likely to be my neighbors". Hence, epidemic distribution of ratings is greatly enhanced if information are spread through similarity links. Conversely, known users that are not recognized as neighbors, have limited probability to be neighbors of my neighbors, especially in a sparse dataset like the extracted tag cloud. Nevertheless, if users exchange and store all information in an epidemic approach, trust busters and missed opportunities can be reduced by fewer steps.

8.4.4 Overhead and Scalability considerations

Considering a simulative scenario using the HT2009 dataset, Figure 8.9 show also how many ad-hoc communications are needed before predicting approximately accurate recommendations. Hence, in order to understand if the overhead is viable with modern mobile devices in terms of their available capacity and to evaluate scalability of the system, we need to estimate step by step the average device's status in terms of memory occupancy. Focusing again on the HT2009 dataset, we can define $|u_i|$, $|s_k|$ as, respectively, the size in bit of a user's identifier and an object identifier. Binary preferences are given by the presence or absence of the couple $\langle u_i, s_k \rangle$ in the preferences list. Let's initially suppose that both user and object identifiers have a size of 32 bits.

The status of a MobHinter node u_i is made of the following elements: Fr_i, U_i, and Kn_i. The neighbors list and known hosts cache grows differently accordingly to the adopted strategy. It is important to recall that for each node u_j in $U(i)$ or in Kn_i, we need to store the node's identifier (i.e., $|u_j|$ bit), and ratings list Fr_j (i.e., $\forall(s_k, r_{jk}) \in Fr_j$, the device consumes $|s_k| \cdot |rate(u_i, s_k)|$). Considering an average of 20 preferences per user, each user in the cache consumes approximately 80 bytes of space. Focusing on the iterative steps where the recommendation performance converges to the values obtained in the reference scenario, for each different strategies, we have an average status for device as in Table 8.1.

Number of contacts	*KNOWN*	*NEIGH*	*EPI*
2500	1200	1600	4320

Table 8.1: Status overhead behaviors (in bytes)

Of course, *KNOWN* is the less expensive in terms of memory consumption, but the reader should observe that when the recommendation performance converges to the values obtained in the reference scenario, the average status of MobHinter node behaving accordingly to *EPI* is about 4.3 KB, which is only limited by the number of users in the experiment. If we consider a dataset with thousands of users, the required space grows quickly to hundreds of KB, and to exchange this amount of data at each encounter starts to be problematic. Using the *KNOWN* approach, the required space grows slowly but also indefinitely. The only approach that is guaranteed to be scalable is the *NEIGH* approach, which limits the cache to the number of users in the neighborhood.

8.4.5 Impact of correlation between user similarity and contact frequency

It is well known that *people with common interests tend to meet more frequently.* Analysis of experimental traces collected during the Infocom 2006 conference showed that the

correlation of meeting frequency and similarity of interest profiles is considerably high when focusing on longer meetings [137]. Therefore, the statistical correlation between user similarity and contact frequency could in great part justify the advantage of the *NEIGH* strategy over the other dissemination strategies.

But in what extent this characteristic of human behavior impacts in the selective spreading of information? In this section, we try to measure this extent by shuffling the users' metadata. Each user in the proximity network will be associated with metadata randomly chosen among the other users. To achieve this, we start by creating a shuffled list of user metadata. For each user, instead of using its associated metadata, we use the metadata from the shuffled list.

By decoupling the user from its metadata, each user's most frequent contacts will happen with random users with independent interests. In Figure 8.11 we compare the results of information spreading in the original dataset, where people with common interests tend to meet more frequently, with the shuffled dataset, where the contacts are independent of user's interest. In this experiment, the information spreading is done in a selective way, by using the *NEIGH* strategy of data exchange, as explained in Section 8.3.

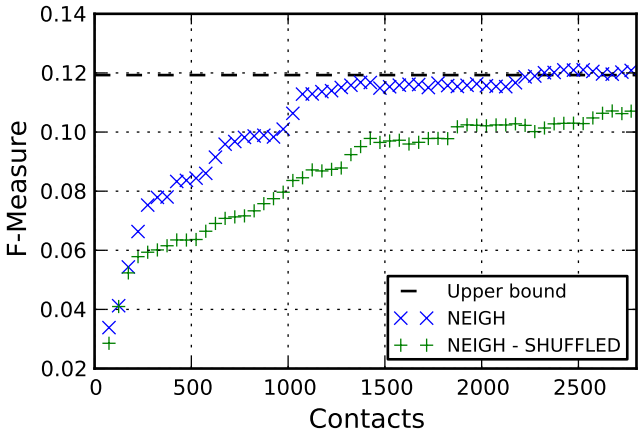


Figure 8.11: Evolution of F-Measure estimation in the Ad-Hoc scenario using the NEIGH approach, by associating user's metadata with the original metadata and with shuffled metadata. The approaches are compared with F-Measure estimation in the reference scenario for top 10 recommended items. The number of contacts is used as the observation clock. The dataset used for this comparison was extracted from the Hypertext 2009 Conference.

The results show that the statistical correlation between user similarity and contact frequency has a great impact in the recommendation performance, and by using a dataset with randomly associated metadata the performance decreases significantly. By showing these results, we also want to emphasize the importance of collecting user's metadata

when collecting user traces, in order to achieve more realistic simulations and more reliable results.

8.4.6 Comparison with synthetic contact traces

This simulation part is composed by the same sequence of iterative steps used in previous sections to model the contacts between users. The contacts are generated by three different contact traces. The first contact trace is the SocioPatterns dataset generated during the Hypertext 2009 conference. The second contact trace is generated by a mobility model, more specifically the Random Waypoint model, as described in Chapter 2. The third contact trace is generated by a simple model where, at each step, a contact between two randomly chosen users is produced.

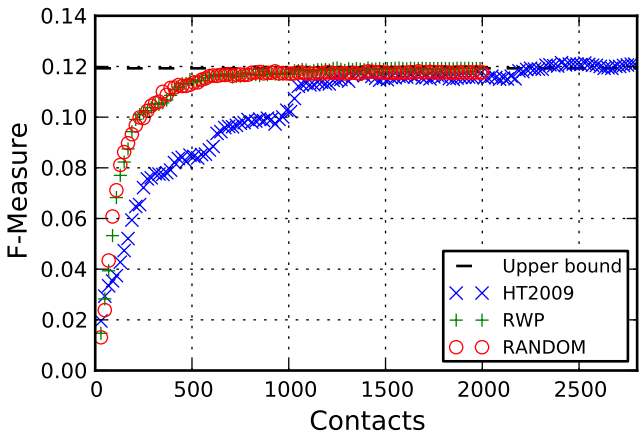


Figure 8.12: Evolution of F-Measure estimation in the Ad-Hoc scenario using the NEIGH approach. The approaches are compared with F-Measure estimation in the reference scenario for top 10 recommended items. The number of contacts is used as the observation clock. There are three different datasets used for this comparison: HT2009 is the experimental dataset collected in the Hypertext 2009 Conference; the other two datasets are synthetic datasets, based in the Random Waypoint model (RWP) and in a random contact sequence (RANDOM).

Figure 8.12 shows the difference in the recommendation performance when using the experimental dataset and when using synthetic datasets. The results show that synthetic datasets can produce unrealistic results when compared with experimental datasets, with the recommendation performance converging to the reference scenario.

8.5 Discussion

In the previous sections we proved that MobHinter converges to optimal recommendation predictions even if only random Ad-Hoc meetings are considered. MobHinter's field of applicability is very wide, and it includes movie selection, tourist attraction suggestion, personalized restaurant advices, and so on. Real world applications have pros and cons that go beyond simulative frameworks, and that cannot be realistically considered in a simplified model. In the following, we state some of the most relevant issues that may arise.

Synchronization: Even if we proved the practicality of our approach in a very extreme domain where nodes meet each other only using Ad-Hoc connections, we can realistically consider the existence of a remote service that can be accessed when Internet is available. During synchronization with such a service, a node can merge information collected during the ad-hoc phase with new information related to geographically remote users. When online, the device is able to refine the personalized predictions previously obtained, for example, accessing to content-based data repositories, downloading more specific information about items and users, recalculating affinities according to more accurate data.

Privacy and profile control: A MobHinter node has complete control over its user's profile and over recommendation calculations and predictions. In fact, every device collects all the information it needs for estimating ratings for unknown objects, and it has to exchange such information with other nodes. This greatly protects user's privacy, because even during a synchronization with a remote service, it has to pull all the data needed for calculation, by means of a (decentralized) directory system, e.g., for example, an user can query the system looking for potential neighbors that matches his preferences according a given similarity. This can be executed, for example, by flooding the searches, or using a Distributed Hash Table overlay network that allows structured and scalable Peer-to-Peer systems. During this process, the user can mask himself behind a presented identity with no relationships with the real entity. The identity is characterized in terms of a set of preferences, and she/he can decide which information to be publicly available.

Pseudonyms - many identities-one entity problem: The reader can question if a massive usage of pseudonyms can add noise to the recommendation system. Even if this problem must be carefully analyzed before estimating the real degree of its potential impact, we can adopt a straightforward solution for filtering out redundant identities. Because Equation 8.3 is not significantly affected by popularity factors, a device can simply discard neighbors and/or known hosts with identical ratings lists, assuming that this is an evidence of a replicated identity.

Chapter 9

Conclusions and Future Work

The widespread adoption of rich data driven services and powerful mobile devices create an unprecedented potential for innovative and popular mobile applications, through which users can communicate with each other by means of a broader and broader set of wireless network interfaces. Such devices provide new and promising means of collecting proximity data in order to understand the dynamics of human interactions. We started by studying synthetic and empirical user mobility datasets, widely used to model and simulate dynamics of information exchange in mobility networks. We also presented the *SocioPatterns platform*, an active RFID-based experimental framework used to collect empirical data of human proximity.

We presented some scientific tools used to model, analyze and characterize mobility data. We showed how human proximity data can be conveniently modeled as dynamical networks and how most of the concepts used in the analysis of static complex networks, which were later extended to the domain of dynamical networks, can be used to characterize this data. We also presented some tools focused on analysis and visualization, some of them developed in the scope of this work, and used as instruments to understand the dynamics behind such data.

We studied the data diffusion process in a real-world dynamic networks of human proximity and we analyzed the topological and temporal dynamics of the networks, focusing on the interactions between participants in large-scale social gatherings. We highlighted the temporal heterogeneity that arises from a number of social activities.

To investigate the general properties of information propagation, we focused on a simple flooding routing protocol that allows us to expose the interplay between network topology and the bursty nature of human activity. We showed that the distribution of message delivery times is strongly affected by the temporal heterogeneity of proximity events. We studied the effect that different definitions of "delivery time" have over the delivery time distribution. Strikingly, we found results that are universal across different

experiments, and are independent of the distribution of the contacts during time.

Moreover, we made a first step at comparing the measured sequences of proximity events with sequences generated by using commonly accepted models of human mobility, such as the Random Waypoint model and the Truncated Lévy Walk model, which are widely used in the domain of opportunistic and delay-tolerant networks, and we reported a strong difference between the propagation processes on model-based and real-world proximity networks. This points to the importance of taking into account realistic contact patterns for studying dynamical processes on dynamical proximity networks. In fact, the dynamics of information diffusion depends on non-trivial properties of contacts and inter-contact time intervals, at least as much as on the topological and temporal heterogeneity of human mobility.

It is important to note that most mobile phones are now equipped with geo-location features, which means that more and more applications and tools can use location based services to bring together location and people in interesting ways. In fact, more complex applications could take advantage of emergent community patterns, which could be inferred from the history of social interactions and places people visit. Such applications could enhance and personalize their user's geo-social experience by, for example, recommending newly identified items or places.

Based on such analysis and findings, we focused on collaborative filtering techniques in mobile networks where spontaneous similarity relations are derived and exploited in order to push personalized suggestions by way of direct meetings between users. We referred to a scenario where devices exchange information with users in the proximity without accessing to any remote online directory services. We proposed epidemic collaborative strategies to spread ratings in a selective way over self-organized communities of users. We simulated the ad-hoc environment and we found that the proposed approach converges to the prediction accuracy measured in a domain with complete knowledge with a sustainable overhead.

Our results on the general properties of information diffusion call for future work in the direction of defining fine observables that can capture the properties of the proximity networks that bear relevance to a variety of general processes occurring over them. Such observables could be used to compare the synthetic proximity networks generated by established models of human mobility with the proximity networks recorded in experimental settings. This will allow to expose the limits of current mobility models, and to devise more realistic modeling schemes.

We also point out that selective dissemination of information through communities with similar preferences using collaborative filtering approaches could be even more effective in densely populated urban areas and during peak usage hours, by using opportunistic networking. By using hybrid architectures based on a mixture of Hotzones with WiFi and mixed zones with mobile-to-mobile communication, costs for network operators can be reduced. Battery usage also can be reduced due to less range requirements.

The implementation of a prototype for the Android[1] platform in order to experiment the *MobHinter* framework in a real scenario is discussed in Appendix A.

[1] http://code.google.com/android/

Acknowledgments

Many people deserve my gratitude for their work and the support they granted me during my doctoral program.

First, I would like to thank my PhD advisor Prof. Giancarlo Ruffo for his wise guidance and his teachings. I will be always grateful to him for believing in me and for having encouraged me to pursue a career in research. Many thanks also to Prof. Francesco Bergadano who supported my work in the institutions of the Department of Computer Science in Torino.

Big thanks to my co-advisor Dr. Ciro Cattuto, who provided me with very valuable assistance. I would also like to thank all the ISI Foundation in Torino for the warm hospitality over all these years, specially to the SocioPatterns team Alan Barrat, Wouter Van den Broeck, Marco Quaggiotto, and Luca Cappa.

A particular thank and a fond farewell goes to my friends and colleagues from the ARC^2S group Luca Aiello, Rossano Schifanella, Marco Milanesio, Alessandro Basso, Martina Deplano, and Francesco Spoto.

Thanks to my coauthors Alain Barrat, Ciro Cattuto, Rossano Schifanella, Wouter Van den Broeck, Marco Milanesio, Alessandro Basso, and Cristina Gena for their crucial contribution to the papers we published during these three years. Among them, a special thanks goes to Alain and Ciro who taught me a lot and made me a better researcher.

I am grateful to the Gephi Team, specially to Mathieu Bastian and Sebastien Heymann for the opportunity to participate as a developer during the Google Summer of Code 2010 program, and all Gephi contributors for this great tool.

My sincere thanks are due to the official referees, Alessandro Flammini and Andrea Vitaletti, for their detailed review, constructive criticism and precious comments to improve the final version of this thesis.

Finally, I acknowledge the financial support from the projects "World Wide Style" (WWS), "Service á la carte" (SALC), and "Interactive Services for Mobiles"(IS4.MOBI), and thanks to Inrete S.r.l. for providing data collected through the Vcast service.

Bibliography

[1] Eytan Adar. GUESS: a language and interface for graph exploration. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 791–800, New York, NY, USA, 2006. ACM.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 835–844, New York, NY, USA, 2007. ACM.

[4] Michael Aizenman and David Barsky. Sharpness of the phase transition in percolation models. *Communications in Mathematical Physics*, 108:489–526, 1987. 10.1007/BF01212322.

[5] Harith Alani, Martin Szomsor, Ciro Cattuto, Wouter Van den Broeck, Gianluca Correndo, and Alain Barrat. Live Social Semantics. *Lecture Notes in Computer Science*, 5823:698–714, 2009.

[6] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[7] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, October 2000.

[8] M. Ángeles, Alessandro Flammini, and Filippo Menczer. Modeling Statistical Properties of Written Text. *PLoS ONE*, 4(4):e5372, 04 2009.

[9] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences of the United States of America*, 106(51):21544–21549, 2009.

[10] Elisa Baglioni, Luca Becchetti, Lorenzo Bergamini, Ugo Colesanti, Luca Filipponi, Andrea Vitaletti, and Giuseppe Persiano. A lightweight privacy preserving SMS-based recommendation system for mobile users. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 191–198, New York, NY, USA, 2010. ACM.

[11] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communication of the ACM*, 40(3):66–72, March 1997.

[12] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 299–331. Springer US, 2011.

[13] Albert-László Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[14] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[15] Alain Barrat, Marc Barthélemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, March 2004.

[16] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. Modeling the evolution of weighted networks. *Phys. Rev. E*, 70:066149, Dec 2004.

[17] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, New York, NY, USA, 2008.

[18] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '98, pages 76–84, New York, NY, USA, 1998. ACM.

[19] Alessandro Basso, Marco Milanesio, and Giancarlo Ruffo. Events discovery for personal video recorders. In *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 171–174, New York, NY, USA, 2009. ACM.

[20] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *International Conference on Weblogs and Social Media*, 2009.

[21] Vladimir Batagelj and Andrej Mrvar. Pajek - Analysis and Visualization of Large Networks. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 8–11. Springer Berlin Heidelberg, 2002.

[22] Luca Becchetti, Ugo Colesanti, Alberto Marchetti-Spaccamela, and Andrea Vitaletti. Recommending items in pervasive scenarios: models and experimental analysis. *Knowledge and Information Systems*, 28:555–578, 2011.

[23] Robert M. Bell and Yehuda Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *ICDM '07: Proceedings of the Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, October 2007. IEEE Computer Society.

[24] Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:55–66, July 2001.

[25] Christian Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '01, pages 19–27, New York, NY, USA, 2001. ACM.

[26] Christian Bettstetter, Hannes Hartenstein, and Xavier Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wirel. Netw.*, 10:555–567, September 2004.

[27] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python.* O'Reilly, 2009.

[28] Douglas M. Blough, Giovanni Resta, and Paolo Santi. A statistical analysis of the long-run node spatial distribution in mobile ad hoc networks. *Wirel. Netw.*, 10:543–554, September 2004.

[29] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[30] Chiara Boldrini, Marco Conti, and Andrea Passarella. Modelling data dissemination in opportunistic networks. In *CHANTS '08: Proceedings of the third ACM workshop on Challenged networks*, pages 89–96, New York, NY, USA, 2008. ACM.

[31] Béla Bollobás. *Modern graph theory.* Springer-Verlag, New York, 1998.

[32] Béla Bollobás. *Random graphs.* Cambridge studies in advanced mathematics. Cambridge University Press, 2001.

[33] Katy Börner, S Sanyal, and A Vespignani. Network science. *Annual Review of Information Science and Technology*, 41(1):537–607, 2008.

[34] J. Bouttier, P. Di Francesco, and E. Guitter. Geodesic distance in planar graphs. *Nuclear Physics B*, 663(3):535 – 567, 2003.

[35] Simon R Broadbent and John M Hammersley. Percolation processes. I. Crystals and mazes. *Proc Cambridge Philos Soc*, 53:629–641, 1957.

[36] R.A. Brualdi and H.J. Ryser. *Combinatorial matrix theory*. Encyclopedia of mathematics and its applications. Cambridge University Press, 1991.

[37] B. Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. Technical Report RR-4589, INRIA, October 2002.

[38] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, February 2009.

[39] Han Cai and Do Young Eun. Crossing over the bounded domain: from exponential to power-law inter-meeting time in MANET. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 159–170, New York, NY, USA, 2007. ACM.

[40] Han Cai and Do Young Eun. Toward stochastic anatomy of inter-meeting time distribution under general mobility models. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 273–282, New York, NY, USA, 2008. ACM.

[41] Guido Caldarelli. *Scale-Free Networks: Complex Webs in Nature and Technology (Oxford Finance)*. Oxford University Press, USA, May 2007.

[42] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[43] Kathleen M. Carley, Jana Diesner, Jeffrey Reminga, and Maksim Tsvetovat. Toward an interoperable dynamic network analysis toolkit. *Decis. Support Syst.*, 43:1324–1347, August 2007.

[44] Francesca Carmagnola, Andrea Loffredo, and Giorgio Bernardi. VCast on Facebook: Bridging Social and Similarity Networks. In *HT '09: Proceedings of the Twentieth ACM Conference on Hypertext and Hypermedia*, New York, NY, USA, July 2009. ACM.

[45] Iacopo Carreras, Francesco De Pellegrini, Daniele Miorandi, David Tacconi, and Imrich Chlamtac. Why neighbourhood matters: interests-driven opportunistic data diffusion schemes. In *CHANTS '08: Proceedings of the third ACM workshop on Challenged networks*, pages 81–88, New York, NY, USA, 2008. ACM.

[46] Ciro Cattuto, Alain Barrat, Andrea Baldassarri, Gregory Schehr, and Vittorio Loreto. Collective dynamics of social annotation. *Proceedings of the National Academy of Sciences*, 106(26):10511–10515, 2009.

[47] Ciro Cattuto, Wouter Van den Broeck, Alain Barrat, Vittoria Colizza, Jean-François Pinton, and Alessandro Vespignani. Dynamics of Person-to-Person Interactions from Distributed RFID Sensor Networks. *PLoS ONE*, 5(7):e11596, 07 2010.

[48] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, april 2006.

[49] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606–620, june 2007.

[50] Augustin Chaintreau, Abderrahmen Mtibaa, Laurent Massoulie, and Christophe Diot. The diameter of opportunistic mobile networks. In *Proceedings of the 2007 ACM CoNEXT conference*, CoNEXT '07, pages 12:1–12:12, New York, NY, USA, 2007. ACM.

[51] Gary Chartrand and Linda M. Lesniak. *Graphs and digraphs (2. ed.)*. Wadsworth & Brooks / Cole mathematics series. Wadsworth, 1986.

[52] Nicholas A. Christakis and James H. Fowler. The Spread of Obesity in a Large Social Network over 32 Years. *New England Journal of Medicine*, 357(4):370–379, 2007.

[53] Nicholas A. Christakis and James H. Fowler. The Collective Dynamics of Smoking in a Large Social Network. *New England Journal of Medicine*, 358(21):2249–2258, 2008.

[54] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, California, 1999. ACM.

[55] Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences of the United States of America*, 103(7):2015–2020, 2006.

[56] Vittoria Colizza, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Characterization and modeling of protein–protein interaction networks. *Physica A: Statistical Mechanics and its Applications*, 352(1):1–27, 2005.

[57] Corinna Cortes, Daryl Pregibon, and Chris Volinsky. Communities of interest. In *Advances in Intelligent Data Analysis*, volume 2189 of *Lecture Notes in Computer Science*, pages 105–114. Springer Berlin / Heidelberg, 2001.

[58] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[59] Paolo Cremonesi and Roberto Turrin. Analysis of cold-start recommendations in IPTV systems. In *RecSys '09: Proc. of the 3rd ACM conf. on Recommender systems*, pages 233–236, New York, NY, USA, 2009. ACM.

[60] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, pages 32–40, New York, NY, USA, 2007. ACM.

[61] Jörn Davidsen, Holger Ebel, and Stefan Bornholdt. Emergence of a Small World from Local Interactions: Modeling Acquaintance Networks. *Phys. Rev. Lett.*, 88:128701, Mar 2002.

[62] Alexandre de Spindler, Moira C. Norrie, and Michael Grossniklaus. Collaborative Filtering Based on Opportunistic Information Sharing in Mobile Ad-Hoc Networks. In *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 408–416. Springer, 2007.

[63] D. DeRoure, W. Hall, S. Reich, G. Hill, A. Pikrakis, and M. Stairmand. MEMOIR − an open framework for enhanced navigation of distributed information. *Inf. Process. Manage.*, 37(1), 2001.

[64] Mukund Deshpande and George Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[65] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *EPNACS: Emergent Properties in Natural and Artificial Complex Systems*, 2007.

[66] Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[67] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Phys. Rev. E*, 66:035103, Sep 2002.

[68] J. P. Eckmann, E. Moses, and D. Sergi. Entropy of dialogues creates coherent structures in e-mail traffic. *Proc Natl Acad Sci U S A*, 101(40):14333–14337, October 2004.

[69] Albert Einstein. On the movement of small particles suspended in stationary liquids required by the molecular-kinetic theory of heat. *Annalen der Physik*, 17(8):549–560, 1905.

[70] Paul Erdös and Alfred Rényi. On Random Graphs. *Publ. Math.*, 6:290–297, 1959.

[71] Paul Erdös and Alfred Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5:17–61, 1960.

[72] Vijay Erramilli, Augustin Chaintreau, Mark Crovella, and Christophe Diot. Diversity of forwarding paths in pocket switched networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 161–174, New York, NY, USA, 2007. ACM.

[73] Afonso Ferreira. Building a reference combinatorial model for manets. *IEEE Network*, 18(5):24–29, 2004.

[74] Klaus Finkenzeller. *RFID Handbook*. John Wiley and Sons, 2003.

[75] Santo Fortunato, Alessandro Flammini, and Filippo Menczer. Scale-Free Network Growth by Ranking. *Phys. Rev. Lett.*, 96:218701, May 2006.

[76] L.C. Freeman. *The development of social network analysis: a study in the sociology of science*. Empirical Press, 2004.

[77] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, March 1977.

[78] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[79] J.J. Garcia-Luna-Aceves and E.L. Madruga. A multicast routing protocol for adhoc networks. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 784–792 vol.2, mar 1999.

[80] M. Gardner. *Martin Gardner's Sixth book of mathematical diversions from Scientific American*. University of Chicago Press, 1971.

[81] Hugh Glaser, Ian Millard, and Afraz Jaffri. RKBExplorer.com:A Knowledge Driven Infrastructure for Linked Data Providers. In *European Semantic Web Conference*, volume 5021/2, pages 797–801. Springer, June 2008.

[82] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

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

[84] Sandra González-Bailón, Javier Borge-Holthoefer, Alejandro Rivero, and Yamir Moreno. The dynamics of protest recruitment through an online network. *Nature Scientific Reports*, 1, 2011.

[85] P. Grobstein. From Complexity to Emergence and Beyond: Towards Empirical Non-foundationalism as a Guide for Inquiry. *Soundings*, 90(1/2):301–323, 2007.

[86] Robin Groenevelt, Philippe Nain, and Ger Koole. The message delay in mobile ad hoc networks. *Perform. Eval.*, 62(1-4):210–228, 2005.

[87] M. Grossglauser and D.N.C. Tse. Mobility increases the capacity of ad hoc wireless networks. *Networking, IEEE/ACM Transactions on*, 10(4):477–486, aug 2002.

[88] Dirk Helbing. Traffic and related self-driven many-particle systems. *Rev. Mod. Phys.*, 73:1067–1141, Dec 2001.

[89] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proc. of the 22nd annual intern. ACM SIGIR conf. on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

[90] Petter Holme. Network reachability of real-world contact sequences. *Physical Review E*, 71(4):046119+, April 2005.

[91] Petter Holme and M. E. J. Newman. Nonequilibrium phase transition in the coevolution of networks and opinions. *Phys. Rev. E*, 74:056108, Nov 2006.

[92] Petter Holme and Jari Saramäki. Temporal networks. *CoRR*, abs/1108.1780, 2011.

[93] Seongik Hong, Injong Rhee, Seong Joon Kim, Kyunghan Lee, and Song Chong. Routing Performance Analysis of Human-Driven Delay Tolerant Networks using the Truncated Levy Walk Model. In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE International Workshop on Mobility Models for Networking Research*, pages 25–32, New York, NY, USA, 2008. ACM.

[94] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '99, pages 53–60, New York, NY, USA, 1999. ACM.

[95] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM '08: Proc. of the 2008 Eighth IEEE Intl. Conf. on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[96] Yifan F. Hu. Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10:37–71, 2005.

[97] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and human mobility in conference environments. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, New York, NY, USA, 2005. ACM.

[98] Pan Hui and Jon Crowcroft. Human mobility models and opportunistic communications system design. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1872):2005–2016, June 2008.

[99] Pan Hui, Jon Crowcroft, and Eiko Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 241–250, New York, NY, USA, 2008. ACM.

[100] Jens Illig, Andreas Hotho, Robert Jäschke, and Gerd Stumme. A comparison of content-based tag recommendations in folksonomy systems. In Karl Erich Wolff, Dmitry E. Palchunov, Nikolay G. Zagoruiko, and Urs Andelfinger, editors, *Knowledge Processing and Data Analysis*, volume 6581 of *Lecture Notes in Computer Science*, pages 136–149, Berlin/Heidelberg, 2011. Springer.

[101] José Luis Iribarren and Esteban Moro. Impact of Human Activity Patterns on the Dynamics of Information Diffusion. *Phys. Rev. Lett.*, 103:038702, Jul 2009.

[102] Lorenzo Isella, Mariateresa Romano, Alain Barrat, Ciro Cattuto, Vittoria Colizza, Wouter Van den Broeck, Francesco Gesualdo, Elisabetta Pandolfi, Lucilla Ravà, Caterina Rizzo, and Alberto Eugenio Tozzi. Close encounters in a pediatric ward: Measuring face-to-face proximity and mixing patterns with wearable sensors. *PLoS ONE*, 6(2):e17144, 02 2011.

[103] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? Analysis of face-to-face behavioral networks. *J. Theor. Biol.*, 271:166 – 180, 2011.

[104] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[105] Joël, Hens Niel, Jit Mark, Beutels Philippe, Auranen Kari, Mikolajczyk Rafael, Massari Marco, Salmaso Stefania, Tomba Gianpaolo Scalia, Wallinga Jacco, Heijne Janneke, Sadkowska-Todys Malgorzata, Rosinska Magdalena, and Edmunds W. John Mossong. Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. *PLoS Med*, 5(3):e74, 03 2008.

[106] Neil F. Johnson, Chen Xu, Zhenyuan Zhao, Nicolas Ducheneaut, Nicholas Yee, George Tita, and Pak Ming Hui. Human group formation in online guilds and offline gangs driven by a common team dynamic. *Phys. Rev. E*, 79:066117, Jun 2009.

[107] Thomas Karagiannis, Jean-Yves Le Boudec, and Milan Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 183–194, New York, NY, USA, 2007. ACM.

[108] Jouni Karvo and Jörg Ott. Time scales and delay-tolerant routing protocols. In *CHANTS '08: Proceedings of the third ACM workshop on Challenged networks*, pages 33–40, New York, NY, USA, 2008. ACM.

[109] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[110] W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society of London. Series A*, 115(772):700–721, 1927.

[111] Minkyong Kim, David Kotz, and Songkuk Kim. Extracting a mobility model from real user traces. In *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1–13, Washington, DC, USA, April 2006. IEEE Computer Society.

[112] Joseph Klafter, Michael F. Shlesinger, and Gert Zumofen. Beyond Brownian Motion. *Physics Today*, 49(2):33+, 1996.

[113] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. ACM Press New York, NY, USA, 1993.

[114] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[115] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proc. of the 15th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[116] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, August 2009.

[117] T. Kos, M. Grgic, and G. Sisul. Mobile User Positioning in GSM/UMTS Cellular Networks. In *Multimedia Signal Processing and Communications, 48th International Symposium ELMAR-2006 focused on*, pages 185–188, june 2006.

[118] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in a social communication network. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 435–443, New York, NY, USA, 2008. ACM.

[119] Vassilis Kostakos, Eamonn O'Neill, and Alan Penn. Brief encounter networks. *CoRR*, abs/0709.0223, 2007. informal publication.

[120] D. Kotz and T. Henderson. CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth. *Pervasive Computing, IEEE*, 4(4):12–14, oct.-dec. 2005.

[121] Jussi M. Kumpula, Jukka-Pekka Onnela, Jari Saramäki, Kimmo Kaski, and János Kertész. Emergence of Communities in Weighted Networks. *Phys. Rev. Lett.*, 99:228701, Nov 2007.

[122] Derek Lam, Donald C. Cox, and Jennifer Widom. Teletraffic Modeling for Personal Communications Services. *IEEE COMMUNICATIONS MAGAZINE*, 35:79–87, 1997.

[123] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[124] Jason Lebrun and Chen-Nee Chuah. Bluetooth Content Distribution Stations on Public Transit. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 63–65, New York, NY, USA, 2006. ACM Press.

[125] Chul-Ho Lee and Do Young Eunt. Heterogeneity in contact dynamics: helpful or harmful to forwarding algorithms in DTNs? In *WiOPT'09: Proceedings of the 7th international conference on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 72–81, Piscataway, NJ, USA, 2009. IEEE Press.

[126] V. Lenders, G. Karlsson, and M. May. Wireless Ad Hoc Podcasting. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 273–283, june 2007.

[127] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 462–470, New York, NY, USA, 2008. ACM.

[128] Michael Ley. The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. In *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings*, volume 2476 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2002.

[129] Yunfeng Lin, Baochun Li, and Ben Liang. Stochastic analysis of network coding in epidemic routing. *IEEE Journal on Selected Areas in Communications*, 26(5):794–808, June 2008.

[130] Vittorio Loreto. Tagora - Emergent Semantics in Social Online. *KI*, 22(1):63–64, 2008.

[131] Hans Peter Luhn. *Selective dissemination of new scientific information with the aid of electronic processing equipment.* International Business Machines Corporation. Advanced Systems Development Division, 1959.

[132] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 641–650, New York, NY, USA, 2009. ACM.

[133] J. G. Markoulidakis, G. L. Lyberopoulos, D. F. Tsirkas, and E. D. Sykas. Mobility modeling in third-generation mobile telecommunications systems. *IEEE Personal Communications*, 4:41–56, 1997.

[134] M. Marsili, F. Vega-Redondo, and F. Slanina. The Rise and Fall of a Networked Society: A Formal Model. *Proceedings of the National Academy of Sciences of the United States of America*, 101(6):1439, 2004.

[135] Michele Mastrogiovanni, Chiara Petrioli, Michele Rossi, Andrea Vitaletti, and Michele Zorzi. Integrated data delivery and interest dissemination techniques for wireless sensor networks. In *Proceedings of the Global Telecommunications Conference, 2006. GLOBECOM '06, San Francisco, CA, USA, 27 Nov - 1 Dec 2006*. IEEE, 2006.

[136] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[137] Alessandro Mei, Giacomo Morabito, Paolo Santi, and Julinda Stefa. Social-Aware Stateless Forwarding in Pocket Switched Networks. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 251–255. IEEE, 2011.

[138] Andrew G. Miklas, Kiran K. Gollu, Kelvin K. W. Chan, Stefan Saroiu, Krishna P. Gummadi, and Eyal De Lara. Exploiting Social Interactions in Mobile Systems. In *UbiComp'07: Proc. of the 9th Int. Conf. on Ubiquitous Computing*, volume 4717/2007 of *Lecture Notes in Computer Science*, pages 409–428. Springer-Verlag, Berlin, Heidelberg, 2007.

[139] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244, December 1990.

[140] Knud Möller, Tom Heath, Siegfried Handschuh, and John Domingue. Recipes for Semantic Web dog food - The ESWC and ISWC metadata projects. In *Proceedings of the 6th International Semantic Web Conference*, ISWC'07, pages 802–815, Berlin, Heidelberg, 2007. Springer-Verlag.

[141] James Moody, Daniel McFarland, and Skye Bender-DeMoll. Dynamic Network Visualization. *American Journal of Sociology*, 110(4):1206–1241, 2005.

[142] Tamara Macushla Munzner. *Interactive visualization of large graphs and networks*. PhD thesis, Stanford University, Stanford, CA, USA, 2000.

[143] P. Nain, D. Towsley, Benyuan Liu, and Zhen Liu. Properties of random direction models. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1897–1907 vol. 3, march 2005.

[144] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*, 98(2):404–409, January 2001.

[145] M. E. J. Newman. Assortative mixing in networks. *Physical Review Letter*, 89:208701, 2002.

[146] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67:026126, 2003.

[147] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68(3):036122, Sep 2003.

[148] Mark Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, Princeton, NJ, USA, 2006.

[149] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and Albert-László Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007.

[150] Christopher R. Palmer, Phillip B. Gibbons, and Christos Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90, New York, NY, USA, 2002. ACM.

[151] R. Pastor-Satorras, A. Vázquez, and A. Vespignani. Dynamical and correlation properties of the Internet. *Physical Review Letters*, 87:258701+, 2001.

[152] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic Spreading in Scale-Free Networks. *Physical Review Letters*, 86:3200–3203, Apr 2001.

[153] Romualdo Pastor-Satorras and Alessandro Vespignani. *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, New York, NY, USA, 2004.

[154] Eric Paulos and Elizabeth Goodman. The familiar stranger: anxiety, comfort, and play in public places. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 223–230, New York, NY, USA, 2004. ACM.

[155] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[156] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill Webert: Identifying Interesting Web Sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

[157] Karl Pearson. The Problem of the Random Walk. *Nature*, 72:294–294, 1905.

[158] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Wireless Communications and Networking Conference, 1999. WCNC.*, volume 3, pages 1538–1542. IEEE, 1999.

[159] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11):134–141, November 2006.

[160] A. Perer and B. Shneiderman. Balancing Systematic and Flexible Exploration of Social Networks. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):693–700, sept.-oct. 2006.

[161] D.D. Perkins, H.D. Hughes, and C.B. Owen. Factors affecting the performance of ad hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 4, pages 2048–2052 vol.4, 2002.

[162] Saverio Perugini, Marcos André Gonçalves, and Edward A. Fox. Recommender Systems Research: A Connection-Centric Survey. *J. Intell. Inf. Syst.*, 23(2):107–143, 2004.

[163] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. MobiClique: Middleware for Mobile Social Networking. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM.

[164] Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, and Steve Lawrence. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 437–444, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[165] Daniele Quercia, Giusy Di Lorenzo, Francesco Calabrese, and Carlo Ratti. Mobile Phones and Outdoor Advertising: Measurable Advertising. *IEEE Pervasive Computing*, 10:28–36, 2011.

[166] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. Truthy : Mapping the Spread of Astroturf in Microblog Streams. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011 (Companion Volume)*, pages 249–252. ACM, 2011.

[167] Lord Rayleigh. The Problem of the Random Walk. *Nature*, 72:318–318, 1905.

[168] Paul Resnick and Hal R. Varian. Recommender Systems - Introduction to the Special Section. *Communication ACM*, 40(3):56–58, March 1997.

[169] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, and Song Chong. On the Levy-Walk Nature of Human Mobility. In *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pages 924–932. IEEE, April 2008.

[170] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[171] E.M. Royer, P.M. Melliar-Smith, and L.E. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 3, pages 857–861 vol.3, 2001.

[172] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[173] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[174] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of Dimensionality Reduction in Recommender System - A Case Study. In *ACM WebKDD Workshop*, 2000.

[175] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354, New York, NY, USA, 1998. ACM Press.

[176] Salvatore Scellato. Beyond the social web: the geo-social revolution. *SIGWEB Newsletter*, pages 5:1–5:5, September 2011.

[177] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative Filtering Recommender Systems. *International Journal of Electronic Business*, 2(1):291–324, 2007.

[178] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[179] Antoine Scherrer, Pierre Borgnat, Eric Fleury, Jean-Loup Guillaume, and Céline Robardet. Description and simulation of dynamic mobility networks. *Computer Networks*, 52(15):2842–2858, 2008.

[180] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34, New York, NY, USA, 2008. ACM.

[181] V. Seshadri, G.V. Zaruba, and M. Huber. A bayesian sampling approach to indoor localization of wireless devices using received signal strength indication. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 75–84, march 2005.

[182] Cosma Rohilla Shalizi and Andrew C Thomas. Homophily and Contagion Are Generically Confounded in Observational Social Network Studies. *Sociological Methods Research*, 40(2):27, 2010.

[183] Shannon, A Markiel, O Ozier, N S Baliga, J T Wang, D Ramage, N Amin, B Schwikowski, and T Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, November 2003.

[184] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, New York, 1995. ACM.

[185] Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE Comput. Soc. Press, sep 1996.

[186] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Efficient routing in intermittently connected mobile networks: the multiple-copy case. *IEEE/ACM Trans. Netw.*, 16(1):77–90, 2008.

[187] Juliette Stehlé, Alain Barrat, and Ginestra Bianconi. Dynamical and bursty interactions in social networks. *Phys. Rev. E*, 81:035101, Mar 2010.

[188] Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, and Vito D.P. Servedio. Folksonomies, the Semantic Web, and Movie Recommendation . In *4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0*, 2007.

[189] Amund Tveit. Peer-to-peer based recommendations for mobile commerce. In *WMC '01: Proceedings of the 1st international workshop on Mobile commerce*, pages 26–29, New York, NY, USA, 2001. ACM Press.

[190] Amin Vahdat and David Becker. Epidemic Routing for Partially-Connected Ad Hoc Networks. Technical report, Duke University, April 2000.

[191] Wouter Van den Broeck, Ciro Cattuto, Alain Barrat, Martin Szomsor, Gianluca Correndo, and Harith Alani. The Live Social Semantics application: a platform for integrating face-to-face presence with on-line social networking. In *Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCol 2010)*, pages 226–231, Mannheim, Germany, April 2010.

[192] A. Vázquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the Internet. *Physical Review E*, 65:066130+, 2002.

[193] Alexei Vázquez, João Gama Oliveira, Zoltán Dezsö, K-I. Goh, Imre Kondor, and Albert-László Barabási. Modeling bursts and heavy tails in human dynamics. *Phys Rev E*, 73(036127), 2006.

[194] Alexei Vazquez, Balázs Rácz, András Lukács, and Albert-László Barabási. Impact of Non-Poissonian Activity Patterns on Spreading Processes. *Phys. Rev. Lett.*, 98:158702, Apr 2007.

[195] Federico Vazquez, Víctor M. Eguíluz, and Maxi San Miguel. Generic Absorbing Transition in Coevolution Dynamics. *Phys. Rev. Lett.*, 100:108702, Mar 2008.

[196] Fernanda B. Viégas, Martin Wattenberg, and Jonathan Feinberg. Participatory Visualization with Wordle. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1137–1144, 2009.

[197] K.H. Wang and Baochun Li. Group mobility and partition prediction in wireless ad-hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pages 1017–1021 vol.2, 2002.

[198] Yu Wang and Hongyi Wu. Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN): A New Paradigm for Pervasive Information Gathering. *IEEE Transactions on Mobile Computing*, 6:1021–1034, 2007.

[199] Stanley Wasserman and Katerine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[200] Duncan J Watts. The "New" Science of Networks. *Annual Review of Sociology*, 30(1):243–270, 2004.

[201] Duncan J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[202] H.C. White. *Identity and control: a structural theory of social action*. Princeton paperbacks. Princeton University Press, 1992.

[203] Jenna Wortham. Customers Angered as iPhones Overload AT&T. *The New York Times*, page B1, 09 2009. `http://www.nytimes.com/2009/09/03/technology/companies/03att.html` [Online; accessed 10-Feb-2012].

[204] Eiko Yoneki. The Importance of Data Collection for Modelling Contact Networks. In *CSE '09: Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 940–943, Washington, DC, USA, 2009. IEEE Computer Society.

[205] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312–1321 vol.2, march-3 april 2003.

[206] Sheng Zhang, Weihong Wang, James Ford, Fillia Makedon, and Justin Pearlman. Using Singular Value Decomposition Approximation for Collaborative Filtering. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pages 257–264, Washington, DC, USA, 2005. IEEE Computer Society.

[207] Xiaolan Zhang, Giovanni Neglia, Jim Kurose, and Don Towsley. Performance modeling of epidemic routing. *Comput. Netw.*, 51(10):2867–2891, 2007.

[208] M. M. Zonoozi and P. Dassanayake. User mobility modeling and characterization of mobility patterns. *Selected Areas in Communications, IEEE Journal on*, 15(7):1239–1252, September 1997.

[209] G. Zyba, G.M. Voelker, S. Ioannidis, and C. Diot. Dissemination in opportunistic mobile ad-hoc networks: The power of the crowd. In *INFOCOM, 2011 Proceedings IEEE*, pages 1179 –1187. IEEE, april 2011.

Appendices

Appendix A

Beyond Simulation: the MobHinter Implementation

The MobHinter algorithm, described in Chapter 8, has been implemented as a prototype for mobile devices (Android phones in particular). To minimize the efforts of portability across different platforms, we chose to use the library *PhoneGap*. Special plug-ins that implement the functionality required by the prototype have been developed for this library.

The prototype can also operate by connecting to a server and requesting the recommendations for a specific user. The server can in turn send the user rates. The recommendations in both modes of operation are generated based on the user profile.

Using the software, the user can search any item on Youtube.com, Flickr.com and Lastfm.com, assigning rates to the search results. These rates are sent to the server, that is used by the epidemic recommender system. The recommendations provided by the software are generated by both the MobHinter system and from the central server.

A.1 Technologies used in the prototype development

- *PhoneGap* http://www.phonegap.com/
- *Eclipse* http://www.eclipse.org/
- *jQuery* http://jquery.com/
- *jQueryMobile* http://jquerymobile.com/

A.2 Software Architecture

The software is designed to be multi-platform. For this purpose the PhoneGap libraries were used to minimize the efforts of portability. By using PhoneGap it has been possible to develop the interface in HTML5 and CSS3.

The features are implemented in JavaScript for the most part, except specifications that are not present in PhoneGap which required the development of specific plug-ins. The only parts platform-dependent are in fact part of these plug-in that interfaces the hardware.

The software architecture is generally illustrated in Figure A.1. The software has been developed for the Android platform. A version for iOS is available and is identical, except for the MobHinter epidemic recommendation system.

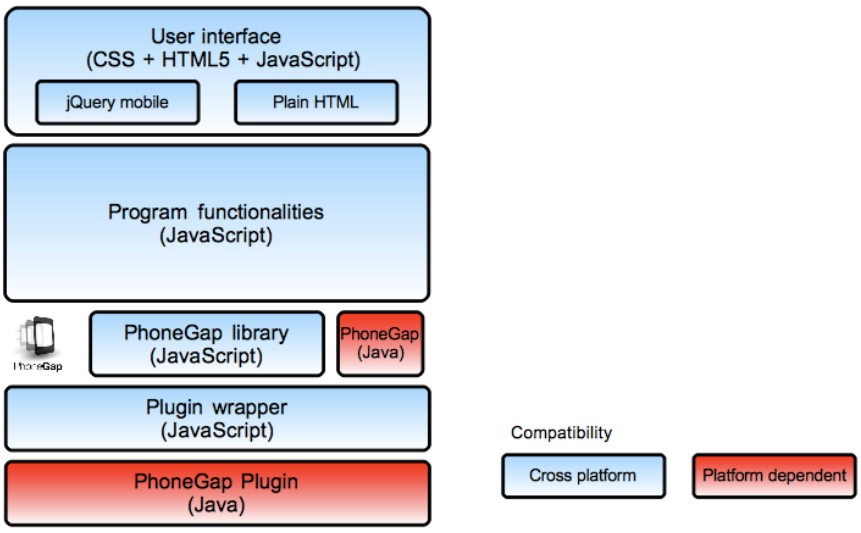


Figure A.1: General software architecture

Source code Organization

Figure A.2 shows the source code organization. All software files are under `assets/www`, as shown in the figure. The software files are in this folder, written in HTML5, CSS3 and JavaScript. This is the platform independent part. All source files that are platform dependent are found in the folder `src`. In the following, there is a description of some of the main source files.

- *css* contains the css files for the HTML interface.

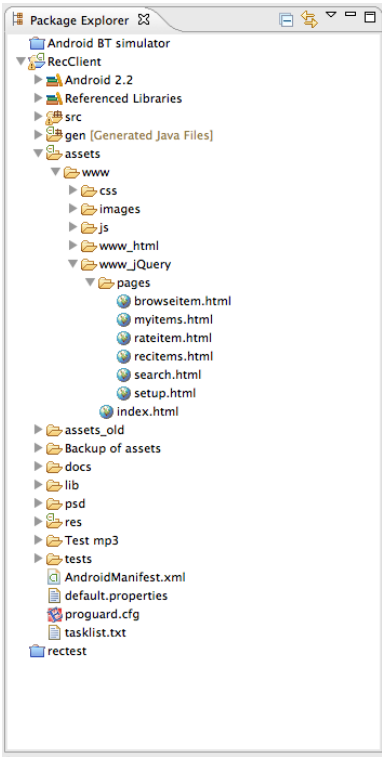


Figure A.2: Structure of the source code organization

- *images* contains the images used in the software.
- *js* contains the JavaScript code.
- *js/reclient.js* the implementation of the client functionalities.
- *js/mobhinter.js* the MobHinter implementation.
- *js/reclient-nojq.js* some of the functionalities in the HTML interface.
- *js/sessvar.js* save the data between different HTML pages.
- *js/jquery-cookie.js* is used to manage cookies.
- *js/json.js* is used to manage json.
- *js/phonega.0.9.5.js* is the PhoneGap library.
- *js/plugin/* contains the front-end for the PhoneGap plug-in.
- *www_html* is the HTML interface.
- *www_jQuery* is the jQueryMobile interface.

- *pages* contains the HTML pages for the different views.

A.3 Software Interface

There are two versions of the interface: the first is based on jQueryMobile and is recommended for more powerful devices (Android > 2.1), while the second is designed for older devices and uses simple HTML.

The functionality of the software is organized in four views: Search, Recommended Items, My Items and Setup.

1. *Login* Figure A.3. The user can log into the application.

2. *Search* Figure A.4. Implements the item search by keyword.

3. *Reccomended Items* Figure A.5. Presents to the user the list of recommended items, both from the server and from the epidemic recommendation system.

4. *MyItems* Figure A.5. Allows to the user to visualize the items available in the device.

5. *Setup* Allows to the user to visualize and change the software parameters.

Figure A.6 shows the same views using the HTML interface (Android < 2.0).

Figure A.3: Login view

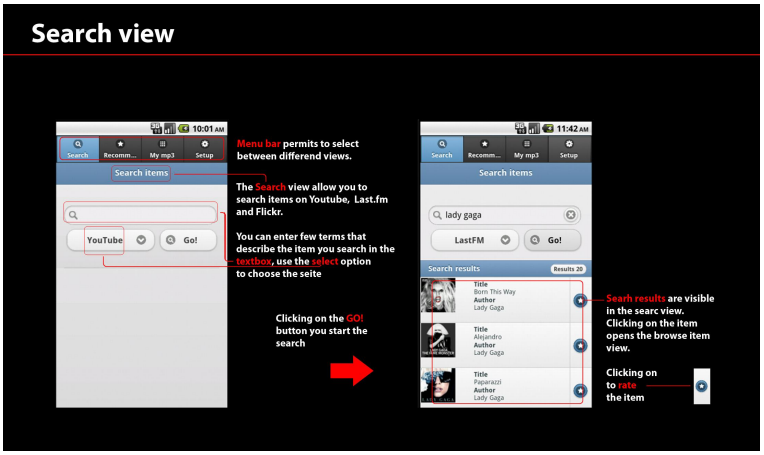


Figure A.4: Search view

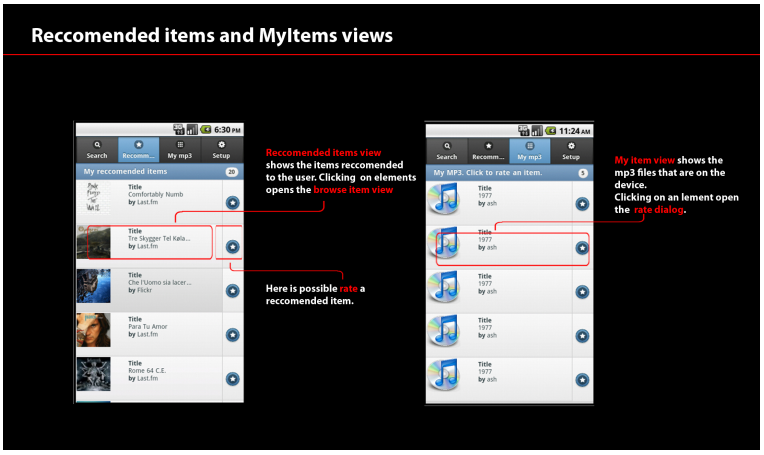


Figure A.5: Reccomended Items e MyItems views

A.4 Phonegap Plug-ins

Three plug-ins have been used in the development of the prototype. Two of them were developed specifically for the prototype, while the third was modified in order to suit your needs.

List MP3 plug-in

This plug-in is used to find and list the MP3 files that are present on the device. This plug-in has been written from scratch.

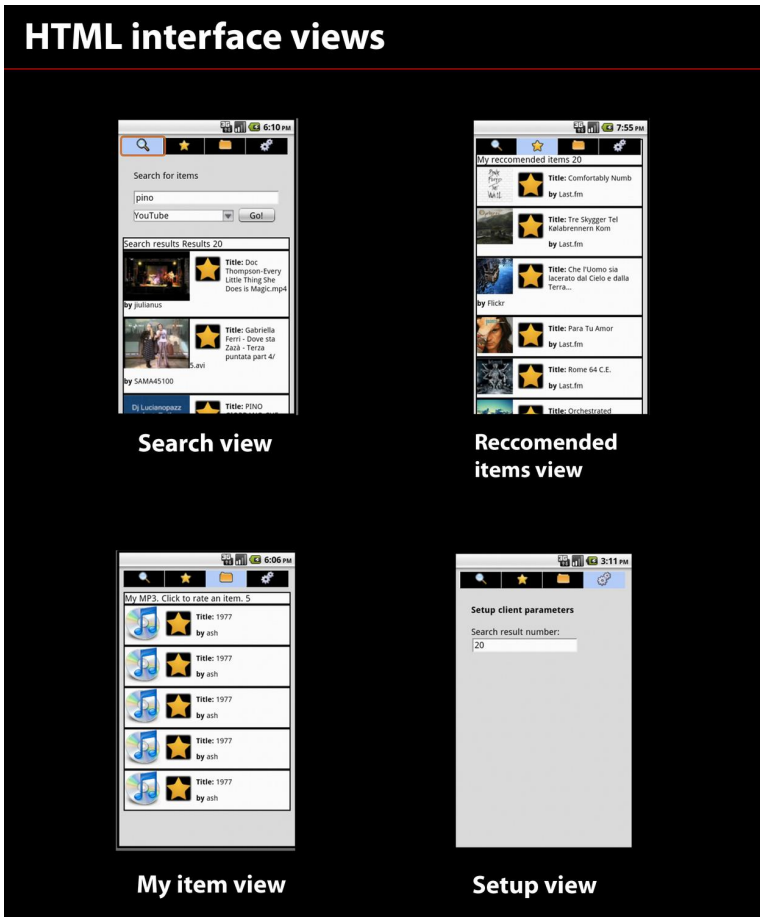


Figure A.6: HTML views

Facebook plug-in

The Facebook plug-in allows user authentication on the Facebook social network. This plug-in has been modified to be included in the prototype.

BlueTooth plug-in

The Bluetooth plug-in allows communication between different devices through the Bluetooth protocol.

Below are some notes for this plug-in that is one of the most critical parts of the software, also due to some bugs in certain versions of the Android system:

1. The Bluetooth protocol requires authorization from both devices in order to exchange data (pairing). This can be avoided for Android versions above 2.3.3 (Gin-

gerbread) and, through the Java reflection, on many Android devices with versions later than 2.0.

2. A device should become detectable to others. In some versions of Android, a bug prevents a device to become detectable for periods longer than 300 seconds.

3. The constant search for other devices can cause excessive consumption on the device battery.

A.5 iOS Version

The IOS version is identical to the Android prototype, exception for the MP3 and Bluetooth plugins that have not been implemented on this system.

Appendix B

Mobility Model Implementations

B.1 Random Walk Mobility Model

```
import numpy as np
from numpy.random import rand

# define a Uniform Distribution
U = lambda MIN, MAX, NR_SAMPLES: rand(NR_SAMPLES) * (MAX - MIN) + MIN

NR_NODES = 100
MAX_X, MAX_Y = 100, 100
NR_STEPS = 1000000
distance = 1.0

for i in xrange(NR_STEPS):
        theta = U(0, 2*np.pi, NR_NODES)
        x = x + distance * np.cos(theta)
        y = y + distance * np.sin(theta)

        # node bounces on the margins
        b = np.where(x<0)[0]
        x[b] = - x[b]
        b = np.where(x>MAX_X)[0]
        x[b] = 2*MAX_X - x[b]
        b = np.where(y<0)[0]
        y[b] = - y[b]
        b = np.where(y>MAX_Y)[0]
        y[b] = 2*MAX_Y - y[b]
```

Listing B.1.1: Python implementation of the the Random Walk model using the Numpy package.

B.2 Random Waypoint Mobility Model

```
import numpy as np
from numpy.random import rand

# define a Uniform Distribution
U = lambda MIN, MAX, NR_SAMPLES: rand(NR_SAMPLES) * (MAX - MIN) + MIN

NR_NODES = 100
MAX_X, MAX_Y = 100, 100
MIN_V, MAX_V = 1., 10.
MAX_WT = 10.
NR_STEPS = 1000000
x = U(0, MAX_X, NR_NODES)
y = U(0, MAX_Y, NR_NODES)
x_waypoint = U(0, MAX_X, NR_NODES)
y_waypoint = U(0, MAX_Y, NR_NODES)
wait_time = np.zeros(NR_NODES)
velocity = U(MIN_V, MAX_V, NR_NODES)

for i in xrange(NR_STEPS):
        # update node position
        theta = np.arctan2(y_waypoint - y, x_waypoint - x)
        x = x + velocity * np.cos(theta)
        y = y + velocity * np.sin(theta)
        # calculate distance to waypoint
        d = np.sqrt(np.square(y_waypoint-y) + np.square(x_waypoint-x))
        # update info for arrived nodes
        arrived = np.where(d<=velocity)[0]
        velocity[arrived] = 0.
        wait_time[arrived] = U(0, MAX_WT, len(arrived))
        # update info for paused nodes
        paused = np.where(velocity==0.)[0]
        wait_time[paused] -= 1.
        # update info for moving nodes
        moving = np.where(np.logical_and(velocity==0., wait_time<0.))[0]
        x_waypoint[moving] = U(0, MAX_X, len(moving))
        y_waypoint[moving] = U(0, MAX_Y, len(moving))
        velocity[moving] = U(MIN_V, MAX_V, len(moving))
```

Listing B.2.1: Python implementation of the the Random Waypoint model using the Numpy package.

B.3 Truncated Lévi Walk Mobility Model

```
import numpy as np
from numpy.random import rand

# define a Uniform Distribution
U = lambda MIN, MAX, NR_SAMPLES: rand(NR_SAMPLES) * (MAX - MIN) + MIN
# define a Power Law Distribution X = [(x1^(n+1) - x0^(n+1))Y + x0^(n+1)]^1/(n+1)
P = lambda n, x0, x1, NR_SAMPLES: ((x1 ** (n+1.) - x0 ** (n+1.)) * rand(NR_SAMPLES)
                                        + x0 ** (n+1.)) ** (1./(n+1.))
NR_NODES = 100
MAX_X, MAX_Y = 100, 100
NR_STEPS = 1000000
MAX_WT = 50.
x = U(0, MAX_X, NR_NODES)
y = U(0, MAX_Y, NR_NODES)
flight_distance = P(-2., 1., MAX_X, NR_NODES)
velocity = np.sqrt(flight_distance)/10.
theta = U(0, 2*np.pi, NR_NODES)
wait_time = np.zeros(NR_NODES)

for i in xrange(NR_STEPS):
        x = x + velocity * np.cos(theta)
        y = y + velocity * np.sin(theta)
        # node bounces on the margins and changes direction
        b = np.where(x<0)[0]
        x[b] = - x[b]; theta[b] += np.pi
        b = np.where(x>MAX_X)[0]
        x[b] = 2*MAX_X - x[b]; theta[b] += np.pi
        b = np.where(y<0)[0]
        y[b] = - y[b]; theta[b] += np.pi
        b = np.where(y>MAX_Y)[0]
        y[b] = 2*MAX_Y - y[b]; theta[b] += np.pi
        # update info for arrived nodes
        flight_distance = flight_distance - velocity
        arrived = np.where(np.logical_and(velocity>0., flight_distance<=0.))[0]
        velocity[arrived] = 0.
        wait_time[arrived] = P(-2., 1., MAX_WT, len(arrived))
        # update info for paused nodes
        paused = np.where(velocity==0.)[0]
        wait_time[paused] -= 1.
        # update info for moving nodes
        moving = np.where(np.logical_and(velocity==0., wait_time<0.))[0]
        theta[moving] = U(0, 2*np.pi, len(moving))
        flight_distance[moving] = P(-2., 1., MAX_X, NR_NODES)
        velocity[moving] = np.sqrt(flight_distance[moving])/10.
```

Listing B.3.1: Python implementation of the the Truncated Lévi Walk model using the Numpy package.