Abstract

Title of dissertation: TOWARDS PROACTIVE CONTEXT-AWARE COMPUTING AND SYSTEMS

Preeti Bhargava
Doctor of Philosophy, 2015

Dissertation directed by: Professor Ashok Agrawala
Department of Computer Science

A primary goal of context-aware systems is delivering the right information at the right place and right time to users in order to enable them to make effective decisions and improve their quality of life. There are three key requirements for achieving this goal: determining what information is relevant, personalizing it based on the users' context (location, preferences, behavioral history etc.), and delivering it to them in a timely manner without an explicit request from them. These requirements create a paradigm that we term as "Proactive Context-aware Computing".

Most of the existing context-aware systems fulfill only a subset of these requirements. Many of these systems focus only on personalization of the requested information based on users' current context. Moreover, they are often designed for specific domains. In addition, most of the existing systems are reactive - the users request for some information and the system delivers it to them. These systems are not proactive i.e. they cannot anticipate users' intent and behavior and act proactively without an explicit request from

them. In order to overcome these limitations, we need to conduct a deeper analysis and enhance our understanding of context-aware systems that are generic, universal, proactive and applicable to a wide variety of domains.

To support this dissertation, we explore several directions. Clearly the most significant sources of information about users today are smartphones. A large amount of users' context can be acquired through them and they can be used as an effective means to deliver information to users. In addition, social media such as Facebook, Flickr and Foursquare provide a rich and powerful platform to mine users' interests, preferences and behavioral history. We employ the ubiquity of smartphones and the wealth of information available from social media to address the challenge of building proactive context-aware systems. We have implemented and evaluated a few approaches, including some as part of the Rover framework, to achieve the paradigm of Proactive Context-aware Computing. Rover is a context-aware research platform which has been evolving for the last 6 years.

Since location is one of the most important context for users, we have developed 'Locus', an indoor localization, tracking and navigation system for multi-story buildings. Other important dimensions of users' context include the activities that they are engaged in. To this end, we have developed 'SenseMe', a system that leverages the smartphone and its multiple sensors in order to perform multidimensional context and activity recognition for users. As part of the 'SenseMe' project, we also conducted an exploratory study of privacy, trust, risks and other concerns of users with smart phone based personal sensing systems and applications.

To determine what information would be relevant to users' situations, we have developed 'TellMe' - a system that employs a new, flexible and scalable approach based on

Natural Language Processing techniques to perform bootstrapped discovery and ranking of relevant information in context-aware systems. In order to personalize the relevant information, we have also developed an algorithm and system for mining a broad range of users' preferences from their social network profiles and activities. For recommending new information to the users based on their past behavior and context history (such as visited locations, activities and time), we have developed a recommender system and approach for performing multi-dimensional collaborative recommendations using tensor factorization.

For timely delivery of personalized and relevant information, it is essential to anticipate and predict users' behavior. To this end, we have developed a unified infrastructure, within the Rover framework, and implemented several novel approaches and algorithms that employ various contextual features and state of the art machine learning techniques for building diverse behavioral models of users. Examples of generated models include classifying users' semantic places and mobility states, predicting their availability for accepting calls on smartphones and inferring their device charging behavior. Finally, to enable proactivity in context-aware systems, we have also developed a planning framework based on HTN planning. Together, these works provide a major push in the direction of proactive context-aware computing.

TOWARDS PROACTIVE CONTEXT-AWARE COMPUTING AND SYSTEMS

by

Preeti Bhargava

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Ashok Agrawala, Chair/Advisor
Professor Doug Oard, Dean's Representative
Professor Atif Memon
Professor Ben Shneiderman
Professor Michelle Mazurek

Dedication

Dedicated to my most beloved, my Guiding Light, the Supreme God ***Shri Krishna***.

Also dedicated to my parents, my husband and my sister.

Acknowledgments

There have been many who have inspired me, influenced me, guided me, and advised me in various stages of my life. I owe my gratitude to all the people who have helped me mould myself into who I am today. I sincerely thank those who made this dissertation possible and because of whom my graduate experience has been very enriching.

I am who I am because of my parents Lt. Col. Rishi Bhargava and Shashi Bhargava as well as my sister Swati Bhargava. They have been with me at every stage of my life. Without their support and encouragement, I wouldn't be where I am. My husband, Dr. Nitesh Shroff, has always been by my side through all the ups and downs in my career and my PhD. He has been my constant pillar of support - advising me and motivating me to do my best.

I'd like to express my heartfelt gratitude to my advisor, Professor Ashok Agrawala for giving me an opportunity to work on extremely challenging and interesting projects over the past five years and for encouraging me to think outside the box. He has always made himself available to help me with the minutest of the problems, both academically and personally. I consider myself very privileged to have worked with a person of this stature. I would also like to thank Prof. Atif Memon, Prof. Ben Shneiderman, Prof. Doug Oard and Prof. Michelle Mazurek for agreeing to be on my dissertation committee. I truly value their suggestions and advice. Prof. Don Perlis was a mentor to me in my initial months at UMD. He has inspired and encouraged me in many ways and I am grateful to him for that. I should also thank the CS department staff members, especially Jennifer Story, whose efforts make the lives of graduate students easy and comfortable.

MIND Lab has been like a second home to me. We have had a very cordial, understanding and wonderful group. I would like to thank Dr. Shivsubramani Krishnamoorthy and Nick Gramsky who have worked with me on building the next generation Rover platform. We have spent a lot of time sharing our ideas, views, brainstorming etc. I would also like to thank Dr. Matthew Mah, Dr. James Lampton and Dr. Neha Gupta for their useful and valuable suggestions pertaining to my papers and work. I have learned a lot from them. I am grateful to Oliver Brdiczka (my mentor at PARC), Thomas Phan, Jiayu Zhou and Doug Terry (my mentors at Samsung Research America) for providing me with the invaluable opportunity to intern with their teams and allowing me to work on exciting new projects.

I would like to thank Aditya Karkada Nakshathri, Anilesh Shrivastava and Zhe Cui who have contributed to some of my PhD projects. I have had a fun time organizing several events for women in our dept. with Dr. Jan Plane (Director, MCWIC), Aishwarya Thiruvengadam and Anusha Gururaj.

I should not fail to thank Arijit Biswas, Sravanthi Bondugula, Rajesh Chitnis, Tom Chan, Jayanta Mondal, Varun Nagaraja, Manish Purohit, Gregory Sanders, Udayan Khurana, Kapil Anand, Kaustubh Jain, Raviteja Vemulapalli, Praneeth Boda, Ashish Shrivastava, Rincy Matthew, Abdul Quamar, Vaibhav Singh, and many more people for the bright moments that I have spent with them during my Ph.D years here at UMD.

Above all, it is *Bhagwan Shri Krishna* who has guided me in every part of my work and, more importantly, in my life. HE has been a friend and teacher to me all along. HE has given me strength and courage to persist and persevere through hard times. He has provided me with direction when I was lost. I offer my prayers and salutations to HIM.

Table of Contents

List of Tables

List of Figures

Chapter 1: **Introduction**

A context-aware system is a system that keeps track of and employs the context of a user to provide relevant information and services to him, where relevancy depends on the user's task [1, 2]. A user's context can include location, physical activity, emotional state, interests and preferences. Thus, a fundamental principle underlying context-aware systems is delivering "the right information at the right place and the right time" i.e. *relevant*, *personalized* and *timely* information to users. This principle entails the following capabilities for a context-aware system:

1. Determining relevant information - This capability is the cornerstone of context-aware computing. In today's world, with the abundance of information available to us, information overload can easily happen. Hence, it is imperative that the system retrieves and displays only that information which is relevant to the user's task at hand.

2. Personalization - This is achieved by acquiring a user's context (needs, preferences, etc.) through implicit or explicit means and using it to filter the relevant information.

3. Timeliness - The system can achieve timely information delivery by providing the

personalized and relevant information to the user at a time when he needs it and can act upon it.

For instance, users who intend to get a medical test done would benefit from relevant information such as suggestions for hospitals and laboratories. Furthermore, this information should be personalized according to their location and needs. Finally, the context-aware system should provide this information to them in a timely manner in order to help them make an informed decision and save their time and effort.

These requirements create a paradigm that can be termed as **Proactive Context-aware Computing**. Most of the existing context-aware systems fulfill only a subset of these requirements. A number of existing systems such as those described in [3–5] focus only on personalization of the requested information based on the users' interests/preferences or their limited context (such as time or location). Moreover, they are specific to domains such as museums and healthcare. In addition, most of these systems tend to be reactive or 'pull' based. The users request or query for some information and the system responds with the requested information. None of these systems are proactive in that they do not anticipate the users' intent or behavior in order to proactively 'push' relevant information to them.

This paradigm poses several key challenges pertaining to:

- Recognizing and anticipating users' current and future context, activities and situations,
- Determining information relevant to those situations, and further personalizing this relevant information,

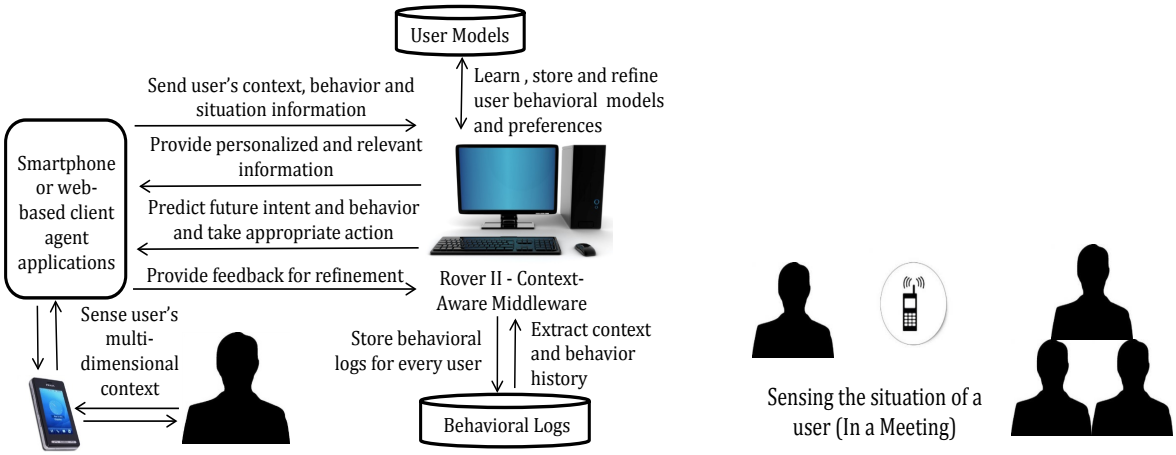
- Providing the users with the personalized and relevant information in a timely manner.

1.1 Our vision of a proactive context-aware system

To achieve this paradigm, we present our vision of a proactive context-aware system. As shown in Figure 1.1(a), we envision a proactive context-aware system which, in addition to storing and retrieving context, will:

- Sense the users' context and situation including location, behavior and activities,
- Determine what information would be relevant to their situation,
- Learn the users' preferences in order to personalize the relevant information,
- Learn the users' behavioral patterns and activities from their behavior and context history,
- Build and store models of users' behavior and utilize them to predict and anticipate the next action of the users,
- Provide the users with personalized and relevant results in a timely manner and take actions on their behalf without being asked explicitly, and
- Refine the stored user models based on the users' feedback as their context, behavior and situations are highly dynamic.

Such a system constitutes what Mark Weiser referred to as disappearing or invisible technology as that which is 'so natural that we use it without even thinking about it' [6].



(a) Our vision of a proactive context-aware system

(b) Situation of a user - In a Meeting

Figure 1.1: Our vision of a proactive context-aware system and the system sensing the situation of a user via a client agent application.

Such a system exhibits three main capabilities:

1. Sensing ability - As mentioned earlier, one of the main goals of a proactive context-aware system is to sense the users' context and situation including location, behavior and activities. Thus, it needs to answer the question: Who, What, When and Where? [2]. To answer this, these systems need to gather multi-dimensional contextual information of types such as:

    - behavioral - users' current indoor or outdoor location or activity,
    - technical - the kind of device that is being used and the applications that are running on it, and
    - environmental - temperature, and climate etc.

Moreover, this information should be acquired without placing undue burden on the

users. As a result, automated sensing ability is highly desirable for any information system powered by proactive context-aware computing. In addition, this ability to sense users should be embedded in devices that they can carry around without effort and the user interaction should be kept to a minimum so that it is unobtrusive. The ubiquitous smartphone, with its multitude of sensors and capabilities, has become the best choice for this purpose.

Today's smart phones come equipped with an increasing range of sensing, computational, storage and communication capabilities that have enabled sensing and tracking applications to emerge across a wide variety of applications areas, such as, personal healthcare, environmental monitoring, social networking etc. A key challenge of mobile phone sensing is to process raw sensory data from multiple sensors (e.g. GPS, accelerometer, temperature and pressure sensors, microphone, bluetooth, ambient light sensor, camera, proximity sensor, gyroscope, Wi-Fi etc) in order to infer higher level activities and context in real-time and in a robust, generic and independent manner using machine learning techniques such as classification and clustering.

Figure 1.1(b) shows the proactive context-aware system sensing the user's situation 'In a meeting' via a client agent application.

2. Learning ability - The proactive context-aware system should store users' sensed activities, context and behavioral logs and extracts their context and behavioral history from it. This can be done by logging all events and activities for every entity and indexing logs according to them. For instance, a log entry en_1 would be of the

form - $< e, a, t >$ where e is an entity, a is the activity performed by the entity and t is the time at which it was done. E.g., $< xyz, walking, 5pmEST >$.

The system would then utilize the users' activities and behavior from their context history to learn their preferences and behavioral patterns in order to predict their future behavior. This involves applying machine learning and data mining algorithms to model user preferences and deriving behavioral rules. For instance, the following rule could be derived from the log of user's daily activities.

currentActivity$<$ xyz,walking $> \wedge$ currentTime $<$ 5 pm EST $> \rightarrow$ currentState$<$ device, silent $>$

This rules states that if user 'xyz' is walking in the evening, then put his/her device on silent mode.

3. Proactivity - Proactivity is the ability of the system to act on the users' behalf in anticipation of future goals or problems without explicit requests. Based on the current context of users and the stored behavioral models and patterns for them, the proactive context-aware system can trigger itself and predict their future intent and behavior. Furthermore, it can take proactive actions based on the predicted behavior to assist the users and save their time and effort.

1.2 Contributions

This dissertation makes several contributions:

1. It introduces the novel paradigm of **Proactive Context-aware Computing and Systems** and describes the requirements and challenges that this paradigm entails. It also presents a vision to realize it.

2. It presents the design and implementation of a proactive context-aware middleware which has been developed by enhancing an existing context-aware middleware, Rover II, and which exhibits the capabilities required by proactive context-aware computing.

3. It presents various systems and approaches that we have developed, some as part of the enhanced Rover II middleware, in order to realize the vision of proactive context-aware computing. These form the building blocks of a proactive context-aware system and include:

    - Locus [7, 8] - a robust and calibration-free indoor localization, tracking and navigation system for multi-story buildings,
    - SenseMe [9] - a system that leverages a user's smartphone and its multiple sensors in order to perform continuous, on-device, and multi-dimensional context and activity recognition,
    - TellMe [10] - a novel, general and flexible framework for bootstrapped discovery and ranking of heterogeneous relevant services and information in context-aware systems,
    - An unsupervised algorithm and system [11], that models users' interests from their Facebook profiles and activities, for personalizing the relevant informa-

tion,

- A system and an approach [12] for leveraging the aggregated sensed context (context history) of several users in order to recommend new information to a particular user via collaborative filtering techniques,
- A Learning Engine [13], within the infrastructure of the Rover II middleware, which implements several novel approaches and algorithms that employ various contextual features and state of the art machine learning techniques for building diverse behavioral models of users,
- A Planning Framework [14], within the infrastructure of the Rover II middleware, which employs HTN Planning and demonstrates the feasibility of enabling proactivity in a context-aware system.

1.3 Organization of the Dissertation

The dissertation is organized into several chapters. In Chapter 2, we describe the existing context-aware middleware, Rover II. In Chapter 3, we explain the enhancements made to Rover II (in the form of new components and subsystems) as well as other approaches that we have developed in order to achieve the vision of proactive context-aware computing. Chapters 4 - 11 explain the various approaches and subsystems in detail. We finally conclude and discuss our contributions and future work in Chapter 12.

Chapter 2: **Existing context-aware middleware - Rover II**

A middleware is an enabling layer of software that resides between the application program and the networked layer of heterogeneous platforms and protocols. It decouples applications from any dependencies on the plumbing layer that consists of heterogeneous operating systems, hardware platforms and communication protocols. [15]

Rover II [16, 17] is an integration and fusion middleware that caters to the development of context-aware applications. It provides the means to store and retrieve contextual information as well as facilitates delivery of relevant services to applications to more effectively use the contextual information.

2.1 Rover Context Model

RoCoM [16] is an ontological model built around four Primitives. These Primitives are the building blocks of every context-aware system built on this model. Each piece of contextual information is associated with at least one of these primitives. We have identified these primitives depending on the role they play in defining and influencing the context of a given situation. We discuss each primitive now:

- *Entity*: An individual element of the context-aware system, such as a person, a place, an organization, or a computing device. The properties or attributes of an

entity constitute its context. An entity can be classified as physical or virtual; permanent or transient; single or group. Typically, entities would be specific to a situation. For instance, in the case of an accident situation, an entity involved can be a person, a place, a car, a building, etc.

- *Event*: An event has one or more entities involved in it and can consist of one or more activities. Every event will have a start time and/or end time, along with a duration, associated with it. It has its own properties or context, and inherits the context of involved entities and activities. An example of an event would be a road accident or simply a request for dining information. An event triggers the context-aware system and sets the implicit desired outcome or the *goal*, whether its long term or short term. Events play the prime role in defining the initial context. Since the event acts as the trigger of the situation, the context of the event influences how the situation is handled. For example, consider the case where an emergency call is made when a road accident happens. The person involved in the accident may have high blood sugar and blood pressure, which would define the context based on which services have to provided to him. But if in the course of the accident, he had a severe head injury, these conditions may become secondary. A medical center handling emergency casualties becomes relevant then. Thus, the event triggers the situation and contributes its context to it.

- *Activity*: An activity occurs for a fixed time and causes a change in context. Every activity is driven by a goal. The goal ceases to exist once the sequence of activities aiming to achieve it have been performed. There can be interaction or coordination

between different activities to achieve the common goal. Every activity has some executable or action associated with it, which is required to carry out the operations necessary for it to achieve the goal. An activity is performed by one or more entities and derives a part of its context from those entities. It also influences the context of the entities involved in it. For instance, when an emergency call is made on M-Urgency [18], the context of the emergency dispatcher changes from "available" to "busy" when the call is accepted. Every following activity will keep changing the context of the involved entities.

- *Relationship*: A relationship describes how two primitives are related and has its own context. Relationships can be derivative or transitive, i.e. if primitive A has a relationship with primitive B, which in turn has a relationship with primitive C, then A may also have a relationship with C. The relationship that an entity shares with another entity influences the context of the situation. The creation, change or end of a relationship between entities redefines the context. For example, when the relationship between dispatcher and an emergency responder is created, the context of the responder changes from "available" to "assigned". And if he/she was the last responder available, the system should behave differently when further calls are made to the him/her.

2.2 Rover II Ecosystem and Architecture

Figure 2.1 provides a high level layout of the Rover II ecosystem architecture. An ecosystem here is a logical view of how different entities interact with the context-aware system.

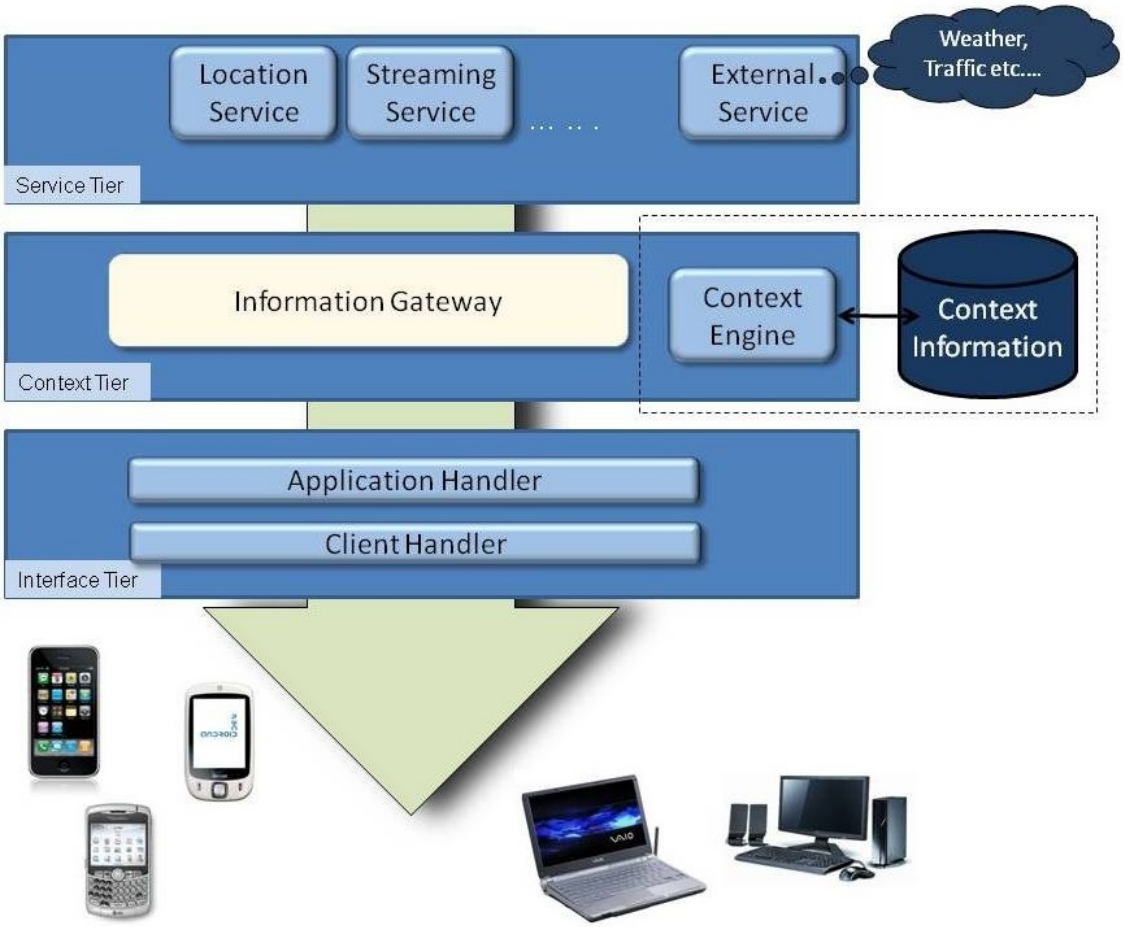


Figure 2.1: Rover 2.0 ecosystem

This architecture is centered on the information flow between the various sources and the end application or users. The architecture is organized in three tiers:

- **Service Tier** This tier consists of external or third party services with which the system communicates to obtain structured/unstructured information. Figure 2.1 shows some of the services that have been implemented in the system to date. The location service resolves GPS coordinates or Wi-Fi signals to a location with the local map system; the streaming service enables any application to establish an audio/video stream with the system; the external service enables incorporation of external infor-

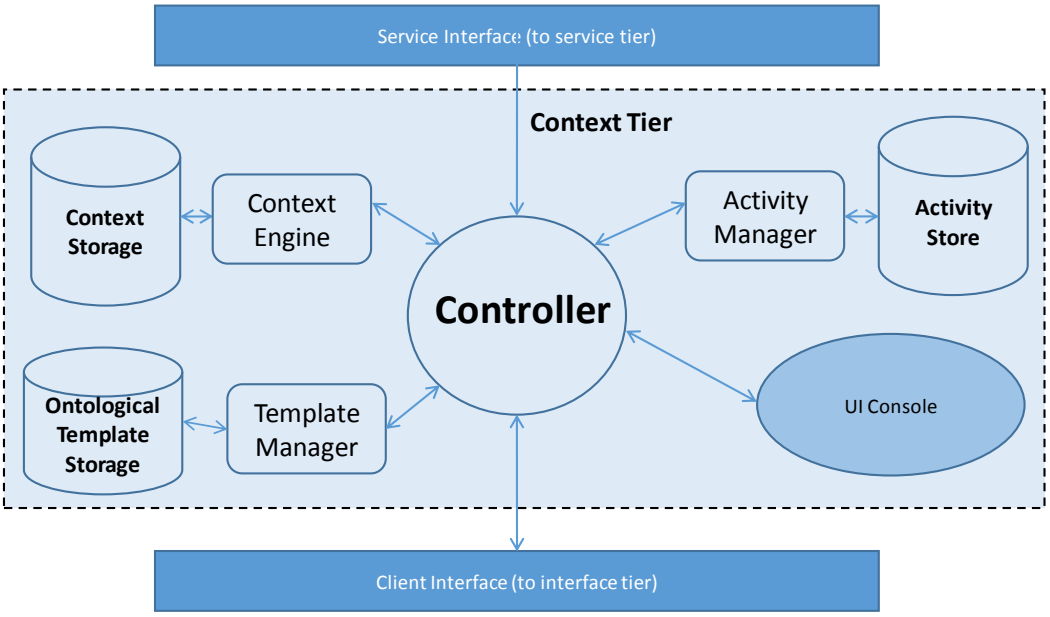


Figure 2.2: Rover II architecture

mation to add to the clarity of the context, e.g. the weather information, restaurant menus, traffic information, etc.

- **Interface Tier** This tier defines the application interface for clients/applications with the system.

- **Context Tier** This is the most critical part of the system which makes use of the contextual information of each user to mediate the flow of information from the source to the client/user. Based on the context of the user, along with the general context of the environment, the information is filtered, consolidated and/or rearranged, and some services are enabled.

Figure 2.2 depicts the detailed design of the Rover 2.0 Context Tier. The three layers are:

1. *Service Interface* - This layer interfaces with the third party services in the Service Tier and that could be Web Services, REST based services, etc.

2. *Client Interface* - This layer interfaces with the client applications. The interaction can be through HTTP, TCP sockets or through Remote Procedure Calls.

3. The *Rover Core* - This layer forms the core layer of the Rover Server. It consists of several modules that handle and propagate context:

   (a) The *Controller* is the main module, which schedules different processes running inside the Rover Core and passes around the context between modules.

   (b) The *Context Store* contains the aggregation of context for every instance of a primitive such as an entity or event, etc.

   (c) The *Template Store* contains the primitives and context templates in ontological form.

   (d) The *Context Engine* determines the relevant context for each primitive based on predefined templates stored in the Template Store.

   (e) The *Template Manager* is used to retrieve the context and primitives templates from the Template Store.

   (f) The *Activity Manager* is responsible for the execution of all the activity(s) that form an event, until the Goal of the event or activity has been achieved.

   (g) The *UI Console* is the user interface for a human entity, for making decisions based on the contextual information provided by the system.

Chapter 3: **Enhanced Rover II**

This chapter explains the various approaches and systems that we have developed, including some as part of the enhanced Rover II ecosystem and architecture, in order to achieve the vision of proactive context-aware computing.

3.1 Rover II - New Architecture and enhancements

Figure 3.1 shows the new architecture for Rover II. Here, we have retained the core structure of the existing middleware while adding components and functionalities, that achieve proactive context-aware computing, such as:

1. A Graph DB based Context Store - This component stores the current context for an entity such as a user. Even though ontologies (used in Rover II) are expressive and promote reuse and interoperability between applications, they often become too cumbersome and intractable for use in practical systems. Hence, instead of ontologies, we have adopted a more flexible approach that uses a NoSQL graph database for storing users' context.

2. Context Interface - This component interfaces with the Context Store and is responsible for storing and retrieving the current context for a user.

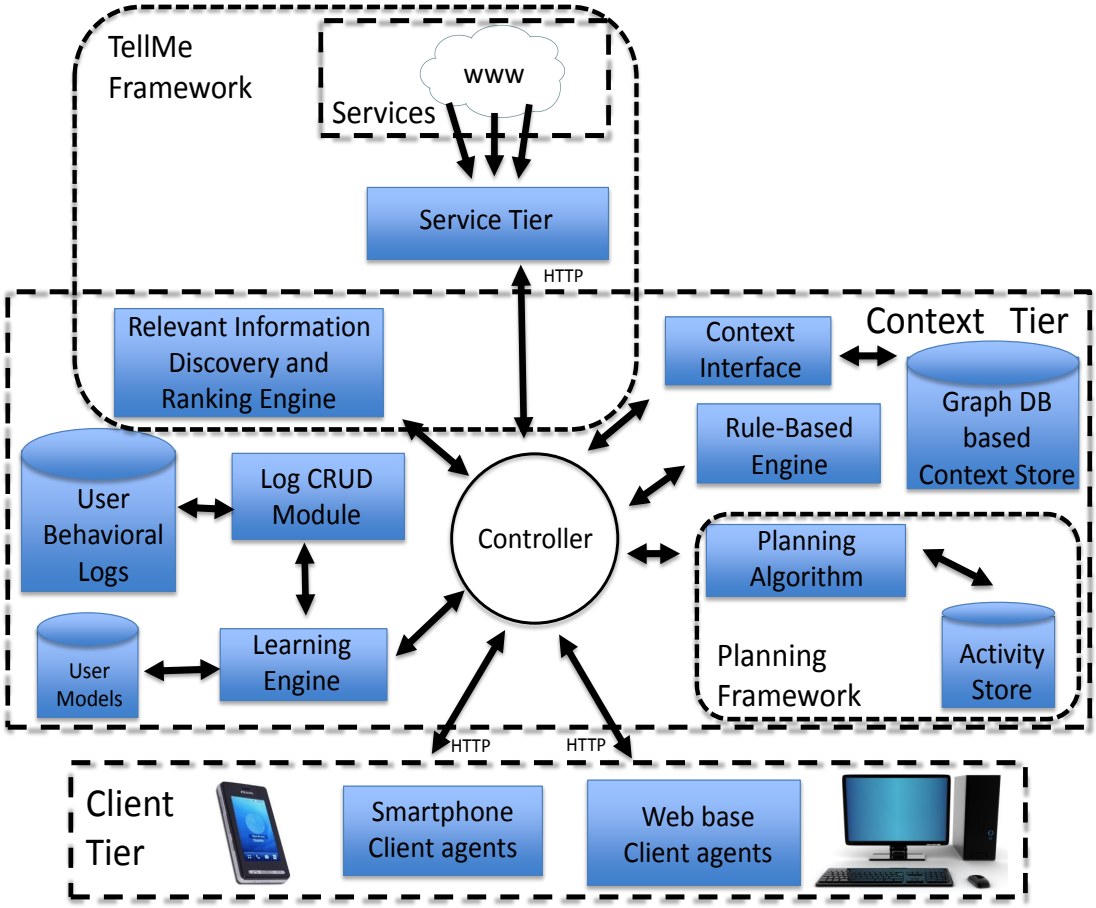


Figure 3.1: Rover 2.0 new architecture

3. A smartphone client agent application called SenseMe for performing multi-dimensional context and activity recognition.

4. Relevant Information Discovery and Ranking Engine - This component is responsible for dynamically determining information relevant to the users' context or situations. The Relevant Information Discovery and Ranking Engine as well as the Service Tier together form a part of the TellMe framework.

   In Rover II, the Context Engine contained predefined templates (created by the system designer) for determining information relevant to a user's situation. However,

as we discuss later, such an approach is not scalable and flexible and cannot be dynamically adjusted.

5. Learning engine - This component is responsible for learning and storing behavioral models of users from their context history (which has been extracted from their behavioral logs). It would also refine these models periodically based on user feedback or changed context.

6. User models store - This component stores the learned user models.

7. Rule based engine - This component is responsible for generating and storing rules derived from user behavioral models.

8. Activity Store - This component stores the actions that can be executed by the system on the users' behalf.

9. Planning Engine - This component is responsible for generating a plan that contains a sequence of actions that the middleware can execute in order to act proactively on behalf of the user. The Planning Engine and the Activity Store together form the Planning Framework of the Rover II context-aware middleware.

3.2 Building blocks of a proactive context-aware system

We now explain the various systems that we have developed which, when integrated together with the components of the enhanced Rover II middleware, form the building blocks of a proactive context-aware middleware:

3.2.1 Determining the indoor location of a user - Locus

Since location is often considered as one of the most significant context of a user, it is important for any context-aware system to determine it accurately. The most popular technology for localization is GPS, which provides worldwide coverage and accuracy of a few meters depending upon satellite geometry and receiver hardware. Its major shortcoming is that it is reliable only in outdoor environments with direct visibility to at least four GPS satellites.

For indoor environments, alternative technologies are required. A fundamental goal of indoor localization technology is to achieve the milestone of combining minimal cost with accuracy sufficient enough for general consumer applications. To achieve this, current indoor positioning systems need either extensive calibration or expensive hardware. Moreover, very few systems built so far have addressed floor determination in multi-story buildings. To address this, we have developed a Wi-Fi based indoor localization, tracking and navigation system for multi-story buildings called Locus [7, 8] (Chapter 4). Locus determines a device's floor as well as location on that floor by using existing knowledge of infrastructure, and without requiring any calibration or proprietary hardware. It is an inexpensive solution with minimum setup and maintenance expenses, is scalable, readily deployable and robust to environmental changes. Experimental results in three different buildings spanning multiple floors show that it can determine the floor with 95.33% accuracy and the location on the floor with an error of 6.49m on an average in real life practical environments. We also demonstrate its utility via two location-based applications for indoor navigation and tracking in emergency scenarios.

3.2.2 Continuous, on-device, and multi-dimensional context and activity recognition - SenseMe

In order to make context-aware systems more effective and provide timely, personalized and relevant information to a user, the context or situation of the user must be clearly defined along several dimensions. To this end, the system needs to simultaneously recognize multiple dimensions of the user's situation such as location, physical activity etc. in an automated and unobtrusive manner. To achieve this, we have developed SenseMe [9] (Chapter 5) - a system that leverages a user's smartphone and its multiple sensors in order to perform continuous, on-device, and multi-dimensional context and activity recognition. It recognizes five dimensions of a user's situation in a robust, automated, scalable, power efficient and non-invasive manner to paint a context-rich picture of the user. We evaluate SenseMe against several metrics with the aid of 2 two-week long live deployments involving 15 participants. We demonstrate improved or comparable accuracy with respect to existing systems without requiring any calibration or input.

As part of this work, we also conducted an exploratory study [19] of privacy, trust, risks and other concerns of users with smart phone based personal sensing systems and applications. The results of this study are presented in Chapter 6.

3.2.3 Bootstrapped discovery and ranking of relevant services and information in context-aware systems - TellMe

A context-aware system uses context to provide relevant information and services to the users, where relevancy depends on the users' situation. This relevant information could include a wide range of heterogeneous content. Many existing context-aware systems determine this information based on pre-defined ontologies or rules. In addition, they rely on users' context history to filter it. Moreover, they often provide domain-specific information. Such systems are not applicable to a large and varied set of user situations and information needs, and may suffer from cold start for new users. We address these limitations and propose a novel, general and flexible approach for bootstrapped discovery and ranking of heterogeneous relevant services and information in context-aware systems. We present the design and implementation of a framework called TellMe [10] (integrated with the Rover II context-aware middleware) which employs four variations of a base algorithm to rank candidate relevant services, and the information to be retrieved from them, based on the Semantic Relatedness between the information provided by the services and the user's situation description. We conduct a live deployment with 14 subjects to evaluate the efficacy of our algorithms. We demonstrate that they have strong positive correlation with human supplied relevance rankings and can be used as an effective means to discover and rank relevant services and information. We also show that our approach is applicable to a wide set of users' situations and to new users without requiring any user interaction history. Chapter 7 explains TellMe and addresses the problem of bootstrapped discovery and ranking of heterogeneous relevant services and information

in context-aware systems.

3.2.4 Personalizing relevant information by unsupervised modeling of users' interests from their Facebook profiles and activities

Once the relevant information has been determined, it should be personalized based on users' individual preferences. These preferences can be either obtained explicitly by asking users to create user interest profiles or by modeling these profiles in an automated manner which is preferable and less burdensome. In today's world, social networks such as Facebook or Twitter provide users with a powerful platform for interest expression and can, thus, act as a rich content source for automated user interest modeling. This, however, poses significant challenges because the user generated content on them consists of free unstructured text. In addition, users may not explicitly post or tweet about everything that interests them. Moreover, their interests evolve over time.

To address these challenges, we propose a novel unsupervised algorithm and system [11] (Chapter 8) that models a broad range of users' explicit and implicit interests from their social network profile and activities without any user input. We perform extensive evaluation of our system, and algorithm, with a dataset consisting of 488 active Facebook users' profiles and demonstrate that it can accurately estimate users' interests in practice.

3.2.5 Recommending new information to the user via multi-dimensional collaborative recommendations

The users' sensed context, such as their location and activities, is aggregated over a period of time to generate their context history. The context history of several users can be leveraged to recommend new information to a particular user via techniques such as collaborative filtering. Previous work has mainly explored recommendations along one dimension such as location; however, users would also benefit from recommendations for activities in which to participate at those locations along with suitable times and days. Thus, systems that provide collaborative recommendations involving multiple dimensions such as location, activities and time would enhance the overall experience of users. The relationship among these dimensions can be modeled by higher-order matrices called tensors which are then solved by tensor factorization. However, these tensors can be extremely sparse. To address this, we present a system and an approach [12] (Chapter 9), for performing multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. We formulate an objective function which simultaneously factorizes coupled tensors and matrices constructed from heterogeneous data sources. We evaluate our system and approach on large-scale real world data sets consisting of 588,000 Flickr photos collected from three major metro regions in USA. We compare our approach with several state-of-the-art baselines and demonstrate that it outperforms all of them.

3.2.6 Modeling Users' Behavior from their Context History

A large volume of research in mobile and ubiquitous systems has been devoted to using data sensed from users' smartphones for inferring their current high level context and activities. However, mining of users' diverse longitudinal behavioral patterns from this data, which can enable exciting new context-aware applications and services, has not received much attention. In this work, we focus on learning diverse patterns from large-scale data collected from users' smartphones [13]. We utilize these patterns to help identify a variety of users' behaviors, habits, and daily life places and activities. To address this, we use their context history to build different types of user behavioral models. The Learning Engine (see Figure 3.1) mines this history for building user behavioral models using a combination of machine learning and data mining algorithms such as Decision Trees, k Nearest Neighbor (kNN) classifier, k Means clustering, Hidden Markov Models, and Association Rule Mining.

Chapter 10 explains the design and implementation of the Learning Engine of the Rover II middleware framework. This engine implements several novel approaches and algorithms that employ various contextual features and state of the art machine learning techniques for building diverse behavioral models of users. Examples of generated models include classifying users' semantic places and mobility states, predicting their availability for accepting calls and inferring their device charging behavior. We evaluate our work on large-scale real-world smartphone data of 200 users, from the DeviceAnalyzer dataset, consisting of 365 million data points. We show that our algorithms and approaches can model user behavior with high accuracy and outperform existing ap-

proaches.

3.2.7 Proactively taking actions on behalf of the user

The context-aware middleware can utilize the behavioral models of users in order to predict their future behavior and proactively take actions on their behalf. These actions could include tasks such as sending emails, calling a phone number, changing the device mode (say from silent to ringer) based on their situation, installing or starting an application on the user's device etc. To this end, we present a new paradigm for enabling proactivity in context-aware middleware systems by means of a Planning Framework based on HTN planning. We present the design of a Planning Framework [14] within the infrastructure of the Rover II middleware in Chapter 11. We implement this framework and demonstrate its utility with several use cases. We also highlight the benefits of using such a framework in dynamic ubiquitous systems.

Chapter 4: **Locus: A indoor localization, tracking and navigation system for multi-story buildings**

Since location is often considered as one of the most significant context of a user, it is important for any context-aware system to determine it accurately. The most popular technology for localization is GPS, which provides worldwide coverage and accuracy of a few meters depending upon satellite geometry and receiver hardware. Its major shortcoming is that it is reliable only in outdoor environments with direct visibility to at least four GPS satellites.

For indoor environments, alternative technologies are required. A fundamental goal of indoor localization technology is to achieve the milestone of combining minimal cost with accuracy sufficient enough for general consumer applications. To achieve this, current indoor positioning systems need either extensive calibration or expensive hardware. Moreover, very few systems built so far have addressed floor determination in multi-story buildings. In this chapter, we present Locus [7, 8] - a robust and calibration-free indoor localization, tracking and navigation system for multi-story buildings that addresses these limitations.

4.1 Introduction and Related Work

A user's location has become increasingly important for mobile computing, providing the basis for services such as navigation and location-aware advertising. The most popular technology for localization today is GPS, which provides worldwide coverage and accuracy of a few meters depending upon satellite geometry and receiver hardware. However, its major shortcoming is that it is reliable only in outdoor environments with direct visibility to at least four GPS satellites.

For indoor environments, alternative technologies are required. A fundamental goal of indoor location technology is to achieve minimal cost with accuracy sufficient enough for general consumer applications. A low-cost indoor positioning system should be inexpensive, both to install and maintain. It should require only available consumer hardware to operate, and its accuracy should be room-level or better. To achieve this level of accuracy, current systems need either extensive calibration or expensive hardware. Most of them are based primarily on time or signal strength information. Time-based systems require hardware support for timestamping that is not available in consumer products. A third alternative, angle-of-arrival information is useful in outdoor environments but is not generally helpful indoors due to obstructions and reflections. For more information on angle-of-arrival or time-based approaches, refer to the survey on indoor positioning by Liu et al. [20].

The use of wireless received signal strength indicators (RSSI) values for localization of mobile devices is a popular technique due to the widespread availability of wireless signals and the relatively low cost of implementation. In its simplest version, it involves

the mobile device measuring the signal strengths of existing infrastructure points such as Wi-Fi access points (APs) or mobile phone base stations. The device then reports the origin of the strongest signal it can hear, or the accesspoint to which it is connected, as its location. This technique may be applied both to short range communications technologies such as RFID or Bluetooth as well as longer range technologies such as Wi-Fi or mobile phones. However, its performance is directly linked to the density of reference points. While there are several other techniques for indoor localization such as ultrasonic ranging (used in [21] and [22]), we focus our discussion mainly on Wi-Fi based techniques and systems.

The accuracy of signal strength approaches is improved to meter-level by fingerprinting techniques, such as those used in RADAR [23] or Horus [24], that use pre-measured fingerprinting radio maps. One such commercial solution is Ekahau [1], which achieves a high localization accuracy of 1 to 3 m but requires proprietary hardware. However, a major drawback of fingerprinting is the overhead of recording the radio map — a significant amount of human effort and money is required to record the signal strength at each desired location using a receiver. To overcome this limitation, organic positioning systems (which aim to eliminate these deficiencies by managing their own accuracy and obtaining input from users and other sources) are being developed. An example of such a system is WiFiSLAM [2] which exploits the power of crowdsourcing to mitigate the manual effort required in fingerprinting and reports a localization accuracy of 2.5m. Similarly, Ledlie et al. [25] utilize crowd sourcing to facilitate fingerprinting and build an organic positioning system called Mole. However, ordinary users who are not trained like surveyors to take

[1] http://www.ekahau.com/ [2] https://angel.co/wifislam

fingerprint scans can introduce noise in the collected radio maps. Moreover, the quality of the system's response depends on the willingness of users to contribute during its entire life cycle. If the user input rate varies with time, it can cause the system to become "unstable" [26].

More importantly, these works do not overcome another significant limitation of fingerprinting i.e. it is not robust to environmental changes. If the infrastructure or environment changes significantly — for instance, if the locations of APs are changed, furniture is moved around, the number of people occupying the closed space increases dramatically, or the test site is changed; the radio map must be remeasured to maintain performance [27]. Similar to WiFiSLAM, PlaceLab [28] uses crowdsourcing and war-driving to create radio maps of existing Wi-Fi/GSM APs. They aim to improve coverage at the expense of accuracy, achieving a median accuracy of 15-20m. However, war-driving can be extremely time-consuming. An ideal solution should be cheap and scalable. Moreover, as mentioned earlier, if the environment changes, the radio map must be recalibrated.

Systems that don't use fingerprinting techniques can suffer from low accuracy. These include Active Campus [29], that uses an empirical propagation model and a hillclimbing algorithm to compute location with a location error of about 10 meters. Li [30] proposed a ratio based algorithm which produces median errors of roughly 20 feet (6.1 m) by predictively computing a map of signal strength ratios. Lim et al. [31] developed an automated system for collection of RSSI values between APs and between a client and an AP. They determine the client's location with an error of 3m. They do not create a radio map but require initial AP calibration and its modification for continuous data collection. Madigan et al. [32] encoded the dependency between RSSI and distance in the form of Bayesian

networks, and rely on statistical sampling to localize. This requires an infrastructure setup that can provide fingerprints (though the associated locations are not needed) from many mobile devices or from a single mobile device over a long period of time.

Lately, some commercial solutions such as iBeacon^{TM}[3] have emerged which enable indoor localization using Bluetooth Low Energy signals and employ a Bluetooth emitter as a landmark. The range of iBeacon varies from "immediate" ($<$ 0.5 m) through "near" (0.5 - 2 m) to "far" (2 - 30m). However, iBeacon requires external hardware such as bluetooth emitters to be installed in all buildings where indoor localization needs to be performed.

More important than the raw error in distance is the computation of the correct floor in indoor multi-story environments. Even a most modest error in altitude can result in an incorrect floor. This leads to a high location error as determined by human walking distance. Identifying the exact floor is also more difficult because there are multiple APs on each floor and a device can receive signals from APs spread across several floors.

Several existing systems either require user input for floor or do not address floor determination. Active Campus [29] has options for user adjustments to correct the computed floor while in [30], the testbed is assumed to be on a single floor. FTrack [33] uses an accelerometer to capture user motion data to determine floor but requires user input for initial floor. Moreover, it achieves a floor accuracy of $>$ 90% after 2 hours of experimentation.

Other systems have employed fingerprinting for floor determination. Skyloc ([34,35]) uses GSM based fingerprinting for floor and location determination. It has been tested in

[3] http://blog.nerdery.com/2013/11/nerdery-labs-ibeacon-experiments/

three multi-story buildings and achieves a best case correct floor estimation in 73% and an estimation within 2 floors in 97% of the experiments. This makes it unsuitable for real world applications. Other works such as Alsehly et al. [36] focus on floor determination alone (without localization on the floor) and achieve a floor accuracy of 86% by finding the k Nearest Neighbors in the radio map obtained by fingerprinting. Marques et al. [37] propose a fingerprinting-based technique that applies similarity functions and majority rules to determine the floor and location of the device. As mentioned earlier, fingerprinting based techniques suffer from several drawbacks including the manual and monetary overhead required to record the radio map, and non-robustness to environmental changes.

To address all these limitations, we present Locus — a Wi-Fi based indoor localization, tracking and navigation system for multi-story buildings. Locus employs a Floor and Location Determination algorithm to determine a device's floor, and location on that floor, in a multi-story building without any calibration. It achieves approximately room-level accuracy by using the existing knowledge of infrastructure but without the need of building radio maps by fingerprinting. It is an inexpensive solution suitable for localization with minimum setup or maintenance expenses. Hence, it has very low computational requirements. By avoiding the dependence on radio maps, it is readily deployable and robust to environmental changes unlike fingerprinting based techniques. Since it relies on existing mobile device capabilities, it does not require an instrumented set up or proprietary hardware. Hence, its hardware complexity is minimal.

An initial version of the Locus system and experiments in a testbed consisting of one multi-story building were presented in Bhargava et al. [7]. Our contributions are:

- We present a modified version of the underlying algorithm which improves the average location error (Section 4.3). We also present an approach for determining shortest path between two indoor locations on the same and different floors of a building.

- We present experiments conducted with commercial smart devices in a real life practical testbed consisting of three different buildings that span multiple floors and a total covered area of over 200,000 sq. feet (Section 4.4). Our experiments show that Locus can determine the floor with an accuracy of 95.33% ($>$ 90% floor accuracy across all the buildings) and the location on the floor with an error of 6.49m on an average.

- Our experiments also show that our system, as well as its underlying algorithm, can be universally used without any calibration and are scalable across different multi-story buildings with varying AP density.

- We demonstrate the robustness of Locus to environmental changes by conducting experiments at different times in a day (with varying number of people), on days spread across a month (with changes such as movement of furniture and locked doors) as well as before and after displacement of APs.

- We validate the utility of Locus via two location based applications for indoor navigation and tracking in emergency scenarios.

Our system has no requirements on the infrastructure other than the locations of landmarks, such as APs, which can be easily obtained. Such independence from the infras-

(a) A.V. Williams Building (b) Susquehanna Hall (c) Holzapfel Hall

Figure 4.1: The three multi-floor buildings in UMD campus which serve as test beds for Locus

tructure means that it can be applied in a wider range of scenarios. As explained in Section 4.5, it can enable indoor location based services and applications that require room-level accuracy. These include a tracking and navigation smartphone application that can download a map of a building the user is in, track his approximate current position on a floor map, and provide indoor navigation directions for destinations such as restrooms, offices, or conference rooms. It is also essential for situations like search and rescue operations in emergency scenarios where knowledge of the exact floor and location of the device or person on that floor is crucial for timely assistance. Though our system has a higher location error as compared to fingerprinting techniques, we believe it still serves as a competitive alternative particularly in scenarios where extensive fingerprinting is not feasible or affordable and it is preferable to trade a little accuracy for saved human effort.

4.2 Experimental Setup and Data Collection

The testbed for Locus is a real life practical environment consisting of three multi-story academic office buildings at the University of Maryland — the A.V. Williams Building

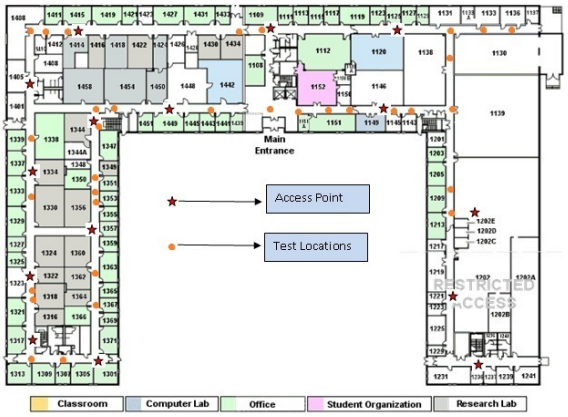
Bldg #	Name	Area (ft.^{2})	Floor	# of APs	AP density (AP per ft.^{2})	# of Test Points
1	AV Williams	152,099	1 2 3 4	15 19 20 10	$\frac{1}{2400}$	120
2	Susquehanna Hall	34,213	1 2 3 4	3 3 3 2	$\frac{1}{2850}$	75
3	Holzapfel Hall	22,197	1 2 3	5 6 4	$\frac{1}{1480}$	50

Table 4.1: Testbed Buildings' Characteristics

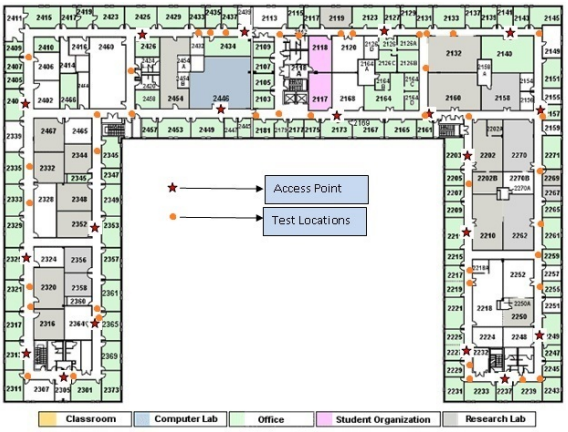
(Building # 1), Susquehanna Hall (Building # 2) and Holzapfel Hall (Building # 3). Figure 4.1 shows the three buildings. Table 4.1 shows the total covered area, number of floors,number of APs on each floor and the average AP density of each of the three buildings. As evident, the buildings have varied structure, floor area as well as AP density.

The AP model deployed in the campus and used in our experimental setup is the Cisco AIR-LAP1142N-A-K9. Each physical AP runs several virtual APs mapped to it. The last hexadecimal digit for the base MAC address (for a physical AP) is 0, for instance 00:xx:yy:zz:96:20, while for each virtual AP, it is varied, for example 00:xx:yy:zz:96:22, 00:xx:yy:zz:96:24, etc. Some MAC addresses of virtual APs were seen to repeat for 802.11a and 802.11b/g/n networks.

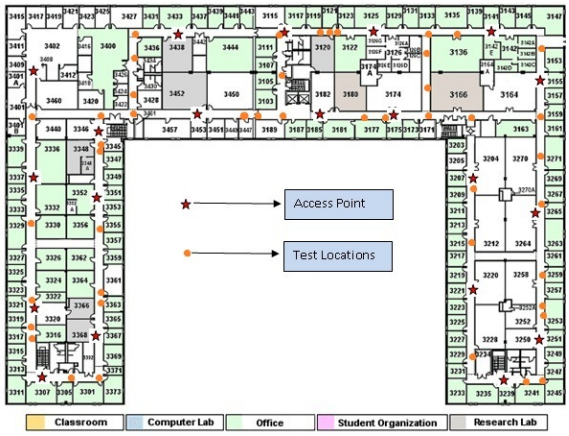
For data collection and experimental evaluation, we defined a two dimensional coordinate system for each building, in our test bed, which spanned the entire building. One corner of the building was marked as the origin (0,0). The horizontal X axis was aligned with one wall of the building while the vertical Y axis was aligned perpendicular to it. The axis points were placed 1 feet apart on each axis. The locations of all the APs and



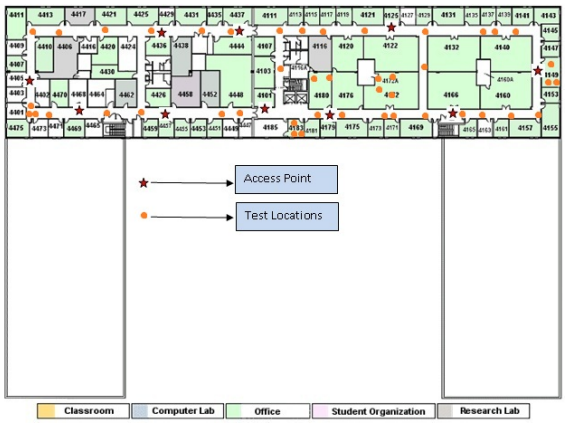
(a) Floor Map of the first floor



(b) Floor Map of the second floor

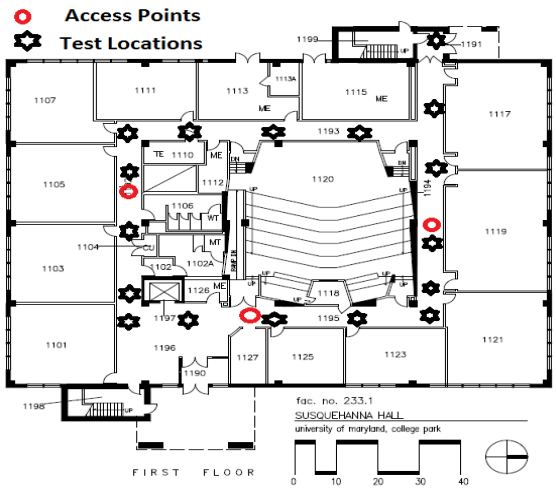


(c) Floor Map of the third floor

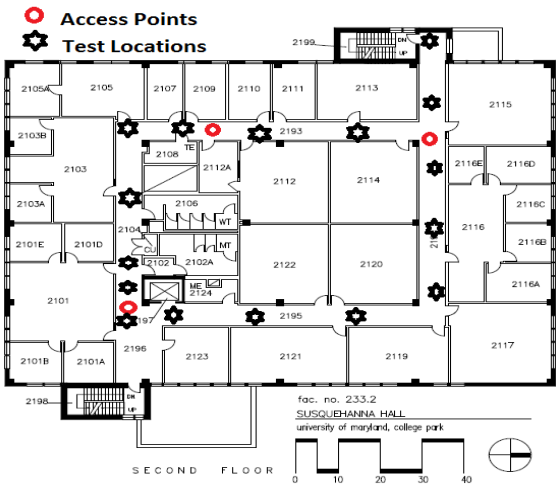


(d) Floor Map of the fourth floor

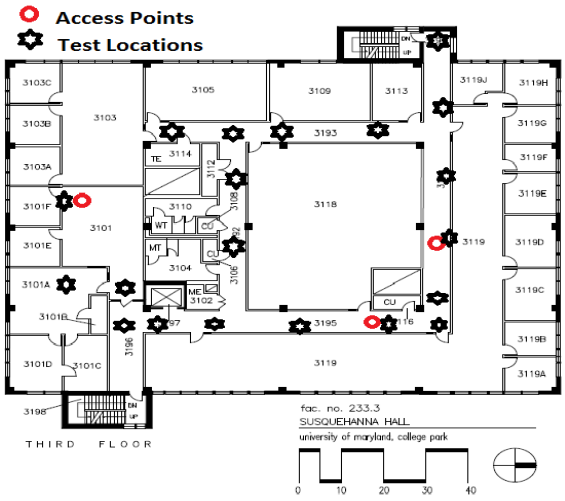
Figure 4.2: Floor Maps of Building # 1 (AV Williams Building) with locations of the APs and some of the test points marked



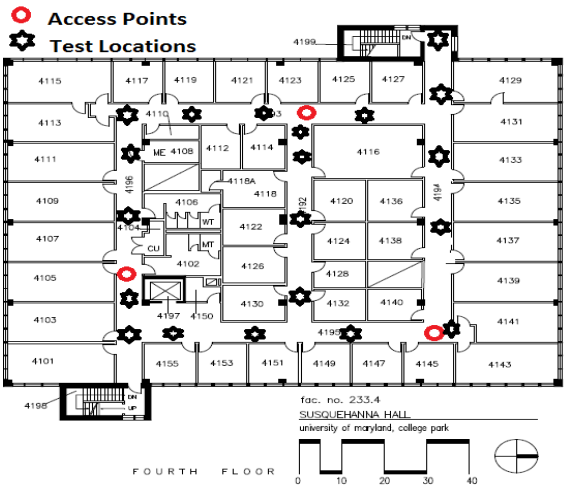
(a) Floor Map of the first floor



(b) Floor Map of the second floor



(c) Floor Map of the third floor



(d) Floor Map of the fourth floor

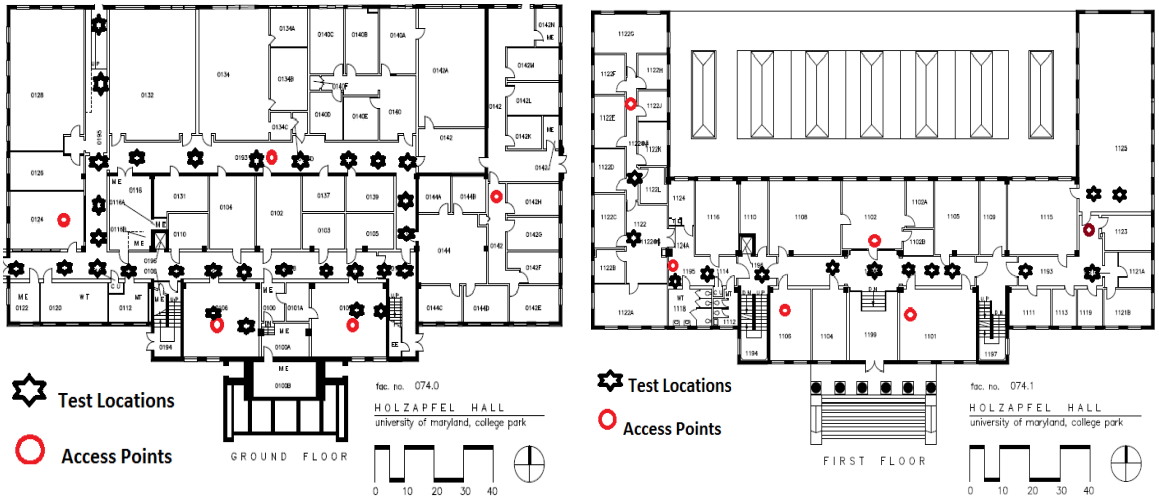
Figure 4.3: Floor Maps of Building # 2 (Susquehanna Hall) with locations of the APs and some of the test points marked

test points (where the RSSI samples were recorded) in that building were then defined in this X-Y coordinate space. The APs are placed at fixed locations as determined by the campus authorities. We placed the test points 5 feet apart in the corridors and a few accessible rooms on each floor of each building. This distribution generated 120 test points in Building # 1, 75 test points in Building # 2 and 50 test points in Building # 3 (Table 4.1). Figures 4.2, 4.3 and 4.4 show the floor maps of each of the testbed buildings with the location of the APs and some of the test points marked. To improve the legibility of the floor maps so that individual locations can be discerned, we have marked the locations of some of the test points only.

We collected RSSI samples at these labeled test points using an Android based mobile client application called LocateMe (explained later in Section 4.3.1). We noted the ground truth (x,y, floor) coordinates of the test point where the samples was collected. We then compared these ground truth values with the actual (x,y,floor) values estimated by Locus for each sample.

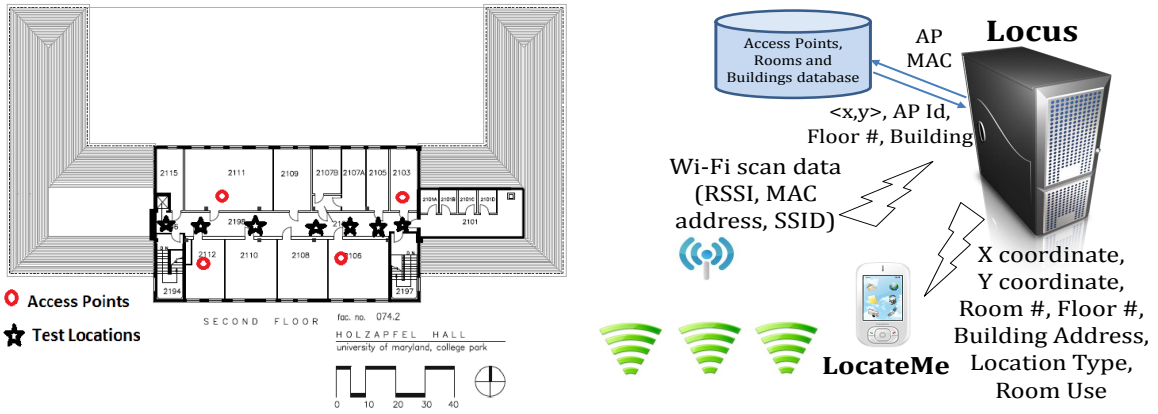
4.3 The Locus System

The Locus system has a client-server architecture. The lightweight client application runs on a mobile device and scans the Wi-Fi environment. Each scan at a test point produces a RSSI sample which contains the network name (SSID), MAC address, signal strength and frequency for each AP heard at the test point. This sample is then sent to the Locus server in a HTTP query. The server side system parses the AP list with signal strength values and employs a Floor and Location Determination algorithm in order to determine



(a) Floor Map of the Ground Floor

(b) Floor Map of the First Floor



(c) Floor Map of the Second Floor

(d) System overview of Locus

Figure 4.4: Floor Maps of Building # 3 (Holzapfel Hall) with locations of the APs and some of the test points marked, and System overview of Locus

an indoor location (x coordinate, y coordinate, Room #, Floor, Building, Address) for the test point which is returned as a JSON response to the client's query.

Figure 4.4(d) shows the overview of Locus. It consists of two subsystems:

4.3.1 Client side application

The client application for Locus is an Android mobile application called 'LocateMe' that can run on a smartphone or a tablet. The application scans the environment for Wi-Fi APs using the standard Android scan functionality [4]. This RSSI sample is then sent by the client to the Locus Server side system. All the samples are pruned to remove detected signals weaker than a threshold value of -85 dBm. This is because AP signal strengths below this threshold are weak and received only occasionally, while stronger signal strength values are received consistently.

A sample XML generated by the client application looks like:

<?xmlversion=Ï.0ÿ?>

<data> <accesspoints>

<accesspoint><name>umd-secure</name><mac>f4:xx:yy:zz:4a:a2</mac>

<signal>-69</signal><freq>2462</freq></accesspoint>

.....

</accesspoints></data>

The LocateMe client application also receives the JSON response returned by the Locus server side system and displays it to the user. This response contains the x coordinate, y coordinate, floor, room and building address information for the location at which the RSSI sample was taken.

[4] http://developer.android.com/reference/android/net/wifi/WifiManager.html

4.3.2 Locus Server side system

The Locus Server side system consists of four components:

4.3.2.1 Access Points, Rooms and Buildings Database

The server side system maintains a database of 4500 APs deployed in the UMD campus[5]. The database includes their MAC addresses, AP IDs (which correspond to the room number of the nearest room), and the floors, wings and buildings where they are installed. It also records the 'Room Use' for each room which corresponds to its category such as 'Faculty Office', 'Staff Office', 'GA room' etc.

In addition, the database also contains the (x,y) coordinates for each AP (in the coordinate system of that building) in each of the three buildings in our test bed. Along with this information, the database also stores the bounding box coordinates (north east and south west corners) for all the rooms in each of our testbed buildings.

4.3.2.2 Locus Floor and Location Determination Algorithm

The Locus server side system parses the XML that it receives from the Android client application and resolves the <x,y> coordinates, AP ID, Floor, and Building Address of every AP, present in the XML, based on its MAC address. In the majority of the samples, the APs being heard in a sample were present within the same building. The system then employs the Locus Floor and Location Determination algorithm (Algorithm 1) to determine the indoor location of the client device.

[5] This data was not collected manually but was obtained from appropriate sources at the university. We updated the database periodically by obtaining an updated list of the APs and their locations.

Algorithm 1: Locus Floor and Location Determination Algorithm

Data: RSSI sample from the client containing the SSID, MAC address, and RSSI value for each AP being heard

Result: Client's estimated location - X,Y coordinate, Room #, Floor #, Building, Address, Location Type, Room Use

Remove all APs with signal strength $<$ threshold;

while *not at end of Data* **do**

    map MAC address of each AP to Building, floor, x,y;

end

foreach *Building floor* **do**

    compute statistical measures : numSS, avgSS, maxSS, varSS

end

Determine floor properties - maxNumFloors, maxSSFloor, maxAvgFloor, maxVarFloor;

```
// Check the combinations and individual properties in
   decreasing order of their accuracies as mentioned in
   Table 4.2
```

foreach *Combination or property that is satisfied* **do**

    if *labelCombination equals null* **then**

        Select the current combination to determine label floor;

        labelCombination ⟵ Higher order combination enclosed in the current selection;

        labelFloor ⟵ floor property satisfying the labelCombination;

end

Compute weights for every AP on labelFloor based on its power;

Location <x,y> ⟵ weighted average of locations of n APs heard from labelFloor;

Map Location to a Room #;

Determine Building Location Type and Room Use;

return *Location, labelFloor, Building, Address, Location Type and Room Use*

The algorithm determines the location in two phases:

- Floor Determination Phase - To determine the floor, the algorithm computes the following four statistical measures, for each building floor, from every signal strength sample that it receives from the client:

    1. *numSS* - # of signals from the floor,
    2. *maxSS* - Highest signal strength received from the floor,
    3. *avgSS* - Average of signal strengths received from the floor,
    4. *varSS* - Variance of signal strengths received from the floor.

    These statistical measures were selected based on the fact that AP signals are attenuated when passing through ceilings and floors. As a result, a client is more likely to hear signals from its current floor than other floors, and those signals are likely to be stronger. AP signal strengths from a different floor will be attenuated and hence their average strength will be lower. The variance of signal strengths of APs from the current floor is also expected to be higher than other floors. This is because all floors will have some APs with low signal strengths, but the current floor will have APs with high as well as low signal strengths.

    Comparing these statistical measures across floors, a floor *property* value is applied to each sample. The property value is a floor number, or in the case of a tie, floor numbers. The properties are:

    1. *maxNumFloors*: Floor(s) with maximum count of signals.

Combination/Property	Recall	Precision	Accuracy
maxNumFloors = maxSSFloor= maxAvgFloor = maxVarFloor	1.0	0.83	0.91
maxNumFloors = maxSSFloor= maxAvgFloor	0.97	0.79	0.87
maxNumFloors= maxAvgFloor = maxVarFloor	1.0	0.72	0.84
maxNumFloors = maxSSFloor= maxVarFloor	0.97	0.73	0.83
maxSSFloor= maxAvgFloor = maxVarFloor	1.0	0.67	0.80
maxNumFloors = maxSSFloor, maxVarFloor = maxAvgFloor	1.0	0.71	0.83
maxNumFloors = maxAvgFloor, maxVarFloor = maxSSFloor	0.99	0.69	0.82
maxNumFloors = maxVarFloor, maxSSFloor = maxAvgFloor	0.99	0.68	0.81
maxNumFloors = maxSSFloor	0.97	0.85	0.92
maxAvgFloor = maxVarFloor	0.92	0.89	0.91
maxSSFloor = maxAvgFloor	0.95	0.81	0.87
maxNumFloors = maxAvgFloor	1.0	0.76	0.86
maxNumFloors = maxVarFloor	0.93	0.81	0.86
maxSSFloor= maxVarFloor	0.95	0.74	0.83
maxNumFloors	-	-	0.91
maxSSFloor	-	-	0.87
maxAvgFloor	-	-	0.86
maxVarFloor	-	-	0.82

Table 4.2: Accuracy for individual properties and combination of properties

2. *maxSSFloor*: Floor with AP with maximum signal strength

3. *maxAvgFloor*: Floor with maximum average signal strength

4. *maxVarFloor*: Floor with maximum signal strength variance

There are exceptions to these generalizations, particularly for the floors below and above the actual floor, but the combined use of these properties can yield the correct floor with very high probability ($> 95\%$), as we observed empirically.

A validation dataset, containing 500 RSSI samples from 120 test points in Building # 1, was collected via the methodology described for data collection in Section 8.7. This dataset was used to evaluate the properties and their combinations to determine their accuracies shown in Table 4.2. The combinations included taking two, three

and all four properties together. If a plurality of the properties agree on a floor, that is the label floor. If there is a tie, the property or combination of properties with the higher accuracy is used as the label floor. This validation is performed only once for tuning the algorithm. The algorithm is then used universally.

The Accuracy (a) measure for a combination is its F-Score which is the harmonic mean of precision and recall. Precision (p) and Recall (r) are defined here as:

$$\text{p} = \frac{\begin{array}{c}\text{\# of instances valid for a combination where} \\ \text{label floor matched with ground truth floor}\end{array}}{\begin{array}{c}\text{\# of instances in the dataset where} \\ \text{label floor matched with ground truth floor}\end{array}}$$

$$\text{r} = \frac{\begin{array}{c}\text{\# of instances valid for a combination where} \\ \text{label floor matched with ground truth floor}\end{array}}{\begin{array}{c}\text{\# of instances that were valid for} \\ \text{a combination}\end{array}}$$

$$\text{a} = \text{F-Score f} = \frac{2 * p * r}{p + r}$$

The accuracy measure for each individual property is:

$$\text{a} = \frac{\begin{array}{c}\text{\# of instances where ground truth floor} \\ \text{was equal to the individual property}\end{array}}{\begin{array}{c}\text{Total \# of instances} \\ \text{in the dataset}\end{array}}$$

Based on the determined accuracy measures, we established an order for checking these properties and combinations in Algorithm 1 to determine the label combination and the label floor. Since the first combination is the highest order combination (that involves all four properties being equal) and encompasses all other combinations, it is tested first. Similarly, the combinations of three properties being equal

are tested next as they include the combinations of two of the properties being equal within them, and so on.

- Location Determination Phase - Once the algorithm determines the label floor of the client, it uses an indoor radio propagation model in order to determine the client's approximate location on that floor. It first normalizes the square root of the power of the signal received from each AP to generate a weight. It then computes a weighted average of the location of the *n* strongest APs on the label floor, where n is varied from 1 to the maximum number of APs heard from the label floor.

  The signal strength for each AP is essentially the average of the signal strength of all the virtual APs running from it. The weights are calculated by converting this averaged signal strength to power (in mW), taking the square root and normalizing it. This nullifies the effect of location of APs that are far away and have weaker signal strengths. This is because they will have a much lower weight as compared to APs that are closer and have stronger signal strengths. Thus, for AP_i :

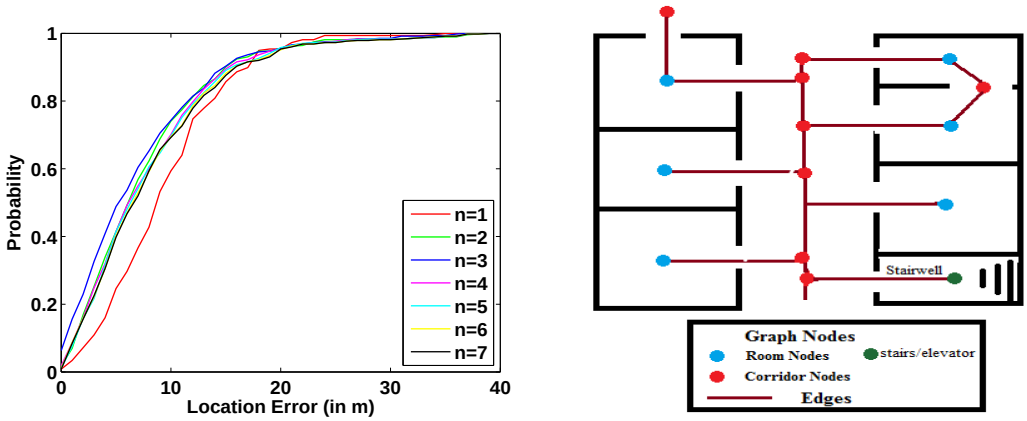
$$\text{Power } P_i(\text{in mW}) = 10^{\frac{\text{signal strength of AP}_i \text{ in dBm}}{10}}$$

$$\text{Weight } w_i = \frac{\sqrt{P}_i}{\sum_{i=1}^{n} \sqrt{P}_i}$$

  The estimated X and Y coordinates for the client are calculated as a weighted average of the (x,y) coordinates of the n strongest APs.

$$\text{X coordinate } x_{estimated} = \sum_{i=1}^{n} x_i \times w_i$$

$$\text{Y coordinate } y_{estimated} = \sum_{i=1}^{n} y_i \times w_i$$



(a) CDF of Location Error for the validation data set for different values of n

(b) Graph G describing the space on a building floor

Figure 4.5: CDF of Location Error for the training data set and Graph G describing the space on a building floor

These estimated coordinates are then mapped to a Room # by comparing them with the bounding box coordinates of each room on the label floor and determining the bounding box within which they lie. If no corresponding bounding box is found, then it is assumed that the test point is in the corridor and the nearest AP ID is reported as its Room #.

Figure 4.5(a) shows the CDF for n $\in [1, 7]$ for the validation dataset collected from Building # 1. n= 3 achieves the least average location error of 7.42 m and hence, we use n=3 in our current implementation[6]. Henceforth, all experimental results will concentrate on n=3.

[6] Plese note that n =1 corresponds to the location of the strongest AP being picked as the client's location.

4.3.2.3 Location Type and Room Use Determination

We also augment the location of the client with the Location Type of the building and Room Use for the room. The Location Type corresponds to one of the FourSquare categories and is obtained by searching for the building using the FourSquare Venues API [7]. The Room Use corresponds to its category such as 'Staff Office', 'Faculty Office' etc.

Finally, the estimated location of the client consisting of the x coordinate, y coordinate, room, floor, building address, location type and room use information is returned as a JSON object to the client application. A sample JSON response returned by Locus is:

{"x coordinate":35.80494601039969,"y coordinate":1.6341576683999484, "Room":"4149", "Floor":"4","Wing":"1c","Address":"A.V. Williams Building, University of Maryland, College Park, MD","Location Type":"College Academic Building","Room Use":"Faculty"}

4.3.3 Shortest Path Determination

Locus can also generate the shortest path between two indoor locations on the same or different floors. This enables indoor navigation applications (explained later in Section 4.5.1) that can plot a path between two rooms and display it to the user on a floor plan.

It employs Algorithm 2 that takes as input a graph *G* (which represents the structure of a building and contains rooms, stairwell/elevators, and corridor points as nodes) and finds the shortest path between two nodes of *G*. An edge between two nodes denotes a direct path between them. Each node essentially represents a geometric point (x,y) stored in the spatial database. The weight of the edge between two nodes is computed based on

[7] https://developer.foursquare.com/overview/venues.html

Algorithm 2: Algorithm for Shortest Path Determination

Data: Graph $G < N = \{rooms, stairs/elevators, corridors\}, E >$, startNode, endNode

Result: Shortest Path between two nodes

floor1 ← extract Floor from startNode;

floor2 ← extract Floor from endNode;

Sub Graph S_1 ← get sub graph for floor1;

if *floor1* ≠ *floor2* **then**

    Sub Graph S_2 ← get sub graph for floor2;

    Graph S ← Merge S_1 and S_2;

Path ← run Dijkstra's algorithm on Graph S with startNode as source;

return *Path*;

the Euclidean distance between them. A sample graph is shown in Figure 4.5(b).

We do not use the entire graph for shortest path computations, but extract only the floor sub-graphs that we will need. To this end, Algorithm 2 extracts the floor numbers of startNode and endNode. If the two floors are different, it generates two sub graphs S_1 and S_2 from G that contain nodes on the two floors (floor1 and floor2). It then combines S_1 and S_2 to form graph *S* by connecting the inter-floor edges at stairwell/elevator nodes. Finally, it performs a shortest path search on *s* using Dijkstra's algorithm. Dijkstra's algorithm is a single source shortest path algorithm that explores all possible nodes starting from the source and keeps updating the shortest distance to reach a node. It also maintains connectivity information based on the minimum distance, which helps to reconstruct the shortest path starting from the source to any node on the graph. Algorithm 2 returns a path as a set of nodes on the graph.

4.4 Evaluation

We now evaluate the Locus system, and its underlying algorithm, using several evaluation metrics which are established measures used in indoor localization ([20]).

4.4.1 Test Dataset

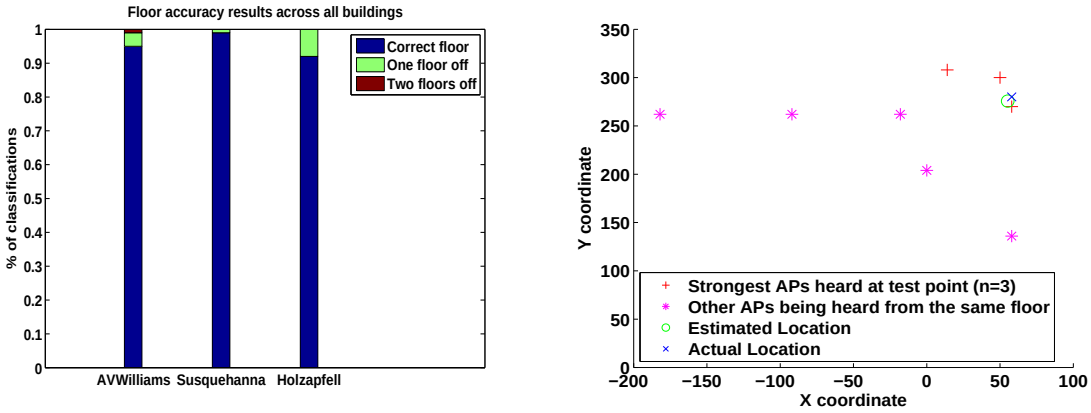
To evaluate Locus, we collected several RSSI samples at different test points in the three buildings in our test bed. These samples were collected by the authors using the LocateMe client application via the methodology described for data collection in Section 8.7. Our final test dataset contains 281 RSSI samples from 90 test points in Building # 1, 217 samples from 74 test points in Building # 2 and 164 samples from 50 test points in Building # 3.

4.4.2 Results

We use seven metrics to evaluate the performance of the Locus system: Floor Accuracy, Location Error, Complexity, Scalability, Universal Applicability, Robustness and Cost.

4.4.2.1 Floor Accuracy

Since our test sites are multi-story buildings, we have considered floor accuracy to be a measure of the percentage of correct floor estimations by Locus. We believe that it is an important performance measure especially for practical multi-story environments such as offices, hotels, or malls that have multiple APs on each floor. Figure 4.6(a) summarizes



(a) Floor Accuracy results across all buildings

(b) An illustrative example showing Location of APs and estimated and actual locations of client in Building # 1 (using n=3)

Figure 4.6: Floor Accuracy and Location Estimation

#	Building Name	Average Error(m)	Median Error(m)
1	AV Williams	6.7	5.31
2	Susquehanna Hall	7.37	7.28
3	Holzapfel Hall	5.41	4.63

Table 4.3: Average and median location errors (in m) for all the buildings for n=3

the accuracy with which the algorithm can correctly determine the current floor, be 1 floor off (i.e. predict the adjacent floor to the actual floor as the correct floor) and be 2 floors off, across all the three buildings. The average floor accuracy is 95.41% for Building #1, 99.54% for Building #2 and 92% for Building #3. The average floor accuracy across the 3 buildings is 95.33%.

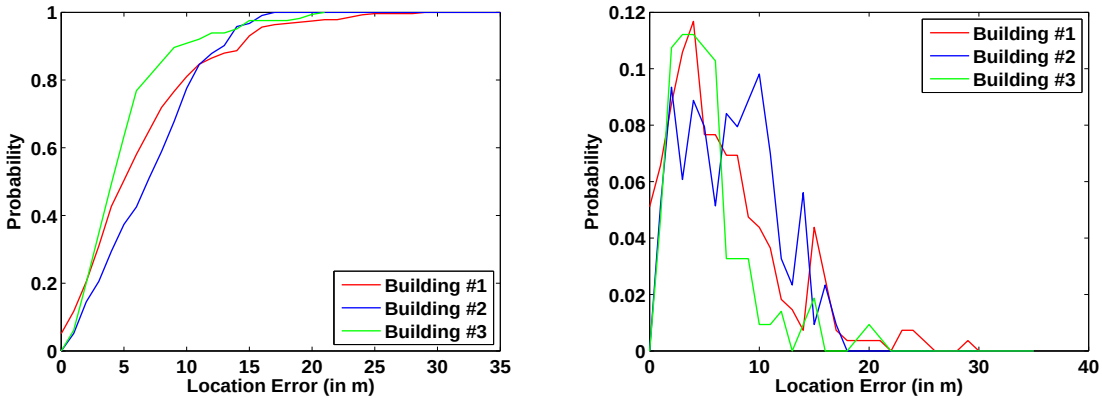
4.4.2.2 Location Error

As mentioned in Section 4.3.2.2, once the floor is determined, we determine the client's location on that floor by calculating a weighted average of the locations of the 3 strongest APs being heard on that floor. Figure 4.6(b) shows an illustrative example with the location of APs and actual and estimated locations of the client in Building # 1 marked.

Table 4.3 show the average and median location errors for all buildings for n=3. The average location error across all the three buildings is 6.49m which corresponds to approximately room level accuracy. Figure 4.7 shows the PDF and CDF of the location errors for n=3 for all the three buildings. In Building #1, 25 % of the errors lie within 2.95 m, 50 % within 5.31 m and 75% within 9.21 m. In Building # 2, 25 % of the errors lie within 3.94 m, 50 % within 7.28 m and 75% within 10.12 m. In Building # 3, 25 % of the errors lie within 2.87 m, 50 % within 4.63 m and 75% within 6.28 m.

4.4.2.3 Complexity

Complexity can be measured in terms of software or hardware. Since our approach requires no proprietary hardware and is based solely on existing infrastructure, the hardware complexity is minimal. Also, the Locus server side system runs on a central server that has ample processing capability and power supply. The client side application is very lightweight and runs on off the shelf mobile devices. In addition, it is restricted only to scanning and detecting the APs being heard, sending this information to the server side system and displaying the response received.



(a) Cumulative Distribution Function of the Location Error

(b) Probability Density Function of Location Error

Figure 4.7: Location Error PDF and CDF for n=3 for all three buildings

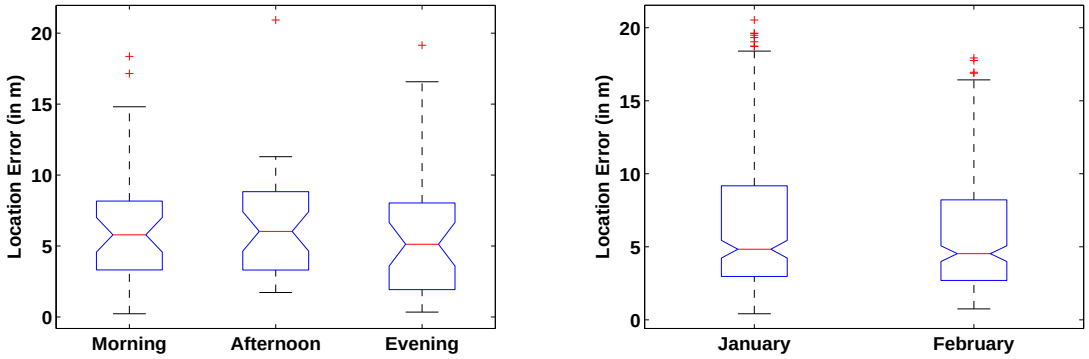
4.4.2.4 Universal Applicability

Locus employs the underlying Floor and Location Determination algorithm for indoor localization across all the buildings. No calibration is required for any building. Thus, it is calibration-free and can be universally used.

4.4.2.5 Scalability

Scalability of a localization system can be assessed in terms of:

- Geographic scalability - This means that the system will work even when area or volume covered is increased, and
- Density scalability, which means that as the number of units located per unit geographic area/space per time period is increased, wireless signal channels may become congested. Hence more calculations or communication infrastructure may be



(a) Readings sampled at 3 different times a day

(b) Readings sampled on different days separated by a month

Figure 4.8: ANOVA Box Plots of Location Error for Building # 1 with readings sampled on different times in a day and on different days separated by a month

required to perform localization.

- Another measure of scalability is the dimensional space of the system.

Locus can be used in multi-story and 3D spaces as shown by the experiments which have been conducted across three different buildings in our campus. Since the density of APs is part of the infrastructure, we have tested Locus on various floors of different buildings where the AP density varies greatly (see Table 4.1). Also, the localization process in Locus is independent of the number of floors in a building and hence, it can be scaled to any multi-story building. In addition, the computational overhead of the SQL queries is minimal. As a result, they can be easily scaled to a larger database.

4.4.2.6 Robustness

Since Locus avoids any dependency on radio maps, it is robust to changes in the environment such as the time of the day, number of people in the closed space etc. Even if the positions of APs are changed, only the AP database will have to be updated. The deployed system and its underlying algorithm will remain unchanged unlike fingerprinting, where the radio map has to be calculated afresh.

To demonstrate the robustness of Locus to environmental changes such as rearrangement of furniture or change in number of people, we carried out the following experiments:

- Change in number of people: We collected 219 RSSI samples at three different times in a day (in the morning at around 8 am, in the afternoon at around 1 pm and in the evening at around 6 pm) at 120 test points in Building # 1. We selected these times because they exhibit the greatest variation in the number of people present. The university work hours are from 8:30 am - 4:30 pm. Hence, there are very few people in the department at 8 am in the morning. Most people are in their offices by 1 pm. Moreover, many people leave for home by 5:30 pm and only some of the students are in the building at around 6 pm.

  We performed one-way ANOVA on the samples with an α (significance) level of 0.05. Figure 4.8(a) shows the ANOVA box plot for the Location Error for readings sampled at three different times in a day. The p-value was 0.71. As can be seen, the mean values for the two box plots are aligned very closely, thus, indicating that there was very little variation in the errors at the different times of the day. This

demonstrates that Locus is agnostic to change in the number of people around.

- Environmental changes: Similarly, we carried out experiments on two days which were a month apart (in January and February) to determine any effects that can arise from environmental changes such as movement of furniture or open and closed doors. We selected these days as they fall during the winter holidays and the subsequent spring semester and exhibit great variation in the environment. Most people go home during the winter holidays and their offices are empty and locked.

  We collected 281 samples in Building # 1 on these two selected days. We performed one-way ANOVA on the samples with an α level of 0.05. Figure 4.8(b) shows the ANOVA box plot for the Location Error for readings sampled on the same day in January and February 2014. The p-value was 0.82. As can be seen, the mean values for the two box plots are aligned very closely, thus, indicating that there was very little variation in the errors at the days spread across a month.

- Displacement of APs: We also analyzed effects of changes such as displacement of APs by conducting experiments on two different days before and after the displacement. To this end, we first collected 265 RSSI samples in Building # 1. We then manually interchanged the position of some of the APs in Building #1 and updated their locations in our AP database. Finally, we collected the second set of RSSI samples at the same test points in Building # 1.

  Figure 4.10(a) shows the one -way ANOVA box plot for the Location Error for the readings belonging to the two sets of samples. It was performed with an α level of 0.05. As can be seen, the mean values for the two box plots are aligned

very closely, thus, indicating that there was very little variation in the errors before and after change in positions of APs. Moreover, as opposed to fingerprinting, no recalibration of the system was required, thus, saving a significant amount of effort.

4.4.2.7 Deployability and Cost

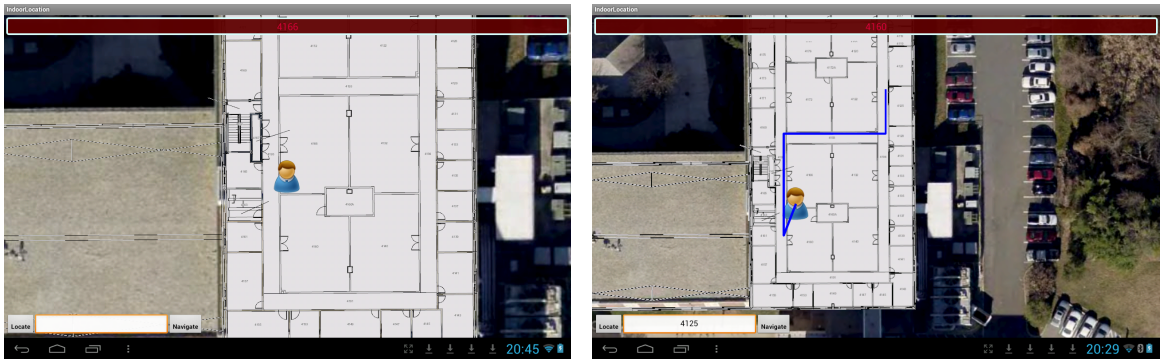
One of the biggest advantages of Locus is that it is readily deployable and has zero cost for deployment and maintenance as it relies solely on the existing infrastructure. The time cost of setting up is also minimal as it only requires setting up access to a database with the APs and buildings information.

4.5 Location-aware Applications

In this section, we describe two indoor localization applications, enabled by Locus, that can be used to localize, track and navigate in indoor spaces. We also briefly discuss other indoor location based services and applications, enabled by Locus, that we are currently developing.

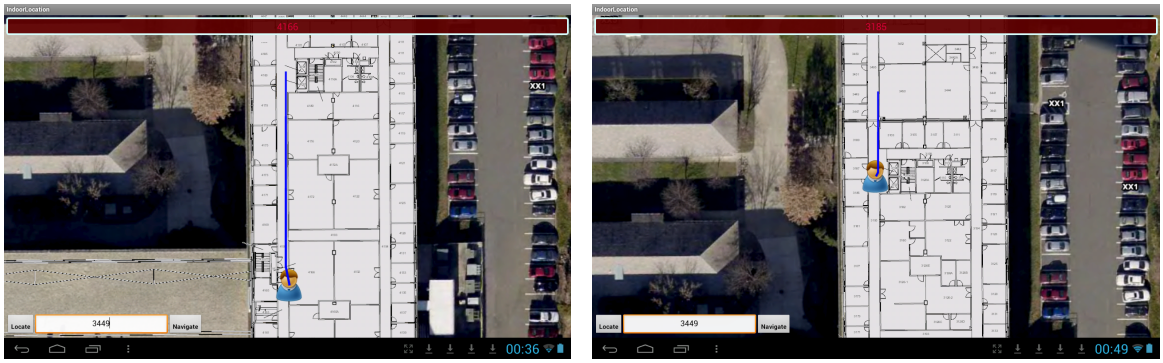
4.5.1 Navigation

We have developed a navigation application called *IndoorNavigation* on top of the vanilla LocateMe client application (explained in Section 4.3.1). This application displays the user's current location on the appropriate floor map of the building he/she is in, tiled over an ArcGIS ESRI map. The user can then enter a destination room # and the application displays a path to it from the user's current location. As explained in Section 4.3.3, the



(a) Localizing the user and displaying his indoor location on a floor plan

(b) Displaying the shortest path to a destination location on same floor



(c) Displaying the shortest path to an elevator to move to a different floor

(d) Displaying the path to the destination from the elevator after switching floors

Figure 4.9: Screen shots of the IndoorNavigation application displaying shortest paths to destination rooms on the same and different floors of a building

Locus server side system employs Algorithm 2 to calculate the shortest path and returns a list of points on the path as a JSON Object. The client application then draws a path connecting these points and displays them on the floor map.

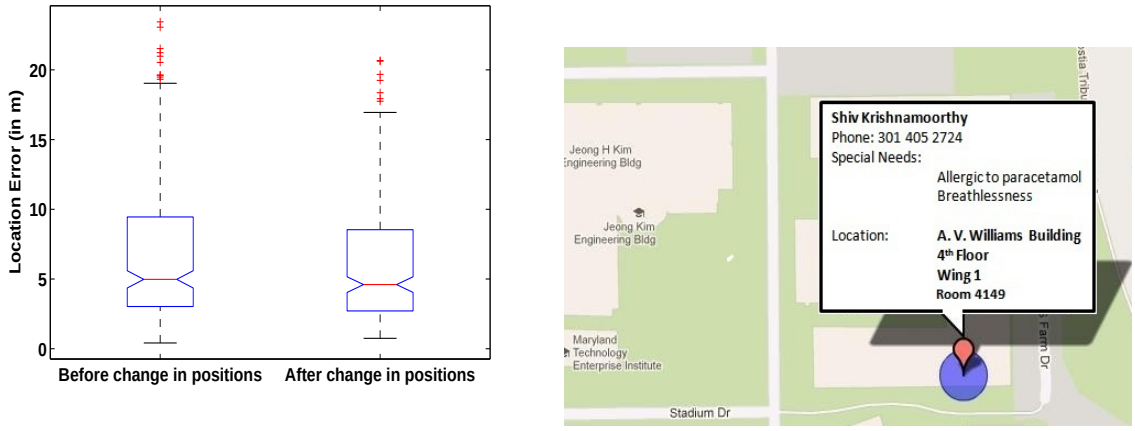
As shown in Figure 4.9(a), the user presses the 'Locate' button and the application localizes him to an indoor location - Room 4160 of Building # 1 (AV Williams Building) and displays it on the floor plan of Floor 4. The user then enters a destination room number - Room 4125 and presses the 'Navigate' button. The application then plots the shortest path to the destination room from the user's current location and displays it on the floor plan (Figure 4.9(b)). In case the destination room is on another floor (say, Room 3449 on Floor 3), the application first computes the shortest path to an elevator (Figure 4.9(c)) and once the user's floor changes, it displays the shortest path from the elevator to the destination room as shown in Figure 4.9(d).

We have tested the *IndoorNavigation* application on different floors of Building # 1 successfully and intend to test it in other buildings on campus.

4.5.2 Tracking in emergency scenarios

M-Urgency [8] [18] is a public safety system that redefines how emergency calls are made to a Public Safety Answering Point (PSAP), such as in the 911 system, and is designed to be context-aware of the situation in which it is used. It enables mobile users to stream live audio and video from their devices to a local PSAP along with the real time location. In such scenarios, a precise information about the caller's location will be extremely helpful for the responders to get to the location of emergency, thus, avoiding confusions and

[8] http://m-urgency.umd.edu/



(a) Readings sampled before and after changing AP positions

(b) Caller's location being displayed on the M-Urgency dispatcher console

Figure 4.10: ANOVA Box Plots of Location Error for Building # 1 with readings sampled before and after changing AP positions, and Screenshot of the M-Urgency Dispatcher console

delay.

During a normal 911 call, the emergency personnel are able to locate the building where the call originated from, but often find it difficult to zero in on the actual floor and the location of the caller on that floor. A system like Locus is essential here. As an M-Urgency call is made to the police department, the caller application makes a *location request* to the Locus system. The indoor location returned by Locus is displayed to the dispatcher by the caller application, as shown in Figure 4.10(b). This feature is already incorporated and ready to be released in next public version of the deployed M-Urgency system.

4.5.3 Other services and applications

Locus can enable several indoor location based services and applications, some of which we are already developing, such as:

- Educational Content and Events - A university student with a smart device can benefit from an application that downloads class notes or other multimedia content for a class based on the current time, his/her current location and class schedule. A visitor could use such an application to get information about all the relevant events or talks happening in a building, along with directions to each venue.

- Retail - Retail organizations would benefit from an application that performs targeted advertising, based on the time, location of customers in a mall and their proximity to public displays. They could also send coupons and offers to customers based on their shopping preferences.

  Moreover, shoppers will welcome a location-based application to assist them in finding a product that they are looking for in a store. A context-aware shopping application can enable shoppers to tag stores in malls with deals, and also help them in searching for deals based on their shopping preferences.

- Healthcare - For healthcare providers, an application that can track patients in a hospital or at their home and monitor their health will be very useful.

- Social Networking - A social networking application can help connect people, with similar interests, at an indoor conference venue or a gathering.

- Entertainment - People can use indoor location to build augmented reality games such as Hide and Seek.

- Location and Time-aware reminders - Users can leave reminders or notes for themselves or others. These will be delivered to the intended recipients based on their location in a building or a room, the current time and their proximity to another user. Potentially, an intelligent personal digital assistant (PDA) can access the calendar of a user and remind him/her about a meeting that will happen in the same building or another. It can also determine how much time it will take to walk there based on his/her current location and generate an alarm at time with appropriate lead.

Chapter 5: SenseMe: A System for Continuous, On-Device, and Multi-dimensional Context and Activity Recognition

In order to make context-aware systems more effective and provide timely, personalized and relevant information to a user, the context or situation of the user must be clearly defined along several dimensions. To this end, the system needs to simultaneously recognize multiple dimensions of the user's situation such as location, physical activity etc. in an automated and unobtrusive manner. In this chapter, we describe SenseMe [9] - a system that leverages a user's smartphone and its multiple sensors in order to perform continuous, on-device, and multi-dimensional context and activity recognition.

5.1 Introduction

A user's situation can have multiple dimensions or aspects at any given instant of time, some of the most important ones are: where he is (location), what he is doing (activity), and when (time) i.e. "Who, What, Where and When?" [38]. These aspects form the W4 model proposed by Castelli et al. [39] to represent contextual information about physical world objects, which can be employed by both users and context-aware systems. An additional aspect - whom the user is with, proposed by Schilit et al. [40], can be added to augment this model. Taken together, these five dimensions paint a context-rich picture of

the user.

To determine these aspects of a user's situation, the context-aware system needs to simultaneously recognize multi-dimensional contextual information of a user at a given instant of time. Moreover, for large-scale adoption by users, this information should be acquired in a non-invasive manner, without placing undue burden on them. As a result, automated sensing ability is highly desirable for any information system powered by context-aware computing. In addition, this ability to sense users should be embedded in devices that they can carry around without effort and the user interaction should minimal so that it is unobtrusive.

The ubiquitous smartphone, with its multitude of sensors and capabilities, has become the best choice for this purpose. Today's smart phones come equipped with an increasing range of sensing, computational, storage and communication capabilities. This has enabled sensing and tracking applications to emerge across a wide variety of applications areas such as location based services, personal healthcare, and social networking etc. A key challenge of mobile phone sensing is to process raw data from multiple sensors in order to infer higher level activities and context in real-time and in a robust, generic and energy efficient manner.

We present SenseMe - a system that leverages the smartphone and its multiple sensors in order to perform **continuous, on-device, and multi-dimensional context and activity recognition** for a user. It achieves this in a robust, automated, scalable, power efficient and non-invasive manner. Table 5.1 summarizes the set of five dimensions of a user's situation (and their possible discrete values) that SenseMe recognizes, with a temporal

resolution of a minute[1], to determine the aspects "Who, What, Where, When, and Who you are with?". The dimensions are:

1. *Environmental context* - This dimension represents whether the user is Outdoors (outside a building), Indoors (inside a building), or Indoor-Outdoors (close to or in a semi-open building or inside a building but near a door or large window). It is significant as it enables context-aware localization(explained next).

2. *Context-aware Location* - This dimension represents location of the user in both indoor and outdoor environments. If he is 'outdoors', SenseMe uses GPS for localization and reverse geocoding in order to resolve the logical address of the user's location. If he is 'indoors' or in the 'indoor-outdoor' area, SenseMe performs indoor localization using Locus [7] which is a Wi-Fi based indoor localization system for multi-story buildings.

   Since the location should be in a human understandable format, SenseMe determines a high-level logical location in addition to a low-level location in a raw format. It further identifies a location type (restaurant, academic building, student center etc.) for each resolved location. This enables tracking of a user's location history at fine and coarse grained levels throughout the day and also helps in identifying the places where they spend a significant amount of their time.

3. *Physical Activity* - This dimension represents the physical activity of the user such as walking, running, stationary or in a vehicle (car, bus, bike etc.).

4. *Device Activity* - This dimension represents the task the user is currently engaged

[1] This time slice duration is long enough to be discriminative and short enough to provide high accuracy labeling results.

Situation Dimension	Possible Values
Environmental Context	{Indoor, Outdoor, Indoor-Outdoor}
Physical Activity	{Stationary, Walking, Running, In-vehicle}
Context-aware Location	Locations determined by Wi-Fi(indoors) or reverse geocoding(outdoors)
Device Activity	Task the user is engaged in on the device such as phone call or messaging
Social Context	Number of people around the user at any given instant of time

Table 5.1: Situation dimensions being captured by SenseMe at any instant of time

in on his/her smart device (checking mail or phone call). It is equally important as the physical activity due to the growing proliferation of mobile devices and their increasing usage as opposed to desktop and laptop computers. Most people carry their smart devices everywhere and perform a substantial set of their everyday activities such as web browsing on it. As a result, we believe that it forms a significant dimension of the user's situation.

5. *Social context* - This dimension represents the social activity of users i.e. how much time they spent interacting or being around people.

Thus, at any instant of time t, SenseMe represents the user's situation S as a feature vector in a multi-dimensional *Situation Space*. For example, S(t) for a user is: <Indoor; Stationary; Phone Call; A.V. Williams Building - College Academic Building; With 4 people>.

SenseMe has been completely implemented on the Android platform and runs on off-the-shelf Android smartphones and tablets. We evaluate SenseMe extensively against

Device	Range		Resolution	
	Magnetic Field (μT)	Light (lux)	Magnetic Field (μT)	Light (lux)
Google Nexus	9830.0	65528.0	0.15	0.2
Motorola Xoom	2000.0	208076.8	0.0625	0.05

Table 5.2: Comparison of Range and Resolution for Magnetic Field and Light sensors on two different devices

several qualitative and quantitative metrics, with the aid of 2 two-week long live deployments involving 15 participants. We demonstrate improved or comparable accuracy with respect to existing systems without requiring any user calibration or input.

The rest of the chapter is organized as follows: In Section 8.2, we discuss existing related work in the field of Context and Activity Recognition and highlight their shortcomings and differences with our approach. Following that, we explain the key contributions of our work in Section 5.3 and training data collection in Section 5.4. Section 5.5 describes the SenseMe system and Section 5.6 describes its evaluation.

5.2 Related Work

There have been several recent efforts in the field of context and activity recognition. We discuss some notable and relevant examples here including those whose goals are similar to ours i.e. they use off-the-shelf devices such as mobile phones rather than proprietary hardware or sensors. We also highlight their limitations and differences with our approach. Moreover, most of these efforts have been isolated and capture a single

dimension of context or activity as opposed to multi-dimensional context and activity recognition.

5.2.1 Environmental Context Recognition

IODetector [41] is a sensing service that runs on the mobile phone and uses light and magnetic field sensors, and cell tower signals in order to detect whether the device is outdoors, indoors, or semi-outdoors[2]. However, sensors such as the magnetic field and light sensors often depend on device manufacturer. As shown in Table 5.2, the range and resolution of these sensors vary with each device. The output of these sensors also varies with time of the day and weather. Hence, extensive calibration and hand tuning as done in IODetector [41] is not a robust and accurate method. In addition, they use the cell tower signal strength but many smart devices such as tablets do not come equipped with the cellular radio. Overall, it has an average accuracy of 88%.

The unavailability of a GPS fix has been used by Ravindranath et al. [42] to infer that the user is in an indoor environment. However, just the availability or unavailability of the GPS fix is not a robust parameter and can lead to many false positives. It is possible to have a GPS fix indoors even if its weak.

TempIO [43] determines environmental context by comparing the environment temperature, measured using proprietary hardware, with the current outdoor temperature obtained from a web service or external thermometer. A major limitation of this work is that web services usually provide temperatures at a coarse granularity of a city or locality rather than an exact fine grained location. Moreover, users have to carry the external

[2] We use the same definitions in SenseMe for Indoor, Outdoor and Indoor-Outdoor respectively.

hardware along with them as temperature sensors may not be available on all devices.

On the other hand, SenseMe performs environmental context recognition using NMEA 0183 data [44], obtained from GPS, which is a standard data specification used for communication between electronic devices such as GPS receivers and other types of instruments. This makes it robust and independent of time, weather, and device manufacturer.

5.2.2 Physical Activity Recognition

CenceMe [45] is mobile phone system which uses accelerometer, GPS, audio and bluetooth to infer human activities such as 'Walking', 'Standing', 'Running', 'Sitting' and 'Vehicle'. It runs on the Nokia N95 with components written in both JME and Symbian C++. To preserve phone resources, certain computations are split between the phone and a back end desktop server. The accuracy of the classifier varies with the activity being classified - high (94%) for 'Walking' but low($<$80%) for the others. CenceMe uses kMeans clustering to identify significant locations inhabited by users. It injects the users' presence and current activity on a social network which can be privacy invasive. Moreover, there are latency challenges in splitting the computation between the phone and a backend server as this slows down the response time of the system and requires network connectivity at all times. In addition, there are privacy concerns and other costs associated with uploading personal data of users to a server or cloud.

Jigsaw [46] is an application with a similar premise but different implementation. It uses three different pipelines for accelerometer, GPS and microphone and runs entirely on the phone. The classifier accuracies for the same set of activities as CenceMe is about

94%. The microphone pipeline detects activities such as brushing, showering, typing, vacuuming etc. and its accuracy ranges from 84% to 88%. While the ability to detect higher number of activities is definitely an advantage, the tradeoff between accuracy, utility and energy consumption must be maintained. The microphone is a power hungry sensor and hence, using it for activity recognition poses significant challenges. Additionally, this raises privacy concerns [47] and can have legal implications as recording audio in any form may require users' permission.

Using accelerometer for activity recognition suffers from several limitations which make it a non-robust method:

- High likelihood of false positives - if a user shakes his phone, it is often labeled as a physical activity.
- Dependency on the gait of a user,
- Dependency on placement of the phone - whether it is placed on the body or if its in a bag or a purse.
- User calibration is required in order to make it independent of orientation and body position [46].

To address these limitations, SenseMe recognizes a user's physical activity based on the speed of the device obtained from the GPS, which makes it agnostic to gait, body position and orientation.

5.2.3 Localization

There exists a wide spectrum of research in indoor and outdoor localization using off the shelf devices as well as instrumented setups. All of them focus mainly on either indoor localization (Wi-Fi based systems such as RADAR [23], Horus [24], Active Campus [29] and Locus [7]) or outdoor localization (GPS based systems such as EnTracked [48] and StarTrack [49]). However, our aim in SenseMe is to enable context-aware localization i.e. localization in both indoor and outdoor environments through technologies that are readily available through the smartphone - GPS and Wi-Fi. This aids in capturing all of the user's locations in an unobtrusive manner using a ubiquitous device.

5.2.4 Social Context Recognition

CenceMe [45] determines social context by scanning a user's environment for recognized bluetooth devices and displaying the number of 'CenceMe buddies' (other CenceMe users) that are around. However, this requires location sharing which can be privacy-invasive. Hence, in SenseMe, we address the general problem of determining how many people are around the user, irrespective of whether they use SenseMe or not. Moreover, many users may not consent to sharing their data especially location with other users and as a result, we do not support that in the current system. Instead, we use bluetooth to recognize a user's social context.

5.2.5 Device Activity Recognition

There are several commercially available smart phone applications that track the application usage of a user and organize the applications based on the Most Frequently/Recently used application. On the other hand, in SenseMe, we add a temporal aspect to the application usage and observe it in tandem with other recognized dimensions.

5.2.6 Logging raw sensory information

In the FunF project [50], Aharony et al. log a variety of sensory information from the devices of 55 users with a maximum temporal resolution of 6 minutes. Wagner et al. [51] undertake the challenge of large scale smartphone usage data collection from Android devices of 21,350 users over a period of 2 years. The data collected includes accelerometer readings, call logs, cell tower scans etc. However, these approaches log low level data without inferring any high-level context or activities from it.

5.3 Key Contributions

Our key contributions in this work are:

1. We present a robust, generic and scalable technique for performing environmental context recognition which is independent of time, weather, and device manufacturer.

2. We present a robust, generic and scalable technique for performing activity recognition, for select physical activities, which is independent of gait, body position and

orientation.

3. We utilize the user's environmental context and physical activity to perform opportunistic context-aware localization using existing technologies that are generic and easily scalable.

4. We capture the user's device activity as well as social context to augment his multi-dimensional situation.

5. We implement the aforementioned techniques as part of a generic system, SenseMe that runs entirely on the smart device and is completely non-invasive.

6. We demonstrate improved or comparable accuracy with respect to other existing systems without requiring any user calibration or input.

Since SenseMe uses GPS and several other sensors, managing power consumption is crucial. To this end, we have implemented a resource efficient duty cycle that employs power conservation techniques to control GPS usage without sacrificing accuracy.

5.4 Training Data Collection

To implement the SenseMe system, we first collected NMEA 0183 and speed data over a period of one month in different environments (such as university, office buildings, high rises, apartment complexes) and in different road conditions (such as highways, downtown, city and local roads).

The NMEA training data samples includes specific NMEA sentences received in a time span of a minute. NMEA sentences consist of several words separated by a ','

Classifier	Environmental Context(%)	Physical Activity(%)	Training time (s)
C4.5	**96.87**	**93.18**	0.1
kNN	92.74	83.77	0.2
Random Forest	92.65	94.79	34

Table 5.3: Comparison of classifier accuracies(%) and average training time(s) for the training datasets

and the first word, also called the data type, defines the interpretation of the rest of the sentence. The two sentences that interest us the most are the GGA and GSA sentences. These sentences contain meta-data about the GPS fix such as number of visible satellites and the Dilution of Precision (DOP) [52]. The DOP is the relative accuracy of horizontal (HDOP) or vertical (VDOP) position as the case may be. It is a number where a smaller value means a higher level of accuracy. Once an NMEA listener is enabled on the device, it starts receiving these sentences every second irrespective of a GPS fix being achieved and independent of the GPS sampling rate. The speed training data samples consist of raw speed values of the device received in a time span of a minute sampled at a minimum interval of 10 seconds.

The data was collected by 4 members of our lab (including the authors) at several times in a day as well as in different weather conditions, to study their effects (if any) on the data. All the participants who collected the data also annotated it carefully to provide ground truth values for each dimension being captured. We computed Pearson's

linear correlation coefficient on the collected NMEA data and determined that the weather specifically outlook had a strong correlation with it. Hence, we removed it from our set of features. The total data collected was approximately equivalent to a continuous run of 168 hours.

We experimented with 3 classifiers on both the NMEA and speed training datasets: C4.5, kNN (k=3) and Random Forest (10 trees). The Weka [53] implementation of each classifier was run on these sets with 10 fold cross validation. Table 5.3 shows a comparison of the classifiers' accuracies as well as average training time for environmental context and physical activity recognition. C4.5 proved to be faster, more accurate and efficient than both kNN and Random Forest for environmental context recognition. For physical activity recognition, it proved to be more accurate than kNN and slightly less accurate than Random Forest but faster than both.

Since smart devices have memory, CPU and power constraints, it is most effective and efficient to use a fast, accurate and light-weight classifier. The C4.5 decision tree is a light-weight classifier as opposed to the kNN classifier (which is an instance based method for classification and hence requires in-memory storage of training instances) and Random Forest (which is an ensemble classifier). As a result, we selected the C4.5 decision tree on the basis of its performance and also because it is fast and not computationally intensive. We implemented it in SenseMe for environmental context and physical activity recognition. There are more advanced techniques like Support Vector Machines which generate a confidence measure with each classification label. This can then be used as an input to a Hidden Markov Model for smoothing. However, in SenseMe we attempt to balance resource usage efficiency with accuracy on a resource constrained platform and

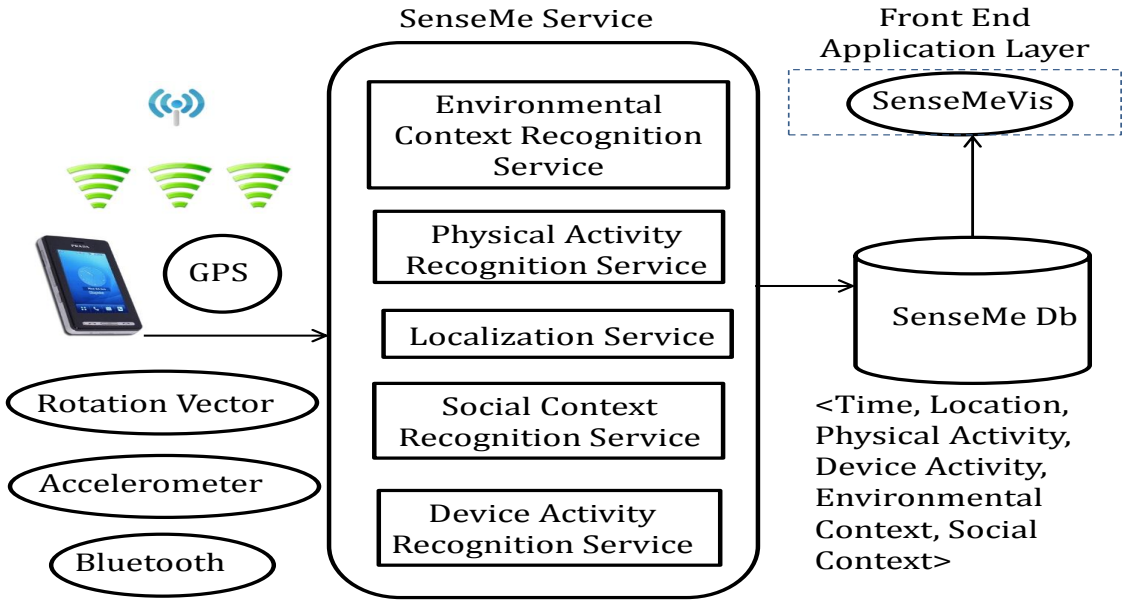


Figure 5.1: Architecture of the SenseMe system

the C4.5 classifier achieves it superbly.

5.5 The SenseMe system

We now describe the SenseMe system in detail.

5.5.1 System Design and Architecture

Figure 5.1 shows the architecture of the SenseMe system. It has been completely implemented on the Android platform. It consists of a background Android service called SenseMeService, which consists of 5 individual services (one for each dimension), a SQLite database called SenseMeDb and a foreground proof of concept visualization called SenseMeVis. The temporal context and activity information recognized by each service is stored in SenseMeDb. SenseMe can run on the user's phone as an application

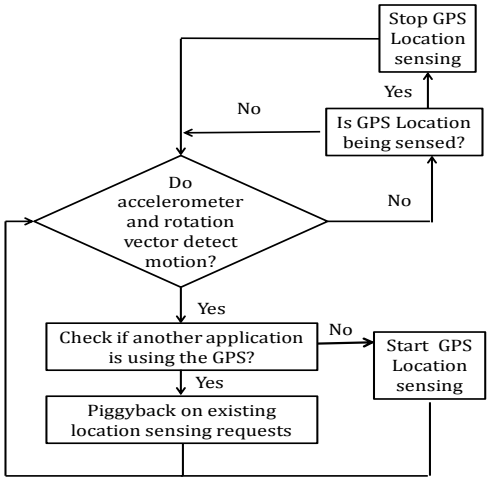


Figure 5.2: GPS Duty Cycle

which can be easily pushed to the background where it continuously functions, collects and processes data. Whenever the application is brought to the foreground, SenseMeVis retrieves the information from SenseMeDb in order to render it on the device display.

5.5.2 GPS Duty Cycle

Zhuang et al. [54] propose several techniques for preserving energy consumption of location based applications specifically on the Android platform. These include: (i) Substitution (replacing a more accurate but energy intensive provider such as GPS with a less accurate but efficient provider such as Network), (ii) Suppression (using low power sensors to suppress the usage of GPS), (iii) Piggybacking (synchronizing the location sensing requests with existing requests) and (iv) Adaptation (adapting the system-wide sensing parameters such as time and distance, when battery level is low). Some of these techniques such as Substitution and Adaptation preserve the battery at the cost of location accuracy.

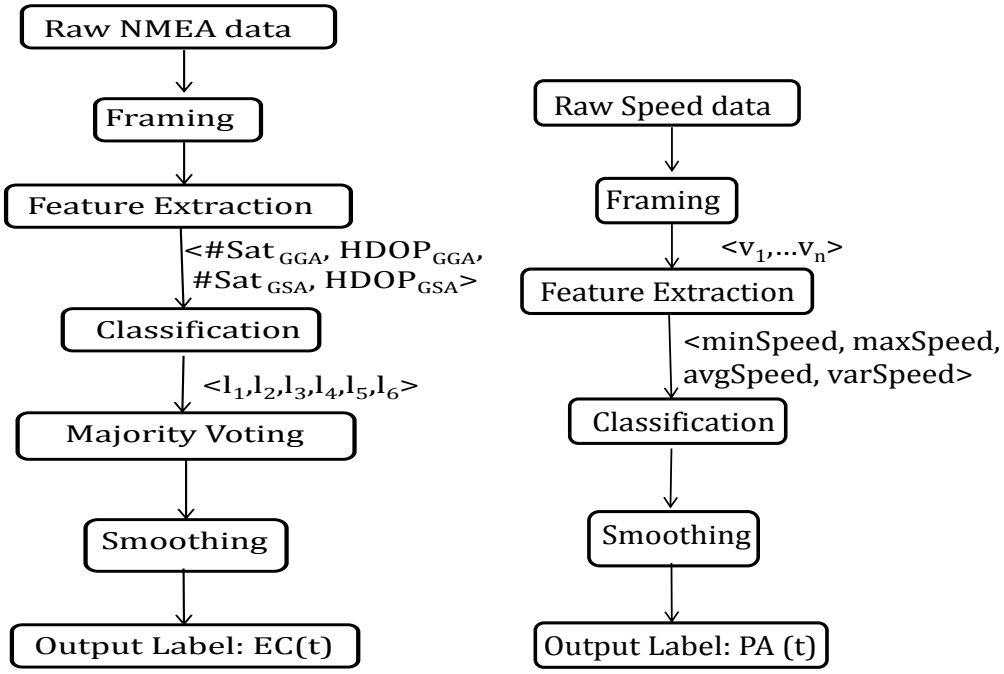
In SenseMe, we utilize the GPS in an energy and resource efficient manner without sacrificing accuracy. Figure 5.2 shows the workflow of the GPS Duty Cycle in SenseMe. It employs the following techniques to conserve power usage:

5.5.2.1 Suppression

Several systems such as SenseLess [55] and Nericell [56] have been proposed to utilize the accelerometer as a control for the GPS. On a similar line, SenseMe uses the linear accelerometer as well as the rotation vector sensor as a means to control or suppress the GPS usage. Since they are light-weight sensors and consume very little power, they can be used as an effective trigger for turning the GPS on when motion is detected and off otherwise.

We calculate the ℓ_2 norm root of the linear acceleration and rotation vectors to get the overall linear acceleration and rotation of the device. Some of the samples can be noisy and hence we do not rely on raw values. Instead, we calculate the average and variance of the acceleration and rotation values obtained from 150 samples to obtain four parameters, termed as μ_{acc}, σ_{acc}, μ_{rot} and σ_{rot} respectively. These samples can be spread over 10 to 30 seconds according to the sampling rate, as specified in the Android API [57].

We empirically studied the linear acceleration and rotation vector values obtained from different Android devices when a user was stationary and when he/she was in motion. We determined suitable thresholds for all the parameters. If all the parameters are greater than their respective threshold values, it implies that the device is in motion. In all other cases, it implies that the device is not moving.



(a) Environmental Context Recognition Service (b) Physical Activity Recognition Service

Figure 5.3: Environmental Context and Physical Activity Recognition Services

5.5.2.2 Piggybacking

When the linear acceleration and rotation vector sensors detect motion, the system checks if another application is already using the GPS. If yes, it piggybacks location sensing on the existing requests. If not, it explicitly starts the GPS for location sensing.

5.5.2.3 Sensing Adaptation

We attempt to balance the trade-off between accuracy and power consumption, by employing an optimum sensing interval (in terms of time and distance) for location updates [58]. This is based on the intensity of the physical activity (walking as opposed to driving) of the user and further aids in preserving battery life and consumption.

5.5.3 The SenseMe Service

The SenseMe service consists of the following individual services:

5.5.3.1 The Environmental Context Recognition Service

This service uses NMEA 0183 data from the GPS to recognize the environmental context for a user. Figure 5.3(a) shows the pipeline for it. It has the following stages:

- Framing - Since NMEA sentences are received every second, we operate on frames of these sentences where each frame consists of τ= 10 seconds.

- Feature Extraction - For each frame, we average the number of satellites and the HDOP for both the GSA and GGA sentences to create a feature vector of the form $< \#Sat_{GGA}, HDOP_{GGA}, \#Sat_{GSA}, HDOP_{GSA} >$.

- Classification - This feature vector is then used as input to a C4.5 classifier, to generate either of the following labels - 'Indoor','Outdoor','Indoor-Outdoor'. Thus, we have a vector of six labels $< l_1, l_2, l_3, l_4, l_5, l_6 >$ for every minute.

- Majority Voting - We then perform majority voting on this vector to generate an environmental context value, *EC(t)*, for each minute.

- Temporal Smoothing - For smoothing outliers, we have implemented this service as a stateful service and modeled the environmental context as a 1st order Markov Chain, where the current state i.e. environmental context at time t, EC(t), is dependent only on the previous state at time t-1 i.e. EC(t-1). Figure 5.4 shows the

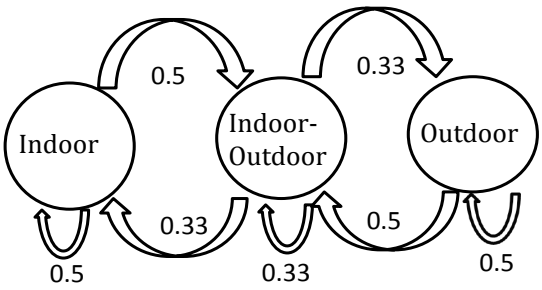


Figure 5.4: Indoor-Outdoor transition Markov Chain

transition probabilities for the Markov Chain and is derived from the fact that any person when moving from 'indoor' to 'outdoor' state or vice versa will always go via the 'indoor-outdoor' state.

5.5.3.2 The Physical Activity Recognition Service

This service uses raw speed data from the GPS to recognize the physical activity of a user. Figure 5.3(b) shows the pipeline for it. It has the following stages:

- Framing - According to the duty cycle, the service receives speed data at a minimum interval of 10 seconds and a maximum interval of 60 seconds depending on the motion of the device. Each frame consists of τ = 60 seconds and for each frame, we get a raw speed vector S = $< v_1, .., v_n >$, where n $\in$ [1,6].

- Feature Extraction - We extract statistical features from this raw speed vector to generate a feature vector of the form <minSpeed, maxSpeed, avgSpeed, varSpeed> where *minSpeed* is the minimum, *maxSpeed* is the maximum, *avgSpeed* is the average and *varSpeed* is the variance, of all the speed values in S.

- Classification - This feature vector is used as input to a C4.5 classifier, to generate

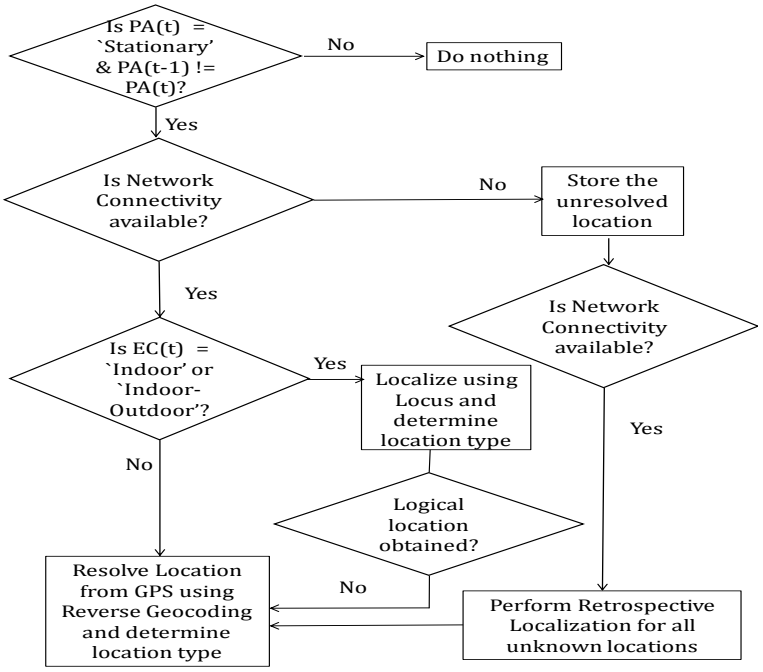


Figure 5.5: Opportunistic context-aware Localization performed by the Localization service

one of the following labels - 'Stationary','Walking', 'Running' and 'In-vehicle'. Thus, we get a Physical Activity value, *PA(t)*, for the user for every minute.

- Temporal Smoothing - This service is a stateful service and we store the values PA(t-1) and PA(t-2) at any given instant of time. We use a sliding window smoother of size 3 to smoothen any outliers out.

5.5.3.3 The Localization Service

Although SenseMe senses the raw location of the user in accordance with the GPS duty cycle, we resolve the location to a logical address only when the user is 'Stationary'. This serves three purposes: (i) It reduces irrelevant contextual information in the form

of multiple logical locations that the user could be passing through. For instance, if the user is driving on a highway, he/she will pass through multiple locations within a few seconds. (ii) It prevents unnecessary network bandwidth usage. (iii) It can help determine the user's mobility patterns such as daily commute routes and travel paths. It also capture the places, where they spend a significant amount of their time (such as home, work, restaurants and coffee shops), with fine-grained accuracy.

Figure 5.5 shows the workflow of the Localization service. As shown, the Localization Service performs context-aware localization on the basis of the user's current environmental context i.e. EC(t) and the current physical activity i.e. PA(t). Thus, if PA(t) is determined to be 'Stationary', after a transition from another physical activity, SenseMe localizes the user to a logical indoor or outdoor address. If EC(t) is 'outdoor', it resolves the location obtained from the GPS (which is in latitude and longitude format) to provide a logical address using the Android Reverse Geocoding API. If EC(t) is 'indoor' or 'indoor-outdoor', it employs the Locus [7, 8] system to determine the Room #, Floor #, and Building the user is in. The main benefits of Locus are that it is a calibration-free, readily deployable, scalable and robust system for floor as well as location determination in multi-story buildings. It relies on existing infrastructure and off-the-shelf mobile device capabilities, and requires no proprietary hardware to be installed.

If an indoor location could not be obtained using Locus (due Wi-Fi being off or disconnected), SenseMe resolves the last sensed outdoor location so that a coarse-grained location for the user can be obtained. This is only to ensure that the user is always localized whenever he/she is stationary and there are no unknown locations in the user's location history. If a network or data connection is not available at any time, SenseMe

performs what we term as *Retrospective Localization*. It stores every unresolved location where the user was stationary and localization could not be performed. As soon as a data or network connection is available, SenseMe performs opportunistic localization of all unknown locations.

Once localization has been done and a logical address is obtained for the location, the Localization Service also determines the location type if available. If the user is outdoors, the location type is a Foursquare category obtained via the FourSquare Venues API[3] (such as "College Academic Building") or a Google Place type via the Google Places API[4]. If the user is indoors, the location type has two fields: a coarse-grained "Building Type" which refers to the type of the Building or establishment the user is in and a fine-grained "Room Use" which specifies the category of the room the user is in, for example, "Research Laboratory". This is useful meta information that can enable semantic place prediction.

5.5.3.4 The Device Activity Recognition Service

This service determines the task that the user is engaged in on his smart device, for instance, a phone call, web browsing, or using a navigation application such as Maps. It polls the device OS every minute to determine which application is running in the foreground while the screen is on and active. If a media file such as an audio or video file is being played, SenseMe also records its metadata (track, artist, album name etc.).

---

[3] https://developer.foursquare.com/overview/venues.htm [4] https://developers.google.com/places/

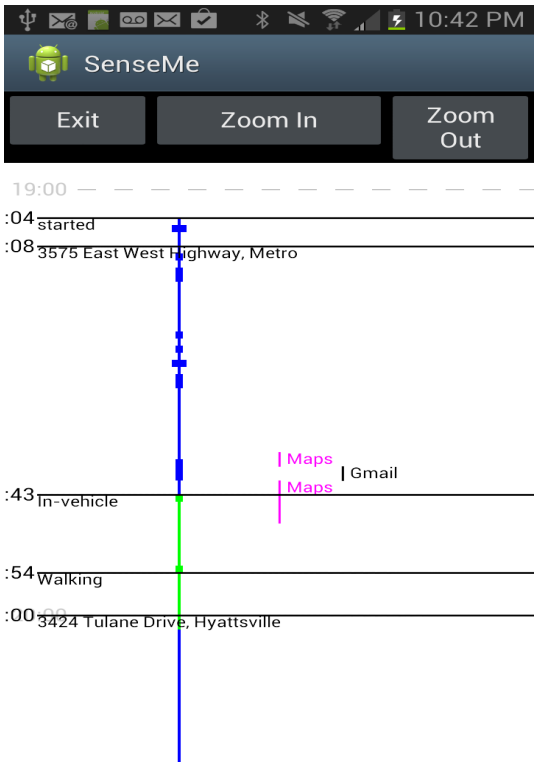


Figure 5.6: Screenshot of the SenseMeVis timeline (best viewed in color)

5.5.3.5 The Social Context Recognition Service

This service uses bluetooth to scan the user's environment every 2 minutes to determine a *social meter* i.e how many people and/or their devices are around the user in any given time interval of 2 minutes. It obtains the device class [59] of all the devices it hears and filters them on the basis of their type. Thus, portable devices such as phones, laptops, hand held PDAs and wearable devices are counted, while other devices are not, as these devices are more likely to be carried around by people. Since bluetooth scans take about 60 seconds on an average to complete [60], we have set a threshold interval of 60 seconds between successive scans in order to balance power consumption with accuracy.

5.5.4 Proof of Concept Visualization - SenseMeVis

As mentioned earlier, SenseMe continuously operates as a background service. Whenever the application is resumed or brought to the foreground, the multi-dimensional context and activity information recognized by it is rendered on the device display via a foreground proof of concept visualization called SenseMeVis. The information is visualized for a maximum period of 24 hours in retrospect from the current time instant.

Figure 5.6 shows a screenshot of SenseMeVis visualizing the multi-dimensional contextual information recognized by SenseMe while it was running on a user's device. It is a scrollable timeline that runs from top to bottom, showing the multiple dimensions of context and activities. Time units are fixed, thus, allowing the user to compare the information without needing to interpret how long something took place. The time and date at which SenseMe was started is at the top of the visualization. Markers on the left hand side of the visual along with dashed lines partition the hours to help temporally anchor the rendered information across the screen.

There are three types of visualizations that portray the 5 dimensions of context and activity that SenseMe recognizes:

1. The vertical central bar of varying thickness (known as the *Context bar*) represents environmental and social context. The environmental context is conveyed through the color of the bar - a blue segment indicates the user was indoors, green symbolizes outdoors and red is the indoor-outdoor state. The thickness of the bar conveys the social context. The wider the bar, the more the number of people and portable devices the user is surrounded by. A very skinny bar indicates the user is not near

anyone.

2. The horizontal solid black lines and subsequent labels partition the timeline based on location and physical activity. As mentioned earlier, the localization service performs context-aware localization only when the user is 'Stationary' after a transition from another activity. Similarly, for clarity and legibility of the visualization, we display either a location (if the physical activity is not 'Stationary') or a physical activity. Thus, if the user were stationary, SenseMe displays his location but if the user were in motion, SenseMe displays his physical activity. This simplification prevents the rendering of multiple locations in a short period of time. For example, if a user is in a car traveling at 60 mph, rather than displaying all the locations he might pass through, we summarize that timeframe with his physical activity - 'In vehicle'.

3. The vertical lines of varying colors, on the right side of the context bar, represent the device activity of the user. Each activity is assigned a column and a color. This ensures that all entries for a given activity always fall within the same column and are colored the same to facilitate ease in understanding of the visualization. As there is no limit to the number of unique activities a user could be doing, activities may share a column but the color will be different allowing the user to distinguish between them easily.

As shown, the current run of SenseMe is started around 7:04 pm. It recognizes that the user is indoors and stationary so it localizes her and resolves a logical address for her location. She is indoors for about 35 minutes and is surrounded by varying number of

people during this time period. She is also performing different activities on her device during this period such as checking her email and accessing a navigation application. She then goes outdoors and is driving alone for about 11 minutes. Finally, she parks her vehicle, starts waking and gets home around 8 pm.

5.6 Evaluation

5.6.1 Evaluation Metrics

SenseMe is a versatile system for continuous, on-device, and multi-dimensional context and activity recognition. Hence, it needs to meet the following qualitative and quantitative requirements:

- High accuracy and scalability - As a generic system that can be used by multiple users on different devices, it should be highly accurate when recognizing context and activities. It should also be scalable to a large and varied set of users.

- Generality and Robustness - It should be general and robust enough to be used at any time in any environment.

- Universal Applicability - It should be applicable to any device and independent of device manufacturer.

- Minimum latency and Robustness to network failure - It should provide responses in real time even when there is sparse or no network connectivity.

- Non-invasive with minimum user calibration required - It should be non-invasive,

capable of operating in the background and require minimum user input or calibration.

- Privacy preserving - Since a user's context history has privacy implications, the data should be kept secure, confidential and shared only with the user.

- Energy efficiency - Being a system running on mobile devices with constrained energy budgets, it should use resource efficient methods and duty cycles.

We now evaluate SenseMe against each of these requirements.

5.6.2 Methodology

To evaluate SenseMe, we conducted two live deployments of 2 weeks each. In the first deployment, we recruited 8 subjects - 7 of whom were male Computer Science graduate students while one was a male post doctoral associate. We incorporated the feedback from them and fixed minor bugs in the system. We then conducted a second deployment, for which we recruited 7 subjects (5 male and 2 female) who were professionals working across USA. None of these test subjects had participated in training data collection. In both the deployments, we installed SenseMe on the subjects' personal devices as giving temporary devices to them for a study may affect their interaction and usage of the device and bias results.

All the subjects were asked to run SenseMe on their devices, in the background, for a period of 2 weeks while going about their daily life. They were also asked to keep a journal of their physical activities, locations, environments and number of people around them throughout the day. As an incentive, the subjects were provided monetary com-

Ground Truth	SenseMe		
	indoor	outdoor	indoor-outdoor
indoor	0.984	0.016	0.0
outdoor	0.07	0.93	0.0
indoor-outdoor	0.11	0.07	0.82

Table 5.4: Confusion matrix for environmental context recognition for test data

pensation. At the end of the two-week deployment period, the subjects were asked to submit their journals as well as the SenseMeDb databases, which contained the context and activity recognition information recognized by SenseMe. This data was used as a test dataset to determine the accuracy of the different SenseMe services. The estimated value for each dimension, as determined by SenseMe, was compared with the Ground Truth values provided by the subjects as part of their journals.

We used this methodology as we did not want SenseMe to be obtrusive and disrupt their daily life. Moreover, this methodology allowed the context and activity information to be captured in a real life practical scenario, thus, making the evaluation more effective.

5.6.3 Accuracy Results

Since Environmental Context and Physical Activity Recognition services employ multi-label classification, we define accuracy for them as:

$$a = \frac{T \cap P}{T \cup P}$$

where T is the set of ground truth and P is the set of predicted labels for all instances.

Ground Truth	SenseMe			
	Stationary	Walking	Running	In-vehicle
Stationary	1.0	0.0	0.0	0.0
Walking	0.0	0.93	0.04	0.03
Running	0.0	0.0	0.95	0.05
In-vehicle	0.0	0.01	0.039	0.951

Table 5.5: Confusion matrix for physical activity recognition for test data

For Localization and Device Activity services, we measure the mean absolute prediction error and define accuracy as

$$\text{a} = \frac{\text{\# of instances where estimated value = ground truth}}{\text{Total \# of instances}}$$

For Social Context Recognition service, we measure the mean relative prediction error and define accuracy as

$$\text{a} = 1 - \sum_{i=1}^{N} \frac{|\text{ estimated value - ground truth }|}{\text{ground truth}}$$

Table 5.6 shows the overall accuracy of all the services.

5.6.3.1 Environmental Context and Physical Activity Recognition Services

As shown, the environmental context recognition accuracy is higher than IODetector [41] which reports an overall accuracy of 88%. The physical activity recognition accuracy is more than CenceMe [45] for all the common activities detected. It is comparable to

SenseMe service	Overall Accuracy (%)
Environmental Context Recognition	91.23
Physical Activity Recognition	95.75
Context-aware Localization	93.12
Device Activity Recognition	99.1
Social Context Recognition	87.5

Table 5.6: Accuracy of SenseMe Services (%)

Jigsaw [46] though it detects a slightly higher number of activities than SenseMe. Tables 5.4 and 10.14 show the confusion matrices for environmental context and physical activity recognition on test data. We believe these results can be improved even further by using more sophisticated classification and smoothing techniques.

5.6.3.2 Device Activity Recognition Service

The number of tasks that a user performs on a smart device during the day can be huge and maintaining a detailed journal for each of them can be quite burdensome. Moreover, because our temporal resolution is a minute, we do not capture device activities that last less than that. Hence, we did not ask the subjects to maintain detailed logs for every task that they performed on their device. Instead, we asked them to look at the visualization periodically to check the accuracy of the device activity and log that. Based on their logs, it was extremely accurate.

5.6.3.3 Localization Service

For all the subjects, the locations recognized by SenseMe were same as or close to their actual locations in both indoor and outdoor environments. This is mainly because the accuracy of this service is directly dependent on the technique being used for localization. Reverse geocoding is usually not 100% accurate as it is often difficult to resolve every latitude/longitude to a logical address. For Locus, the floor accuracy is approximately 95% and it achieves room level accuracy on the floor.

5.6.3.4 Social Context Recognition Service

Most users logged an estimate of the number of people around them and results from this service matched that reasonably. However, a limitation of determining social context using bluetooth is that it is constrained by the distance over which bluetooth operates. As a corner case, one of the subjects mentioned that he was at a wedding with several people in a large hall and SenseMe recognized the people on his table only. Though co-location might be a better alternative for determining social context, it requires location sharing by the users which they may not consent to. Hence, it is essential that a trade-off between accuracy and privacy is maintained. Since bluetooth performs reasonably well (being off by a small margin only) we have used it in our current implementation.

Battery usage metric	Google Nexus	Samsung Galaxy	Motorola Xoom
Maximum consumption (%)	18	28	40
Average consumption (%)	16	27	37.4
Average runtime (hrs)	24	24	24

Table 5.7: Comparison of Battery consumption and runtime during a 24 hour continuous run of SenseMe

5.6.4 Qualitative Results

5.6.4.1 General, scalable and universally applicable

SenseMe has been tested on 15 subjects with varied schedules and mobility patterns. All the subjects carried devices made by different manufacturers such as Samsung, HTC, LG etc. and running different versions of Android OS ranging from 4.0 to 4.2. SenseMe ran without any major errors on most of the devices as it is based on techniques that are independent of device or manufacturer.

5.6.4.2 Minimum latency, robustness to network failure, and privacy preserving

In SenseMe, all computation and processing is carried out on the device and it does not require an external server. As a result, there is a minimum latency of a minute (which is the smallest granularity of computation). Only localization needs a network or data

connection, but the localization service pipeline ensures that the system carries out opportunistic localization. Moreover, the user's data is kept private and confidential on the mobile device and is visible only to him/her, thus, mitigating privacy concerns.

5.6.4.3 Non-invasive and calibration-free

SenseMe is non-invasive i.e. it can easily run in the background in order to collect and process user's data without the need for any intervention. Also, it performs context and activity recognition using techniques that are agnostic to orientation, body position, time or weather, and hence, no calibration by the user is required.

5.6.5 Resource Utilization Results

We used two methods to determine SenseMe's battery consumption[5]:

5.6.5.1 Measuring the Average and Maximum % of battery consumed

For the average case, we measured the battery consumption of SenseMe while it was running in the background continuously for 24 hours on a user's device while he went about his daily activities.

To determine the maximum limit, we measured the battery consumption of using the GPS alone, which can be the most power consuming component of our system, in the worst case scenario. This, of course, varies with each user since the mobility and usage patterns can be quite different for everyone. Hence, we first determined the fraction of time the GPS was sensed during the entire run of SenseMe. Our analysis indicated that as

[5] The actual battery consumption often depends on device usage and its age.

a worst case, the GPS was actively sensed for 15% of the total time for which SenseMe ran, mainly due to an effective duty cycle that uses three different power conservation techniques. Thus, we estimated the maximum energy consumption of SenseMe by sampling the GPS for 15% time of a continuous run of 24 hours i.e. 3.6 hours, on battery without charging.

5.6.5.2 Measuring the average runtime of battery without charging

We tested the average runtime of the device battery (without the need for recharge) in a day when SenseMe was running in the background on it. The screen was set to a low brightness level since the screen display can consume a major chunk of the battery.

Table 5.7 summarizes the results of battery consumption and runtime on three different devices. All the devices were used moderately while these experiments were conducted. As shown, the average and maximum battery consumption were at most 40% and the battery lasted for more than 24 hours during a continuous run of SenseMe without requiring a recharge.

5.6.6 User feedback and Survey

On conclusion of each of the live deployment periods, we held a short interview with each subject to discuss which dimensions they found useful and interesting and to evaluate their user experience with SenseMe. We also wanted to uncover any issues with the system and ask them for open-ended valuable feedback.

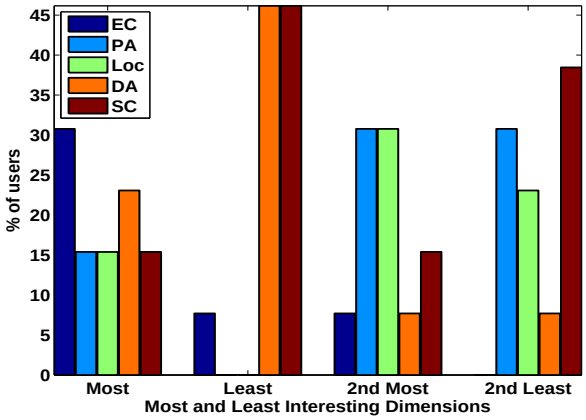


Figure 5.7: Dimensions users found most and least interesting

5.6.6.1 Most and Least Interesting Dimensions

Figure 5.7 shows the plot for the dimensions users found most and least interesting (EC = Environmental Context, PA = Physical Activity, Loc = Context-aware Location, DA = Device Activity and SC = Social Context). As shown, a significant proportion found the Environmental Context dimension to be most interesting. Two of the subjects even referred to this dimension as "Intriguing" and "Insightful". A majority of the users found the Device Activity and Social Context dimensions to be least interesting.

5.6.6.2 Perceived accuracy

In order to quantify the subjects' perceived accuracy of the system, we asked them to look at the visualization at least once a day, during the deployment period, in addition to keeping a journal of their day. This was done in order to help them gauge whether the captured information represented their daily life log accurately. During the interview, the subjects rated SenseMe's accuracy on a Likert scale from 1 (Very Accurate) to 10 (Very

Inaccurate). The average score across the 15 subjects was 2.2 suggesting that the system was highly accurate in capturing their daily life log.

5.6.6.3 Feedback about benefits and insight

One of the subjects acknowledged that the multi-dimensional information could *"help me identify patterns which I didn't realize before"*, indicating the benefit of its temporal aspects and long-term continuous usage. Another subject wanted to employ the location and physical activity dimensions for increasing his productivity as they could *"help me determine if I followed my schedule."*

Some subjects expressed the desire to use the system in a more reflective manner. One subject mentioned that she would like to use the environmental and social context dimensions in a *persuasive* manner to help ensure she gets outside and interacts with people in times of heavy and prolonged workloads. Reflecting on the information captured during her two-week user study period, she noted *"Man, I live in a box..."* Another subject, emphasizing the reflective value of the device activity dimension, stated, *"It could help me find the apps I use way too much and also the amount of time that I wasted."*

Chapter 6: To Sense or not to Sense: An Exploratory Study of Privacy, Trust and other related concerns in Personal Sensing Applications

As part of the SenseMe user study explained in Chapter 5, we also conducted an exploratory study [19] of privacy, trust, risks and other concerns of users with smart phone based personal sensing systems and applications. The results of this study are presented in this chapter.

6.1 Introduction

With the advent and ubiquity of smart devices such as smartphones and tablets, that come equipped with an increasing range of sensory, computational, storage and communication capabilities, a number of applications (also referred to as 'apps') and systems that can **sense** the user have emerged. This includes 'Hard Sensing' carried out through hardware sensors as well as 'Soft Sensing' done via application access or content extraction. These applications focus on different types of context and activity recognition such as indoor and outdoor detection, physical activity recognition, and localization [9, 41, 43, 45, 46] and in different application areas such as location based services, social networking, health care

etc. Commercial examples of such apps include Google Now[1] and Tempo[2] that sense and model a user's behavior based on his browsing history, emails and calendar data in order to provide him with personalized and relevant content.

Due to increasing proliferation of smart devices, many users carry them around all the time and perform a majority of their day to day activities (such as web browsing, listening to music etc.) through them. In addition, users store a significant proportion of personal data such as photographs, text messages, emails, calendars, and financial information etc. on the phone. As a result, personal sensing applications (that track a user's location, activities, behavior, browsing history and calendar content via his smart device) have greater privacy implications than traditional personal computers based applications as well as sensing applications that run on closed, proprietary devices such as Fitbits.

We conduct an exploratory study of privacy, trust, risks involved and various other related concerns of users with such personal sensing systems and applications. In particular, we study several behaviors of users including those pertaining to:

- Their general privacy concerns with personal sensing applications and apprehensions about misuse of their sensed data.
- Data sharing - Their willingness to share data with other users, friends on a social network and other software.
- Sensitive Data Collection - Their willingness to allow the sensing app or system to collect sensitive health data as well as their perceived trade offs involved in storing this data on their smartphones vs on a cloud or server.

[1] `http://www.google.com/landing/now/` [2] `http://tempo.ai/`

- Benefits to users - Their willingness to use a sensing app or system that stored sensed data on a server or cloud if it made smart decisions for them which had strong quantized benefits in terms of saving them time and money.

- Brand Recognition and User awareness - Their usage of services such as email, navigation etc. provided by a major technology company and their awareness about tracking of location, emails, as well as search, browsing, and video history by that company.

- Brand Trust - Their willingness to use services provided by the major technological company, despite the knowledge that their information is tracked and stored, if the services had strong quantized benefits such as saving them time and money

- Brand Reputation - Their willingness to use a sensing app if it were developed by a major technology company instead of a research prototype and saved them time and money.

We report results obtained from a live deployment with a smart phone sensing application and a web-based study involving 70 participants in all. Our results show that users are concerned that their sensed data[3] can be misused, used for personal identification and tracking or for commercial purposes. They are also concerned that the system or app may have unauthorized access to sensitive content on their devices and may be sharing their data with an external third party or sending it to a cloud or server. Moreover, the users want more control of what data they want to share, where it should be stored and how it should be mined. However, they are willing to trade privacy for additional significant ben-

[3] We use information and data interchangeably to refer to the users' sensed information.

Category	Pertinent question	Type
App Installation	"Before installing any smart phone application, do you read the EULA and privacy rules?"	Likert scale
Data misuse	"Are you concerned that the data sensed and collected by a smart phone sensing app could be misused?"	Likert scale
Privacy concerns	"What privacy concerns would you have with a smart phone sensing app?"	Free text
Data control	"If a personal sensing app allowed you to limit the data collected, what would you limit and why?"	Free text
Data Sharing	"If a smart phone sensing app shared your sensed data (such as activity or location) with other users of the app in order to alert them that you are nearby (say for finding friends) OR with your friends on a social network that you used often OR with other software, services or systems for user modeling purposes , would you use it?"	Likert scale
Data Storage and Retention	"Suppose the data sensed and collected by SenseMe or a similar smart phone sensing app was stored in a server or cloud (in an encrypted but unanonymized format) OR (in an encrypted and anonymized format). Would you use the app if it made smart decisions for you?"	Likert scale
Sensitive Data Collection	"If the smart phone sensing app or system was able to sense and collect sensitive health data, such as heart rate, blood pressure, etc. while keeping this data on the phone OR while sending this data to a cloud or server OR while sending this data to a cloud or server and using it ONLY for saving lives , would you use it?"	Likert scale
Benefits to users	"Would you be willing to use SenseMe or a similar app if it stored sensed data on a server/cloud and made smart decisions for you that saved you 10 minutes of your time OR 1 hour of your time OR 1 % of your salary OR 10% of your salary?"	Likert scale
Brand Recognition	"Do you use any of the following services: email, navigation, Personal Digital Assistant (PDA), Location based services (LBS), cloud storage and search, provided by a major technology company?"	Yes/No
User Awareness	"Are you aware that the major technology company, which is mentioned above, tracks your location, emails, search history, browsing history, video history, and location searches?"	Yes/No
Brand Trust	"Now that you are aware that this company has the ability to track so much information about you, would you be willing to use the services provided by it if they could save you 1% OR 10% OR a significant fraction of your salary?"	Likert scale
Brand Reputation	"Would you be willing to use SenseMe or a similar app if it were developed by the major technology company mentioned above?"	Likert scale
Brand Reputation	"Would you be willing to use SenseMe or a similar app if it were developed by the major technology company mentioned above and saved you 1% OR 10% OR a significant fraction of your salary?"	Likert scale
Brand Reputation	"Would you be willing to use SenseMe or a similar app if it were developed by the major technology company mentioned above and saved you 10 minutes OR 1 hour of your time?"	Likert scale

Table 6.1: Privacy and trust related questions from our study questionnaire

efits or if their sensed information is used for effective and beneficial causes. In addition, they are willing to trust reputed technology companies, which have a brand name, with their data if the benefits are significant despite being aware that their data is sensed and collected by these companies. Based on these results, we propose a few design guidelines for designers of personal sensing apps and outline some interesting directions for future research. To the best of our knowledge, other papers have not addressed such a broad spectrum of concerns with personal sensing applications.

The rest of the chapter is organized as follows: Section 6.2 describes the methodology used for conducting our evaluation. Section 9.6.5 describes results extracted from the evaluation and Section 6.5 explains design guidelines inferred from them. Finally, we discuss related work in Section 11.7.

6.2 Methodology

The evaluation and results presented in this chapter come from two studies:

- SenseMe system user study - Exit interviews conducted after two, 2-week long live deployments of the SenseMe [9] system with 15 subjects
- Web-based personal sensing privacy study - Web based surveys conducted among a population of 55 subjects that used or were aware of several smart phone based personal sensing applications but did not use SenseMe or take part in its user study.

We briefly describe these two studies now.

6.2.1 SenseMe System User Study

SenseMe is an Android based system that leverages the smartphone and its various sensors such as accelerometer, GPS, WiFi, and Bluetooth in order to perform continuous, on-device, and multi-dimensional context and activity recognition for a user. It achieves this in a robust, automated, accurate, scalable, power efficient and non-invasive manner. SenseMe captures the following dimensions of a user's situation:

1. *Environmental context* - whether the user is outdoors (outside a building), indoors (inside a building), or indoor-outdoor (inside a building - near the door or a window),

2. *Location* - indoor/outdoor locations and type of location,

3. *Physical Activity* such as Walking/Running etc,

4. *Device Activity* - the task the user is currently engaged in on his/her smart device (checking mail, phone call)

5. *Social context* - how many people are around the user.

In SenseMe, all computation and processing is carried out on the device without requiring an external server. Moreover, the users' data is kept private and confidential on their devices and is visible only to them in order to mitigate privacy concerns.

Two user studies, each lasting 2 weeks, were conducted for evaluating SenseMe. The studies involved 15 participants from the USA and their ages ranged from 21 to 40 (μ =27.5). 46.7% of the participants were female and 53.3% were male. The participants

reported a variety of occupations including software engineers, health and wellness coordinators, educators, post doctoral associates etc. However, the majority of the subjects were students. Self-reported completed levels of education ranged from college to doctorates.

In both the studies, SenseMe was installed on the subjects' personal devices in order to capture the context and activity information in a real life practical scenario, thus, making the evaluation more effective. All the subjects were asked to run SenseMe on their devices, in the background, for a period of 2 weeks while going about their daily life. They were also asked to keep a journal of their activities, locations, environments and number of people around them throughout the day. This allowed them to present an actual portrayal of the day for an effective evaluation of the application.

On conclusion of the user study periods, each subject was interviewed to discuss which dimensions they found useful and interesting as well as to evaluate their user experience. They were also asked in detail about their privacy concerns with such personal sensing applications, the data it could sense, where the data should be stored, how it could benefit them etc. Results that focused on the system performance, accuracy and resource utilization of SenseMe are presented in [9]. This chapter presents results on the participants' responses to questions on privacy, trust and other related implications of smart phone based personal sensing applications such as SenseMe.

6.2.2 Web-based Personal Sensing Privacy Study

55 participants, who used or were aware of several smart phone based personal sensing apps, were recruited via social media, emails and word of mouth to participate in a survey of 41 questions. Their ages ranged from 21 to 50 years (μ = 31.5) and their demographic distribution was as follows: 60% from USA, 18.2% from United Kingdom and 21.8% from India. 40.1% of the participants were female and 59.9% were male. The participants reported many occupations including researchers, engineers, full time graduate students, consultants, entrepreneurs etc. Some of the participants were home makers. Self-reported completed levels of education ranged from some college to doctorates.

To maintain consistency, these participants were first given a short description of SenseMe. They were then given the same privacy related questionnaire as the subjects in the SenseMe User Study. The goal was to survey a large number of subjects, with no firsthand experience with the application, as part of the same study.

6.3 Results

Table 6.1 shows the various categories of open-ended and quantitative questions pertaining to privacy, trust and other related issues from our two studies. A majority of the quantitative questions had responses on a Likert Scale ranging from 1 (Highest or Most Likely) to 8 (Lowest or Least Likely) while others were either free text based or dichotomous (Yes/ No). We present results for each now.

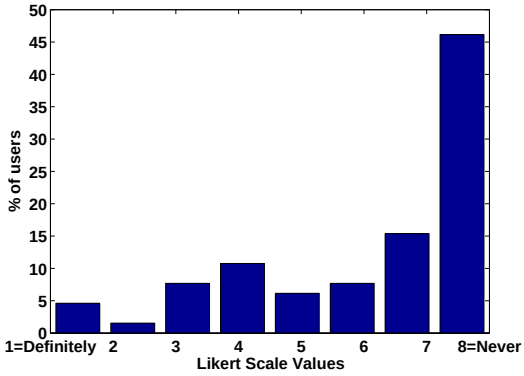


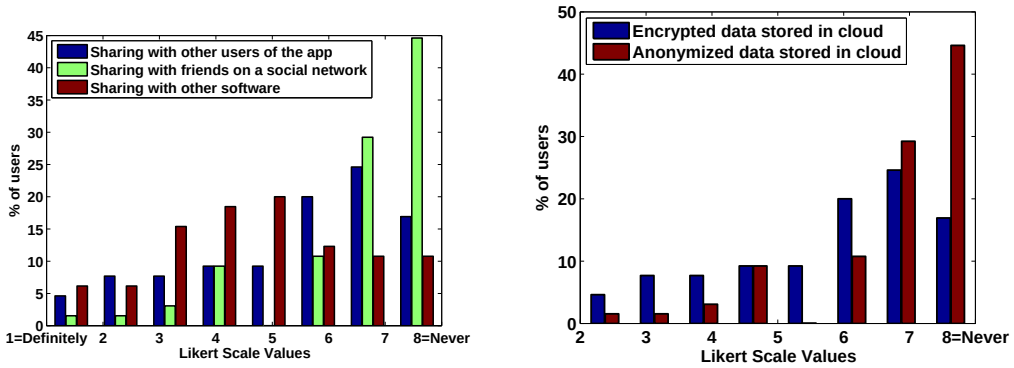
Figure 6.1: App Installation and EULA Responses

6.3.1 App Installation and EULA

We first investigated whether the subjects read the End Users' License Agreement (EULA) and kept track of what apps they were installing on their devices. Figure 6.1 shows the distribution ($\mu = 6.28$, $\sigma = 2.14$) of responses of all the participants to the App installation question. More than 45% of the users never read the EULA (which includes the data, sensors and services on the phone that the app would access) while only 5% responded that they definitely read it. These results support the findings of Staiano et al. [61] that most users do not read the Terms of Service of smartphone apps. These findings also corroborate with those of Good et al. [62] with respect to users not reading computer software license agreements.

6.3.2 General Privacy Concerns

We first gauged whether the subjects had concerns about their sensed information being misused. The participants' responses had a $\mu = 1.1$ and $\sigma = 0.29$. As evident by the low standard deviation and the high mean, all of the users were apprehensive that their sensed



(a) Distribution of Data Sharing Responses

(b) Distribution of Data storage and Retention responses

Figure 6.2: Responses to Data sharing and Data Storage and Retention Questions (best viewed in color)

information could be misused and hence, expressed specific privacy concerns about its collection, monitoring and storage. Out of the 70 participants, 57 responded to this question. We applied the open coding method [63] to their responses, which is a standard method for analyzing qualitative data. We categorized the responses into several categories:

6.3.2.1 Concerns regarding sensed information being used for Personal Identification and Tracking

18 of the 57 participants (31.6%) expressed concerns regarding the information being used for Personal Identification and Tracking. Some of the comments include:

- *"How much can be inferred about me from my app usage"*
- *"If people can hack it to tell when I am out of my house."*

- *"Whether or not 3rd parties could access data to determine patterns of life"*
- *"No Location Tracking"*
- *"If it is known that all members of a household are not at home by their locations, then it is possible that someone could use the system to find the best time to rob a house."*

6.3.2.2 Concerns regarding sensed information being used for Commercial Purposes

6 of the 57 participants (10%) said that they did not want their sensed information to be used by companies for commercial purposes such as targeted advertising. One subject remarked *"I don't want to be shown ads based on what videos I watched"*.

6.3.2.3 Concerns regarding sensed information being used for Unintended or Undeclared Purposes

4 of the 57 participants (7%) declared that they wouldn't use the app if it used their sensed information for purposes that weren't declared or intended by the app designer or provider.

6.3.2.4 Concerns regarding Unauthorized Access to sensitive information, phone sensors and services

19 of the 57 participants (33.3%) expressed concerns regarding the app having unauthorized access to sensitive information such as phone book or photo gallery or to intrusive sensors such as GPS, camera and microphone. Some of the comments include:

- *"I also don't want it to access my contacts and call or message them."*
- *"Whether it has access and saves my telephone number, password of accounts directly synced on my mobile like gmail, bank accounts etc"*
- *"I don't want an app listening/seeing things around me."*

6.3.2.5 Concerns regarding Sharing or storing of sensed information with a Third party or on a cloud/ server

16 of the 57 participants (28%) expressed concerns about their sensed information being shared with a third party, posted on a social network or stored in the cloud. Explicit comments include:

- *"The app should not send out any information like my location, my contacts, my passwords, etc. to any server."*
- *"Posting my data to Facebook or other social networking platform without my knowledge"*

- *"I'd want to know whether the data was transmitted to another machine for collection. I'd also like the ability to decline transmission of the data on a case-by-case basis (perhaps you're in a situation that you don't wish to be recorded)"*

- *"..if the app professed to save lives but also shared data collected in order to market ads to me, I wouldn't use it"*

6.3.2.6 Concerns regarding technical side effects

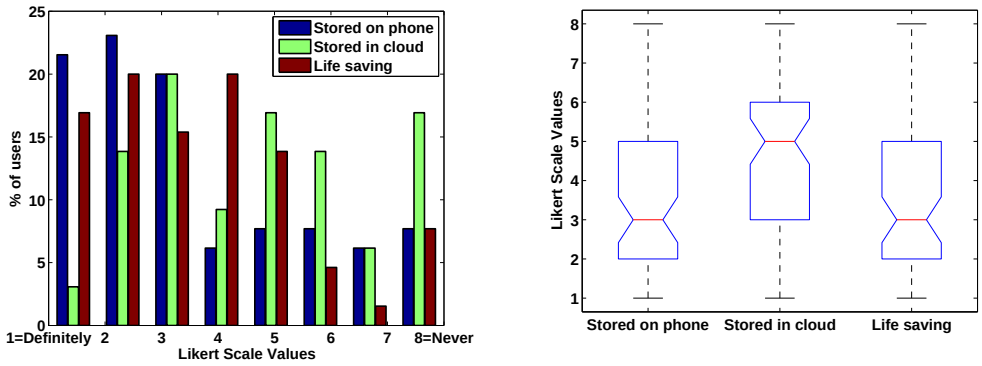
1 of the 57 participants (1.7%) said that the app should not slow down the phone's performance or reduce the battery life.

6.3.2.7 No privacy concerns

5 of the 57 participants (8.8%) said that they would have no privacy concerns and would use such apps only if they have very specific needs for them. One of these subjects stated that he shuts off all tracking sensors such as GPS and Wi-Fi as soon he leaves home.

6.3.3 Data control

As mentioned, most subjects expressed concerns regarding privacy and misuse of their sensed data. We then asked them if they would like to have more control of the data being sensed and if there is any data on their phone that they would limit and never allow an app to sense. 55 subjects responded to this question. 41 of the 55 participants (74.5%) wanted more control of their data and the ability to decide what should be monitored, where it should be stored, and how it should be used or mined. They also wanted the ability to



(a) Distribution of Responses to Sensitive Data Collection

(b) One way ANOVA of responses to Sensitive Data collection

Figure 6.3: Responses to Sensitive Data Collection Question (best viewed in color)

delete the data when they wanted to, and limit or disable data sensing. 6 of the 55 participants (10.9%) said they would not limit the sensing and monitoring and instead focus on limiting what *they* stored or used on the device. They would also like to see *"stringent enforcement against abuse of information"*. As one participant mentioned that *"if a device is capable of recording the data, I assume it will"*. 8 of the 55 participants (14.5%) said that it would depend on many factors such as the data being sensed or collected and what it was being used for.

These 41 participants, who said they wanted to limit or disallow sensing, specified several types of information that they would not allow an app to sense. The specified information can be categorized into the following categories (the % indicate the fraction of the 41 users who specified items of this category):

- Personal identification data such as name (16%) or location (9%)
- Private or sensitive data such as photos and music (22%)

- Browsing and search history, chats (18%) and emails (16%)
- Calendar, notes and contacts (22%)
- Apps being used (4%)
- Calls (4%) or text messages content (18%)
- Access to sensitive sensors such as camera or microphone (5%) or services such as Wi-Fi or 3G plan (2%)
- Social networks data and video history (6%)
- Financial information such as bank accounts, SSN, stored passwords and online purchases (13%)

6.3.4 Data Sharing

We now investigate the subjects' willingness to share their data. Figure 6.2(a) shows the responses of the subjects when asked if they were willing to use an app if it shared their sensed data (such as activity or location):

- With other users of the same app - The responses had $\mu = 5.54$ and $\sigma = 2.1$.
- With their friends on a social network - The responses had $\mu = 6.92$ and $\sigma = 1.36$.
- With another software or service for user modeling purposes - The responses had $\mu = 4.74$ and $\sigma = 1.96$.

As evident from Figure 6.2(a), 45% of the subjects would never allow posting of sensed data such as activity or location on a social network, 15% responded that they would

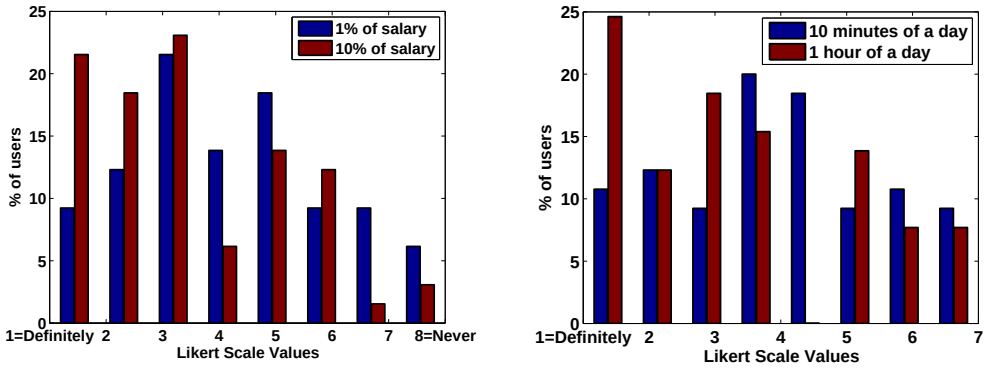
never share their sensed data with other users of the same app, and only 10% responded that they would never share their sensed data with another software for user modeling purposes. Thus, the subjects' willingness to share data with friends on a social network is lower than that for sharing data with other users of the same app or with another software.

6.3.5 Data Storage and Retention

Many of the subjects had expressed concerns regarding the sensed data being transported to another system or cloud over the network (see Section 6.3.2). However, for several context-aware and ubiquitous computing systems, a back end server side system is necessary. In such cases, the data transmitted over the network maybe either encrypted or anonymized but this can lead to a downgrade in performance.

Hence, we asked the subjects whether they would use such an app or system which transported and stored their data in an encrypted format in the cloud and made smart decisions for them. We also asked them if they would use the system if it anonymized the data but wasn't as effective as the system that only encrypted it.

Figure 6.2(b) shows the distribution of responses which are mildly positive. For a system that stored data in a cloud in an encrypted format but made smart decisions for the user, the subjects' responses had $\mu = 4.54$ and $\sigma = 1.96$. For a system that anonymized the data but wasn't as smart, the responses had $\mu = 4.55$ and $\sigma = 2.02$. Thus, the subjects were slightly more willing to risk the data being unanonymized if it improved the system's performance in making smart decisions for them though the difference is negligible.



(a) Saving money

(b) Saving time

Figure 6.4: Responses to Benefits to users questions (best viewed in color)

6.3.6 Sensitive Data Collection

We next investigated the users' willingness to use apps or systems that collected and analyzed sensitive data but with certain tradeoffs and benefits. For this purpose, we asked them to rate their willingness to use an app that sensed their health information such as heart rate, blood pressure, etc. (which is highly sensitive information) and stored it on their phone, in the cloud and in the cloud but only used it for the purpose of saving lives.

Figure 6.3(a) shows the distribution of responses. If the data was stored on the:

- Phone only - μ was 3.42 and σ was 2.23.
- Cloud - μ was 4.74 and σ was 2.13,
- Cloud but used for the purpose of saving lives - μ was 3.52 and σ was 2.0.

In addition, we also performed a one-way ANOVA (with significance level α as 0.05) on the subjects' responses. Figure 6.3(b) shows the box plots for the subjects' responses.

As evident from this distribution, the participants' willingness was higher when the data was stored on their phone. It decreased when the data was stored in the cloud but increased once again when it was mentioned that it will be stored in the cloud but used for saving lives. The mean values of the two box plots - for responses to health data being stored on the phone and responses to health data being on the cloud if it saved lives, are aligned very closely. This indicates that these distributions are not statistically different. Thus, if there is a tradeoff for the sensitive data to be utilized in a way that can prove beneficial, the users' willingness is higher and similar as compared to when the data is stored on their devices.

6.3.7 Benefits to users: Time=Money

As mentioned earlier, many subjects had concerns about their data being stored in the cloud or on a server but several context-aware and mobile systems require back end processing in order to be effective and efficient. We had investigate the subjects' willingness to use an app that stored their data in a cloud and made smart decisions for them in Section 6.3.5. However, in that case, the benefits were hypothetical. Here, we investigated their willingness to use SenseMe or a similar app if it stored data on a server or cloud and made smart decisions for users that had quantized benefits in terms of saving them time and money.

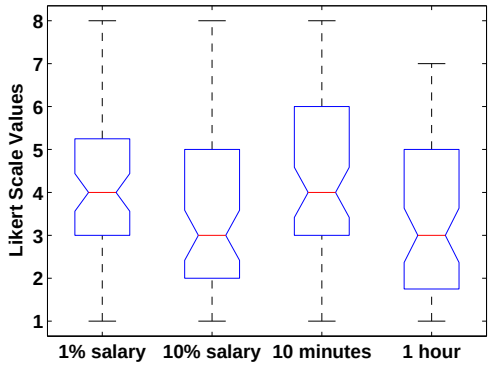


Figure 6.5: One way ANOVA of responses to User Benefits Questions

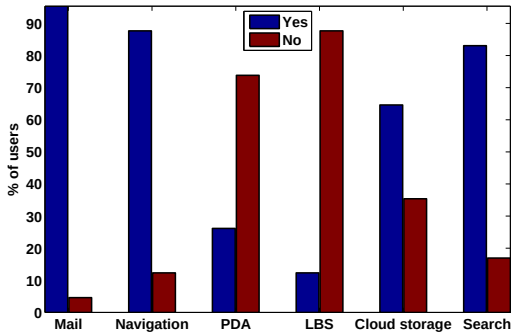
6.3.7.1 Saving money

Figure 6.4(a) shows the distribution of responses. When asked if the system saved them 1% of their salary, the responses had $\mu = 4.15$ and $\sigma = 1.99$. Intuitively, the willingness was higher ($\mu = 3.31$, $\sigma = 1.94$) when the amount of money saved was increased to 10% of their salary.

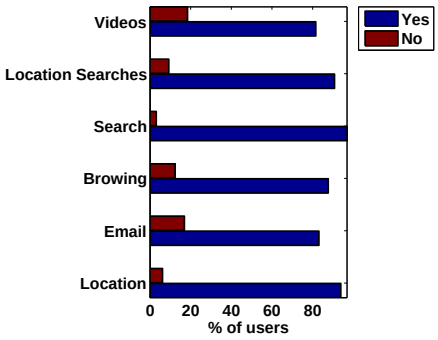
6.3.7.2 Saving time

Figure 6.4(b) shows the distribution of responses. If the system saved them 10 minutes of their time, the response had $\mu = 4.4$ and $\sigma = 2.11$. If the system saved an hour of their time, the willingness was higher with $\mu = 3.35$ and $\sigma = 1.92$.

Thus, as the hypothetical benefits increased in terms of time or money, user acceptance of storing sensed data on the cloud increased. In addition, we also we also performed a one-way ANOVA (with α level as 0.05) on the subjects' responses. Figure 6.5 shows the box plots for the subjects' responses. As shown, the mean values of the box plots for re-



(a) Usage of technologies and services provided by a major tech company



(b) Awareness about tracking and sensing by a major tech company through its various services

Figure 6.6: Responses to Brand Recognition and User Awareness Questions (best viewed in color)

sponses to saving 10 minutes and 1% salary are aligned very closely. Similarly, the mean values of the box plots for responses to saving 10% salary and 1 hour of time are aligned very closely. This indicates that these two response distributions are not statistically different. Thus, the subjects seem to view the benefits for time and money as equivalent and greater the benefit, the higher their willingness to use the app or system.

6.3.8 Brand Recognition and User Awareness

So far in the study, we had been referring to SenseMe or a hypothetical smart phone app or system. However, several technology companies are building such systems already or have deployed similar systems in the real world where they continuously monitor and sense their users. Therefore, we wanted to evaluate whether the subjects recognized this, were aware of the technologies and brands, and if they might vary their behavior based

on their awareness.

To this end, we investigated whether our study subjects recognized the brand of a major technology company and used various services (Mail, Navigation, Personal Digital Assistant (PDA), Location Based Services (LBS), Cloud storage and Web Search) provided by it. We also investigated if they were aware that their video history, location trace and location searches, and browsing and search history, were tracked by this company. In addition, we asked the subjects if they were aware that the company stores their data on servers and in the cloud.

Figure 6.6(a) shows the distribution of responses to the dichotomous questions related to usage of technologies and services provided by a major technology company. On an average, 61.54% of the subjects used at least one of the technologies or services. As shown, majority of the users were aware of and used the more popular services such as search, navigation, cloud storage and mail. The significantly lesser usage of the other two services - Personal Digital Assistant and Location Based Services could be because the former is available only on select devices while the latter is not so well known. Figure 6.6(b) shows the distribution of subjects' responses to being aware that the company tracked their video history, location and location searches etc. On an average, 88.97% of the subjects were aware of the tracking.

This awareness of how major brands track personal data and the widespread acceptance of services that use personal data seem to contradict earlier input from the subjects. All but one subject use the email service provided by the major tech company that we asked about in the survey. This email service tracks the social network that the emails themselves create, mines specific content in the email messages and uses it to for targeted

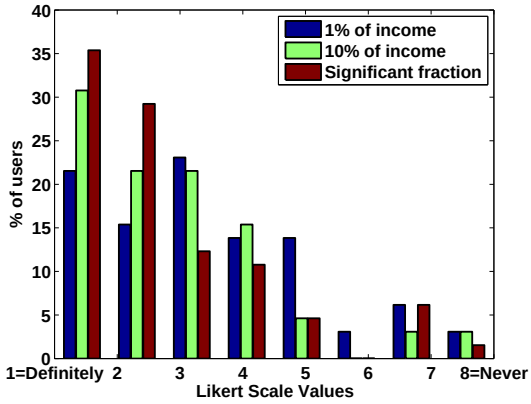


Figure 6.7: Usage of technologies and services provided by a major tech company despite being aware that users' data is sensed (best viewed in color)

advertising. Yet when we asked the subjects to freely list items a smart phone app should never sense, over 25% of respondents mentioned chats, text messages and emails. Additionally, many felt that using personal data for monetization purposes was wrong. This appears to show that users lower their convictions about private data with brands that they trust[4]. We will investigate this further in the future.

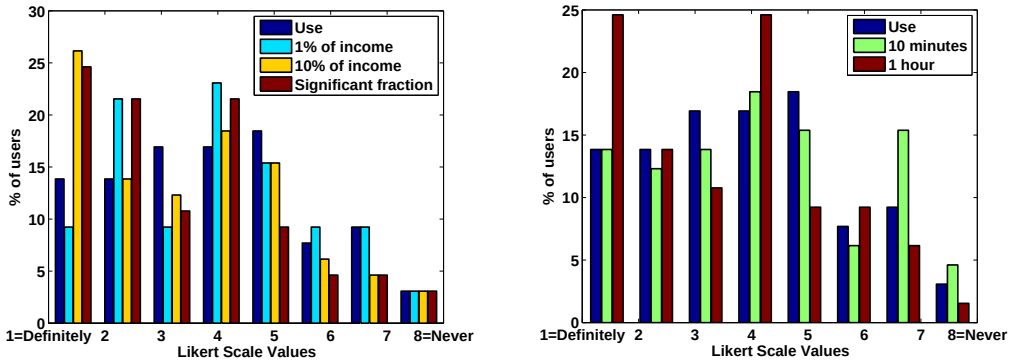
6.3.9 Brand Trust

6.3.9.1 Responses of all subjects

We then investigated whether the subjects would continue to use the services provided by the company despite being aware that their data was tracked along several dimensions. As an incentive, we again applied the hypothetical benefit of saving money, in order to see the trade-off between benefit and data sharing.

Figure 6.7 shows the distribution of responses. If the continued use saved the subjects:

[4] Although this compares an email service with a strict smart phone sensing app, email can also be a form of soft sensing.



(a) Usage of an app similar to SenseMe if it were developed by a major tech company and saved them money

(b) Usage of an app similar to SenseMe if it were developed by a major tech company and saved them time

Figure 6.8: Responses to Brand Reputation Question (best viewed in color)

- 1% of their salary - The distribution of responses had $\mu = 3.32$ and $\sigma = 1.92$.
- 10% of their salary - The distribution of responses had $\mu = 2.69$ and $\sigma = 1.73$.
- A significant fraction of the salary - The distribution of responses had $\mu = 2.52$ and $\sigma = 1.79$

Thus, the subjects were willing to use the technologies and services if the benefits were significant, despite being aware that their data was monitored. The % increase between the subjects' willingness to use these services if it saved them 10% of salary and if it saved them a significant fraction of it, is not very high. We believe that a possible reason for this could be that people are inclined to think that there is no concept of a 'free lunch'. Thus, if a service claims to save them a significant amount of money, they would actually be skeptical of it or mistrust it and may dismiss it as fraudulent.

We next attempted to draw comparisons between these responses and the subjects'

responses to the dichotomous Brand Recognition questions from Section 6.3.8 (regarding the usage of these technologies before being made aware that their data was tracked). To this end, we standardized the dichotomous results and Likert scale values to z-scores. Borenstein et al. [64] explain that it is possible to combine effect sizes from studies that used different metrics if there are comparable in relevant ways. Both the Brand Trust and Brand Recognition questions are same but the responses are on different scales. For the Brand Recognition question responses, we first converted the Yes and No responses for the 6 services to binary values, aggregated these binary values to convert them to a 6 point scale, and then converted the aggregated values to z-scores. For the Brand Trust questions, we directly converted the Likert scale response values to z-scores. The z-score is computed as: $z = \frac{x-\mu}{\sigma}$ where x is the raw value, μ is the population mean and σ is the population standard deviation. The distribution of the standardized responses for the usage of technologies:

- Before being made aware, had μ = 0.73 and σ = 0.67.
- Being aware of the tracking and if it saved subjects' 1% of their salary, had μ = 0.81 and σ = 0.57.
- Being aware of the tracking and if it saved subjects' 10% of their salary, had μ = 0.77 and σ = 0.63.
- Being aware of the tracking and if it saved subjects' a significant fraction of their salary, had μ = 0.77 and σ = 0.63.

Thus, when compared to their responses earlier, the willingness is lower but increases

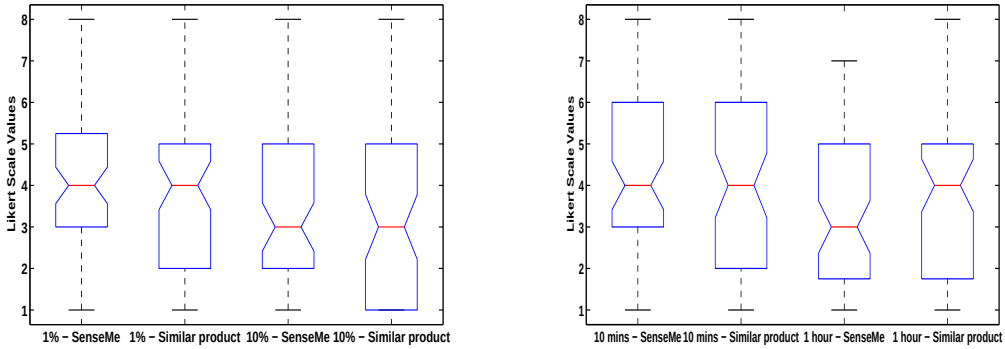
Saving	Group A	Group U
1 % salary	μ = 3.28 , σ = 1.95	μ = 3.55 , σ = 1.86
10 % salary	μ = 2.67 , σ = 1.77	μ = 2.82 , σ = 1.6
Significant fraction	μ = 2.36 , σ = 1.88	μ = 2.56 , σ = 1.29

Table 6.2: Relationship between participants' awareness status and responses to the Brand Trust questions

slightly as the benefit increases.

6.3.9.2 Relationship between participants' awareness status and responses

To further investigate whether the new awareness impacted their decision to continue the use of technologies provided by this company, we divided the participants into two groups. These groups represented the participants' awareness status and were based on the z scores computed from their dichotomous responses to the User Awareness questions. The groups were the A group which consisted of subjects who had been aware of the tracking before we informed them (z score $>$ 0.0) and the U group which consisted of the subjects who had been unaware of the tracking before we informed them (z score $<=$ 0.0). We then compared the Likert scale responses, of these groups, to the Brand Trust questions. Table 6.2 shows the distribution. Clearly, the U group is more reluctant than the A group to use these technologies after being made aware.



(a) Usage of SenseMe and usage of a similar app developed by a major tech company and both saved the subjects' 1% and 10% of salary

(b) Usage of SenseMe and usage of a similar app developed by a major tech company and both saved the subjects' 10 minutes and an hour

Figure 6.9: Anova for usage of SenseMe and a similar app developed by a major tech company and both saved the subjects' different amounts of money and time

6.3.10 Brand Reputation

We next analyzed whether the study subjects were willing to use an app similar to SenseMe if it were developed by the same major technology company and furthermore, had the same hypothetical benefits such as saving them time and money. Our intent was to determine if brand trust and reputation had a significant impact on their responses. Thus, our hypothesis was that since they already trusted the brand, their willingness to use the application and let it sense their data would be higher. As one subject from the SenseMe user study had mentioned in his exit interview, *"I would probably not use an app that monitors me through my smartphone, unless it's from a trusted source like Google."*

6.3.10.1 Responses of all subjects

Figure 6.8 shows the distribution of responses. We further compared the subjects' responses to this question with their responses earlier to the Benefits to Users (Section 6.3.7) questions where they were asked if they would use SenseMe or a similar app and share their sensed data if it saved them money and time. Figure 6.9 shows the results of one way ANOVA (with α = 0.05) on the subjects' responses with respect to both money and time. For:

- Saving Money - If SenseMe saved them 1% salary, the responses had μ = 4.15 and σ = 1.98 while for a similar product by a major technological company, the responses had μ = 3.94 and σ = 1.92. Similarly, if SenseMe saved them 10% salary, the responses had μ = 3.31 and σ = 1.94 while for a similar product, the responses had μ = 3.35 and σ = 1.99.

- Saving Time - If SenseMe saved them 10 minutes in a day, the responses had μ = 4.4 and σ = 2.1 while for a similar product by a major technological company, the responses had μ = 4.12 and σ = 2.1. Similarly, if SenseMe saved them 1 hour of a day, the responses had μ = 3.35 and σ = 1.92 while for a similar product, the responses had μ = 3.4 and σ = 1.96.

Thus, in both the cases, the willingness of all the subjects to use SenseMe was slightly lower than that for a similar app developed by the company mentioned. This seems to indicate that since the participants are not familiar with the reputation of the entity that developed the SenseMe app i.e. our research group, they are less willing to trust us. On

the other hand, the technology company mentioned is very well known and has a major brand so they are more willing to trust it.

6.3.10.2 Relationship between participants' group and responses

As mentioned in Section 6.2, the results come from two sources: the exit interviews of 15 subjects who participated in a live deployment of SenseMe, and web-based surveys filled by 55 participants. We next analyzed the differences in responses of these two groups - the L group which consists of subjects who participated in the live deployment and the O group which consists of the subjects who participated in the web-based surveys. Our intent was to reveal any influence that the first-hand usage of the app had on their responses.

A chi-squared test revealed that there was significant differences in the responses of the two groups. Table 6.3 shows the distribution. As shown by the μ and σ values for the Likert scale responses, the L group was more willing to use SenseMe than a similar app developed by a major technological company. On the other hand, the O group was more inclined to use a similar app developed by a major technological company rather than SenseMe. This difference could be attributed to the fact that the L group had participated in a live deployment and had first hand experience with SenseMe.

6.4 Limitations

In what follows we discuss some limitations of this work.

The Likert scale ratings and open-ended questions utilized in our studies are not ab-

Saving	Group L		Group O	
	SenseMe	Similar app	SenseMe	Similar app
1% salary	$\mu = 3, \sigma = 1.89$	$\mu = 3.9, \sigma = 1.66$	$\mu = 4.36, \sigma = 1.95$	$\mu = 3.95, \sigma = 1.98$
10% salary	$\mu = 2.4, \sigma = 1.58$	$\mu = 3.1, \sigma = 1.97$	$\mu = 3.47, \sigma = 1.96$	$\mu = 3.4, \sigma = 2.01$
10 minutes	$\mu = 3.2, \sigma = 1.81$	$\mu = 4, \sigma = 1.76$	$\mu = 4.62, \sigma = 2.1$	$\mu = 4.14, \sigma = 2.18$
1 hour	$\mu = 2.1, \sigma = 1.1$	$\mu = 2.9, \sigma = 1.73$	$\mu = 3.58, \sigma = 1.95$	$\mu = 3.49 , \sigma = 2.0$

Table 6.3: Relationship between participants' group and responses to the Brand Reputation questions

solute measures of user concern because our surveys explicitly asked respondents about privacy, trust and their willingness to use these apps. Surveys that directly ask questions about privacy may suffer from inflated user concerns about privacy [65] and therefore are not reliable measures of absolute levels of concern. We expect that this applies to our study as well.

Our survey questions compared users' responses to different mechanisms and alternatives for data sharing, data storage and retention, data collection, and benefits in terms of time and money etc. Where appropriate, we provided users with scenarios, which were meant to help them with assessment of possible risks. In our results, we weighed the risks and involved trade offs relative to each other. Thus, the same set of priming biases are applied equally to all of the alternatives presented in the surveys, so the priming effect should not influence the results.

We do not claim to predict the users' decisions when confronted with these risks and

trade-offs in real life because our study relies on self-reported data. As with the priming bias, we do not believe that self-reporting affects the validity of our results because this bias is equally present for all alternatives.

Our web-based survey did not reach professionals from various other backgrounds, who may have different concerns. However, it was taken by a large number of participants with varying ages, demographies and occupations. Secondary studies may be needed to target specific groups that could potentially have their own privacy and security concerns, such as doctors (who handle health records), lawyers (who handle client data), or company executives (who handle corporate data).

6.5 Design Guidelines

The aim of our study has been to explore privacy, trust, risks involved and other related issues in smart phone based personal sensing systems and applications. In this section, we propose some high-level guidelines derived from the results presented which we believe would be useful for personal sensing app or system designers in order to mitigate these concerns and outline some interesting directions for future research.

6.5.1 Maintaining Transparency

As evident from our results, many users are wary of apps that use their data for purposes other than what it declares. Also, they do not want apps to have unauthorized access to any data, sensors or services. Hence, its important that the app or system designer maintain transparency in the design and documentation of the app. They should clearly define what

the purpose of the app is, and what sensors, content and services it will access. Moreover, this information should be provided to the users at the time of app install or download and via a medium that would be most pervasive and accessible. As evident from the results, the EULA may not be the best medium since 50% of the users never read it. Thus, a medium other than the EULA (say, a dedicated pop-up message that catches the users' attention at the time the app is installed) may be more suitable for this purpose.

In addition, further research should be conducted to test the best formats and media to educate users about an app's use of private data. Previous research [66] has demonstrated that displaying required permissions and privacy information in a clearer fashion could play a more active role in influencing users to make privacy protecting decisions at the time of app selection. Lin et al. [67] found that telling users the purpose of an app's access to phone resources improved decisions and eased privacy concerns. Research has been carried out on online privacy notices formats for websites [68, 69]. Similar studies on privacy notices for mobile apps will be beneficial taking into consideration the mobile devices form factor, user attention span, user demographics etc.

6.5.2 Access to sensitive data or intrusive sensors

As evident from our results, if an app or system requires access to highly sensitive data such as photos, contacts or health data or to intrusive sensors such as GPS or microphone, many of the users are reluctant to use it unless they have very specific needs for it. There is a significant fraction of users (33%) who feel very strongly about this and would never use an app that accessed such data or sensors. Hence, it is imperative that the designers avoid

this and find other alternatives. For instance, rather than using raw GPS coordinates, it could be sufficient for an app to know which general geographic zone the user is currently in if accuracy is not a major concern.Or the designers could design the app to sense a user's location via network or cellular provider to localize the user at a city or locality level, instead of using GPS to localize the user to an exact location.

However, a significant number of popular smartphone apps such as Foursquare and Facebook make use of location. These popular commercial location sharing apps seem to mitigate users' privacy concerns by allowing them to selectively report their location using check-in functionalities instead of tracking them continuously and automatically. Hence, selective sharing of accurate location, sensed from GPS, based on the users' discretion is also a viable option.

6.5.3 Sharing of the sensed information

Sharing users' data with other applications, other users of the same app or with their friends on a social network should be done at the users' discretion. Thus, the designer should give the users' full control of what they want to share, when they want to share it and if they want to opt out of this facility. Alternatively, developers could provide annotations that reflect their privacy and data sharing policies, and this information could be incorporated into warnings or data access requests. Similar to tools like AppFence [70], that tell users whether their data is being sent to advertisers or other known third parties, other tools should be developed that can inform users if their data is being shared with other applications, other users or being posted publicly anywhere.

6.5.4 On-device vs on cloud or server

Since most users do not want their data to be stored in the cloud or on a third party server, designers of the system or app should consider the design in such a way that majority of the processing and storage is carried out on the user's device. The subjects' responses suggest that they would prefer that only a limited amount of data is transmitted over the network in a secure or encrypted manner and stored on a cloud or server. In addition, the system or app designers should clearly declare to users, what data is being processed on the device and what data will be transmitted to a server, and how it will increase the benefit or value of the system to the users. If an app is accessing and transmitting more data than it has declared in order to provide the required services, it can be easily discovered by users who use the app, and this could potentially be reflected in the app's reviews.

While Balebako et al. [71] have experimented with notifying the user and visualizing the amount and type of information being shared on his device in order to understand his perceived concerns, they do not comment on whether indicating the benefit or value of the shared information would influence the users' decision to share it.

6.5.5 Data for Benefits

As evident from our study, users are more willing to share their data if the benefits to them are significant, such as saving them time or money, or if the app provides them with services such as timely and relevant information. Though our study does not determine what payoff would result in an optimal number of users sharing private information, it is evident from the results that most users place a value on their private information which

dictates how and under what circumstances they would surrender it. Thus, the app or system designer should balance privacy invasion with the benefits to users in a way to maximize user participation. Staiano et al. [61] have investigated the monetary value that people assign to different kinds of personal information as collected by their mobile phone, including location and communication information.

6.5.6 Usage of the sensed information

As opposed to commercial purposes and targeted advertising, if the users' sensed information is utilized for a beneficial and effective cause such as saving lives, users' are more willing to share it. Yet this increased willingness to share sensitive data is not without reservation, as evident by the fact that more than 25% of the subjects were unwilling to share such data even if the benefits of sharing data can result in saving lives. This may be attributed to the negative connotation that surrounds usage of such data for commercial and advertising means. One subject opined that the only use of sensitive data by a sensing party would be to capitalize on it for targeted advertising. Another stated that he was skeptical of the inability of companies to not monetize on the collected data, making him very unwilling to share sensitive data even if to save lives. Nonetheless, it is essential that the app designer clearly state their intentions for the usage of the sensed data.

6.5.7 Build trust and reputation

As we observed, users are more willing to trust a known brand or company with their data as opposed to an unknown entity. Hence, it is important to establish a reputation

and gain their trust. As evident from our results, subjects were more likely to use an app created by a well known brand then a smaller, obscure company. Yet if they have first hand experience with the app, they are more willing to use it. Additionally, data they claim they would never share with a cell phone application do not align with what they currently share with major email implementations. All but two respondents use an email service provided by a well known brand, that uses email content and contact listings, in order to enrich the email experience; something that one-fourth of the users claimed they would never allow on a cell phone app.

Moreover, we believe that users' reluctance to share their data, even if to save others' lives, also relates to trust. This reluctance of an individual to provide resources that can save another's life and require no effort on his/her part, should be startling to app designers. This should be a testament to both the lack of trust and the importance to regaining this trust.

Hence, one of the ways via which designers can build trust is to perform live deployment of their systems and apps in the wild. This could be conducted with the aid of professional market research companies which recruit users from different demographics. The designers can conduct these deployments in-lab and *in situ* with follow-on user feedback surveys. This would allow the users to gain first hand experience with their apps in a practical real life scenario and allow the designers to get valuable feedback on their apps.

6.6 Related Work

Since we explore a multitude of privacy, trust, risks involved and other related concerns with personal sensing systems and applications, we have divided the related work into various sections and differentiate our work from them. /moreover, as stated before, none of the existing works have addressed such a broad spectrum of concerns with personal sensing applications.

6.6.1 Privacy concerns with personal sensing via proprietary devices

Klasnja et. al. [47] explored privacy concerns with personal sensing in a field trial of the UbiFit system [72]. Unlike SenseMe, which runs on the user's smartphone, the sensing in UbiFit was carried out using proprietary hardware and hence, wasn't privy to personal information. For this reason, our user study focused more on what was being sensed and inferred as opposed to the sensors being used. Also, UbiFit only recognized physical activities while SenseMe performs temporal context and activity recognition along several dimensions. Moreover, we investigate several factors such as brand trust, recognition and awareness in addition to just sensors. However, our results support their arguments on Data Retention and Perceived Value of the applications.

6.6.2 Privacy concerns with location tracking and sharing

Mobile privacy research [73,74] has traditionally focused on location tracking and sharing and has examined users' privacy concerns about sharing mobile location data. Iachello and Abowd [75] described how to build appropriate privacy controls into a social location-

sharing application.

Numerous studies [76–78] have explored location sharing behavior of users and the factors that influence it. Their findings suggest that who is requesting the user's location, why they are requesting it and to what level of granularity significantly affects the user's decision to share it. Lederer et al. [79] found that the identity of the location requester matters more than the place in a user's willingness to share his or her location, and Anthony et al [80] focused on the effect of the specific place that the user is asked to share. Moreover, it seems that the users' age, gender, mobility, and geographic region also play a role in location sharing behavior. While we asked users in general about sharing various forms of data in addition to location (such as activities), similar factors may influence users' decisions. In addition, we questioned participants about their privacy concerns related to all types of fine-grained personal data from their smartphone. Hence, it is difficult to directly compare our work with them.

6.6.3 Other smartphone privacy concerns

Smartphone apps have the ability to access a number of resources beyond location data. Smartphone APIs let applications read many types of data (e.g., photographs) and make changes to the phone (e.g., delete data). Few studies have explored the space of smartphone privacy and security beyond location.

Lane et. al. [81] discuss the issue of privacy in their survey of smart phone sensing applications and systems. However, they do not present any evaluations or quantifiable results. Instead they draw conclusions from existing work in smart phone sensing.

Muslukhov et al. [82] asked 22 smartphone users about the value and sensitivity they assigned to eleven types of data (SMS messages, photos, contacts, emails etc.) on their phones. One aspect of our study, where we asked users about Data Control (see Section 6.3.3) and list the types of information for which they would limit or disallow sensing, shares similar goals with this work.

Felt et al. [83] report that users' concerns about data sharing depend on who the data is being shared with. Their findings suggest that for different data types, publicly sharing the data most concerning than sending the data to a server. Sharing with friends and advertisers rank in the middle, between public sharing and sending the data to a server. While we asked users about data sharing as well (Section 6.3.4), the sharing mechanisms that we suggested were different.

Staiano et. al. [61] performed auctions of users' data and found an optimal price for which users would sell it. This auction however released the data in general and not to a commercial entity looking for ways to capitalize on the data and target the users. Our results show that users are not uniform in their sharing behavior or how they feel their data should be used. Thus varying the consumer of the data might greatly influence a user's ability to share it.

Chapter 7: TellMe: Bootstrapped Discovery and Ranking of Relevant Services and Information in Context-aware Systems

A context-aware system uses context to provide relevant information and services to the user, where relevancy depends on the user's situation. This relevant information could include a wide range of heterogeneous content. Many existing context-aware systems determine this information based on pre-defined ontologies or rules. In addition, they rely on users' context history to filter it. Moreover, they often provide domain-specific information. Such systems are not applicable to a large and varied set of user situations and information needs, and may suffer from cold start for new users. We address these limitations and propose a novel, general and flexible approach for bootstrapped discovery and ranking of heterogeneous relevant services and information in context-aware systems. In this chapter, we present the design and implementation of TellMe [10] - a novel, general and flexible framework (integrated with the Rover II context-aware middleware) for bootstrapped discovery and ranking of heterogeneous relevant services and information in context-aware systems.

7.1 Introduction

A context-aware system uses context to provide relevant information and services to the user, where relevancy depends on the user's task or situation [1]. This involves delivering "the right information at the right place and the right time" i.e. *relevant*, *personalized* and *timely* information to users. Thus, a context-aware system should exhibit the following three capabilities:

- Determining relevant information - This capability is the cornerstone of context-aware computing. In today's world, with the abundance of information available to us, information overload can easily happen. Hence, it is imperative that the system retrieves and displays only that information which is relevant to the user's task at hand.

- Personalization - This is achieved by acquiring a user's context (needs, preferences, etc.) through implicit or explicit means and using it to filter the relevant information.

- Timeliness - The system can achieve timely information delivery by providing the personalized and relevant information to the user at a time when he needs it and can act upon it.

However, what constitutes relevant information to an individual user may vary widely according to his tasks or situations. For instance, a user who intends to get a medical test done would benefit from relevant information, such as recommendations for hospitals and laboratories, retrieved from the appropriate sources. A user who is about to leave

for work may be interested in the weather forecast and traffic enroute to her workplace. Furthermore, this information should be personalized according to the users' context such as location, preferences and needs. Finally, the context-aware system should provide this information to the users in a timely manner in order to help them make an informed decision regarding where they should go, which routes they should take and when they should leave so that they can reach their destination on time, thus, saving their time and effort. Ultimately, this enhances their efficiency and facilitates effective decision making.

As evident from this discussion, a user's intended events, activities, tasks and situations (henceforth, collectively referred to as *Situations*) are the most crucial factors in determining what information is relevant to him. It is also evident that a user's information needs in the real world vary according to his situations and could include a wide range of heterogeneous content, such as weather, news, traffic information etc. This presents two challenges: recognizing a user's situations and tasks, and determining information relevant to them. In this chapter, we focus on these two challenges.

In many existing context-aware systems and applications such as Siri[1], the user *explicitly* provides his situation or task information and requests for relevant information. For instance, if he is going for lunch, he will request information about restaurants or food options nearby. However, for widespread adoption, it is essential that his situations be detected in an automated and unobtrusive manner. An existing way to recognize a user's current situation is via activity recognition (as done in [9, 46]) though it can recognize only a limited number of diurnal situations (such as 'Walking', 'Driving' etc.). On the other hand, user generated content from scheduling resources (such as calendars,

[1] www.apple.com/ios/siri/

reminders or to-do lists), which users employ prolifically nowadays, can provide a rich platform for *implicit* recognition of everyday situations and tasks. This content can be easily used to augment activity recognition.

Thus, to address the first challenge, we propose to glean as much information, about the user's tasks and situations, as we can by *soft* sensing of these user generated sources on smartphones and desktops through application access or content extraction. For instance, consider a user who has a scheduled event, 'Lunch', marked in his calendar. A context-aware system can now infer, from this event, that the user's situation would be 'Having Lunch' in the near future.

The second key challenge is to determine what information is relevant to a user's situation. As mentioned earlier, the relevant information varies widely with the user's situations and could include a range of heterogeneous content. Clearly, it would be infeasible to build an individual system to provide each type of content - a single generic and unified system that could integrate and retrieve a variety of such heterogeneous information from various sources, based on its relevance to the user's situation, would be the ideal solution.

Some of this relevant information can be extracted from *internal* sources such as user's emails. However, most of it needs to be obtained from *external* sources. There are several web-based services and websites (henceforth, referred to as *Information services* or just *services*) that complement web search results. They provide domain-specific information (e.g. traffic, weather etc.) aggregated from expert and user contributions and can serve as an excellent source of such content. The question that can be posed now is - how to determine the most relevant services, and the most relevant information to be retrieved

from them, for a given user situation?

Mostefaoui et al. [84] identify that predicting what information would be relevant to a user is challenging. They state that it is possible to determine it using techniques that are based on users' feedback or the system developer's observations. Thus, in order to automate the process of discovery and retrieval of relevant information in a context-aware system, the system designer could encode or program rules by hand. Consider the example user situations mentioned earlier - 'Getting a medical test done' and 'Going for lunch'. For these, a system designer could encode rules such as:

- If a user's situation is 'Get medical test done' then retrieve information about 'Hospitals' or 'Medical laboratories',
- If a user's situation is 'Lunch' then retrieve information about 'Restaurants' or 'Food' near the user.

However, this approach suffers from several drawbacks:

1. These static rules are expensive to generate and maintain. In addition, they cannot be dynamically adjusted in response to changes in a user's behavior.
2. A user may be involved in innumerable situations and, hence, it is not possible to enumerate such rules for every situation.
3. Furthermore, these rules, and systems that employ them, can only cover the set of possibilities that system designers anticipated and will not scale to a large number of unanticipated situations and information needs of different users.

Xu et al. [85] proposed an OLAP based approach for mining user interaction logs in order to filter information based on its relevance to the user's context. For instance, if a user has accessed her shopping list most frequently in a certain context (location, day of week and time), then it implies that this is the most useful information to her for that context. Even though user interaction logs are a valuable source for determining relevant information, a major limitation of this approach is that for new users (for whom the system has little or no context history), the system may suffer from a *cold start* and may not be able to determine any relevant information. Thus, new techniques need to be developed in order to augment this and *bootstrap* the context-aware system so that it can retrieve information for unanticipated user situations or for new users and avoid a cold start. Yet, enough attention has not been given to this problem.

Thus, to address the second challenge and all these limitations, we propose a novel, flexible and general approach based on Semantic Relatedness [86] - a metric for determining similarity of two documents or phrases based on their semantic meaning. It is normalized to a value between 0 (no relatedness) and 1 (very high relatedness) and is significantly more powerful than simple keyword based matching. This metric has been steadily gaining attention among Natural Language Processing (NLP) researchers and has been used in several applications such as targeted advertising [87] and web search [88] with positive and beneficial results.

We propose to utilize this metric as a measure of relevance of the information, provided by a service, to the user's situation or task. For instance, the semantic relatedness between 'lunch' and 'food' is 0.76. Clearly, services that provide information or recommendations for food are relevant to a user's situation 'Going for lunch'. This demonstrates

that this metric can be effectively used for discovering and ranking information relevant to users' situations. Ultimately, this creates a context-aware system that is flexible, generic, easier to maintain and can retrieve information beyond what could be anticipated by a system designer. Moreover, such a system does not rely on hand coded rules and will be able to provide relevant information for new situations and to new users (for whom there is no past interaction or usage history), thereby, avoiding a cold start.

We implement both our proposed ideas in a single generic system which employs various algorithms to discover and rank candidate services relevant to a user's situation. It retrieves the relevant information from the ranked services, aggregates it and presents it to the user. Our contributions are:

- We propose and implement the idea of inferring a user's situations via soft sensing of user generated content from sources such as calendars etc.

- We address the problem of bootstrapped discovery and ranking of heterogeneous services and information, relevant to a user's situations, in context-aware systems. We propose a novel approach, based on Semantic Relatedness, to solve it.

- We design and implement four variations of a base algorithm that ranks candidate relevant services, and the information to be retrieved from them, based on the semantic relatedness between the information provided by the services and the user's situation description.

- We implement these algorithms as part of a system, called TellMe, and conduct a live deployment and a web-based study with 14 subjects to evaluate their efficacy. We demonstrate that they show strong positive correlation with human supplied

relevance rankings and can be used as an effective means to discover and rank relevant services and information.

- We also show that our approach is general, applicable to a wide set of users situations and can provide relevant information to new users without requiring any user interaction history, thus, avoiding a cold start.

7.2 Related Work

Since our work spans several ideas, we have organized the related work into several different subsections. We highlight their shortcomings as well as differences with our approach.

7.2.1 Use of Software Sensors

Garlan et al. [89] propose utilizing location information from a user's calendar, in the Aura system, in order to predict future locations of a user and take appropriate actions on his behalf based on this information. On the other hand, we propose a generalized approach that utilizes user generated content from calendars, reminders and to-do lists to infer the tasks and situations of a user.

7.2.2 Service Discovery and Selection in Ubiquitous Computing Systems

Service discovery refers to a mechanism that allows users to locate services on-demand and in reasonable time [90]. Existing works [91–93] perform context-aware service discovery and selection based on how closely a user's request and the service description

matches. The matching is usually done using a pre-defined ontology. Most of these works are constrained in their approach as they consider services to be either hardware or network resources such as printers or projectors. In addition, a major drawback of using pre-defined ontologies is that they are usually domain specific and limited to a finite number of concepts that the ontology designer has taken into account. Even though our approach shares similar goals with these works, it stands out in several ways:

- We incorporate a variety of services that offer heterogeneous content instead of focusing on just hardware resources.
- We utilize state of the art NLP techniques to calculate relevance using content from any of the services present in a context-aware system.
- We do not limit ourselves to a pre-defined ontology. We employ powerful, domain-independent and exhaustive databases and repositories such as Wikipedia, Wordnet and other corpora. The use of these repositories ensures that a large amount of world knowledge is exploited for determining relevance instead of relying only on the system designers' knowledge.

7.2.3 Smartphone application usage prediction

Predicting smartphone application (referred to as an 'app') usage has received significant attention recently. Several works such as Huang et al. [94] and Shin et al. [95] focus on predicting the next smartphone app that a user would use based on contextual information such as time, location, or usage information such as most frequently or recently used app. In contrast, we focus on determining the information and its source, from among several

heterogeneous sources, that would prove most relevant to the user's situation irrespective of the source type - whether its an app or a service. Moreover, we do not rely on usage information in our current system as our main goal is to demonstrate the feasibility of our proposed approach for bootstrapped discovery of relevant information.

7.2.4 Relevant information discovery in context-aware systems

In most of the notable context-aware systems such as Context Toolkit [96], Context Broker Architecture [97], and Gaia [98], determining relevant services and information based on a user's situation or task is either not addressed or is achieved using pre-defined rules. As explained earlier, such static pre-defined rules are not suitable for scaling and adapting to a dynamic environment. They will need to be adjusted according to changes in the user's behavior or situation, or due to new unanticipated situations.

A number of existing systems [4, 5, 99] focus on determining relevance of information based on the user's interests and preferences or their context such as time and location. Furthermore, most of these systems have narrow applicability and are specific to domains such as meeting room environments [97], museums [100], airports [99], tour guides [4, 101–103] and healthcare [5]. Although these systems serve their intended purpose well, they are limited in scope and can only handle information specific to their particular domain rather than the heterogeneous content that users require and that could help them in effective decision making. In contrast, we do not focus on a specific application or domain. Instead, we develop a generic infrastructure for aggregation of a variety of content in order to satisfy a diverse set of user information needs.

7.2.5 Relevant information discovery in commercial systems

Tempo[2], a smart calendar iPhone app, uses calendar data to provide users with relevant internal information such as related emails and documents, before an appointment is due to happen. It also determines navigation directions to locations mentioned in the user's upcoming calendar entries. However, to the best of our knowledge, it does not retrieve any other relevant information from external sources that could prove helpful to the user's situation.

Google Now[3] is a smartphone app that provides users with information such as flight schedules, weather, and traffic in the form of 'Google Now Cards'. These cards are often displayed statically based on user's context such as location (for instance, the 'Weather', 'Photography Spots nearby' or 'Events nearby' cards). They can also be generated dynamically based on the user's recent search history and emails. Even though internal information (such as that from email) can be valuable to a user, it has limited benefits if he wants to discover new information serendipitously. For instance, a user heading to the airport may benefit significantly from external information (such as transport options to the airport) in addition to internal information (such as flight schedules) from his email. More importantly, displaying relevant information based only on location can prove to be ineffective. As mentioned earlier, a user who needs to get a medical test done would be better assisted by recommendations for hospitals rather than spots for photography nearby. In addition, even though email can be a rich source of situations, we believe that it is constrained for recognizing everyday situations and tasks. Other user generated

[2] `http://tempo.ai/` [3] `www.google.com/landing/now/`

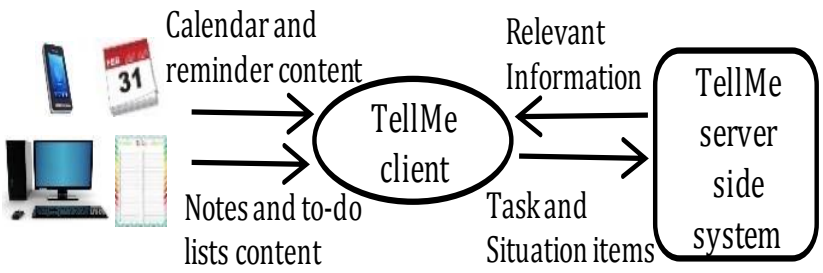


Figure 7.1: Overview of the TellMe system

sources such as calendars and reminders can serve as a more abundant source for them and can be easily mined from a smartphone or a desktop.

To address all these limitations, we propose to discover and rank a wide variety of external heterogeneous services and information relevant to a user's situation (sensed from user generated sources) to enhance his efficiency and quality of life.

7.3 The TellMe System

In this section, we describe the TellMe system, its components and its underlying algorithms in detail.

7.3.1 TellMe client side system

Figure 7.1 shows an overview of how the TellMe system functions. A TellMe client application running on a smartphone or desktop aggregates situations and tasks information from several user generated information sources such as calendars, reminders, and notes. These situation and task *items* are sent to the TellMe server side system. The client ap-

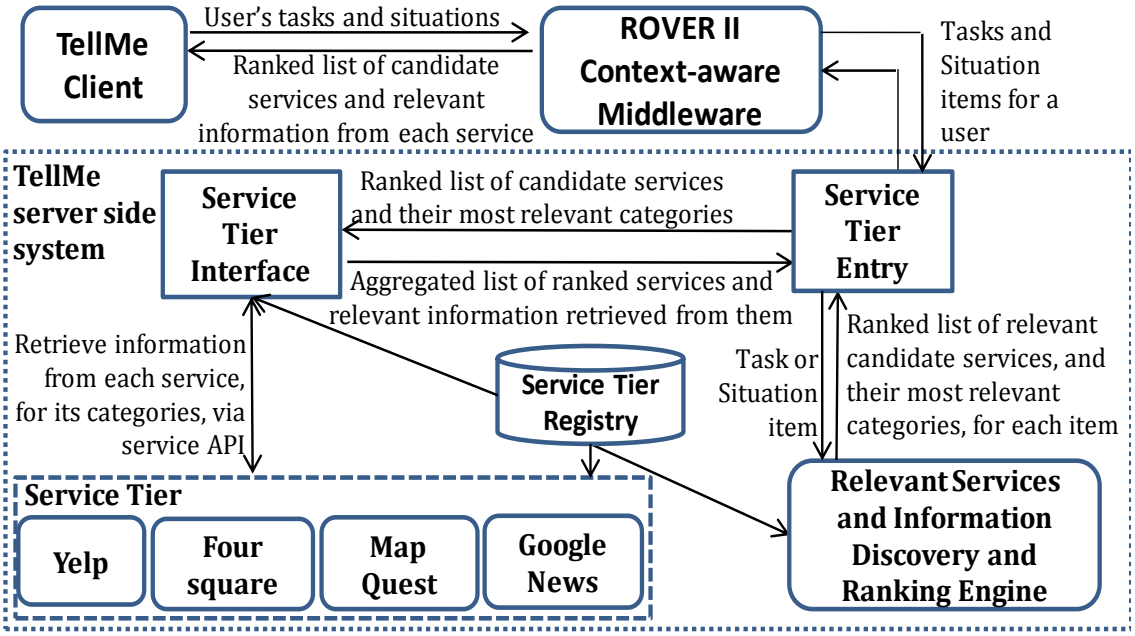


Figure 7.2: Architecture of the TellMe server side system

plication receives the relevant information for each item from the server side system and displays it to the user.

For prototyping, we have implemented a TellMe client application as a Google Calendar Event Gadget [104]. Users have to import this gadget in their existing Google calendars via a browser and authorize it to access their calendar entries and current geolocation. The gadget appears as an icon on top of each day in the calendar. On clicking the icon, a user can see a list of heterogeneous services and information relevant to each of his calendar entries for that day embedded as a web page in the gadget (see Figure 7.4).

Please note that though the current client prototype focuses on calendar entries, it can be easily extended to include situation information from other sources such as reminders and to-do lists that may reside on a user's phone. To this end, we are also currently developing a smartphone based TellMe client application.

Service	Top level Information Categories (# of subcategories)
Yelp	Active Life (74), Arts and Entertainment (20), ...
Google News	Sports (26), Entertainment (7), ...
FourSquare	Arts and Entertainment (52), College and University (38), ...
Angie's List	Home, Auto, Wedding, Parties and Entertainment ...
MapQuest	Navigation, Maps, Route, Traffic, Incidents, ...
LastFm	Music, Music album, Music artist, Music chart, ...
Weather Underground	Weather, Environment, Outlook, Temperature, ...

Table 7.1: Services registered in the Service Tier and their top level information categories

Buy Groceries
Drop clothes for dry cleaning
Clean apartment
Laundry
Pay utility bill
Go to Gym
Mow the lawn

Table 7.2: The top 10 most commonly recurring calendar entries, to-do and reminder items collected in our user study

7.3.2 TellMe Server side system

Figure 7.2 shows the architecture of the TellMe server side system. It is integrated with the Rover II context-aware middleware, introduced in Bhargava et al. [16], which can store and retrieve relevant contextual information and learn user behavior models. The server side system has the following components:

7.3.2.1 Service Tier

This component registers several external web-based information services, from which relevant information can be retrieved, and consists of client stubs to them. It also provides an API to add or register more services. For our initial prototype, we have selected 8 services based on the wide spectrum of information content that they provide, their popularity and ease of API availability, documentation and usage. These are:

- MapQuest[4] - a navigation service used for obtaining directions to a destination, traffic information etc.
- WeatherUnderground[5] - a service typically used for obtaining information on current weather conditions.
- LastFm[6] - a popular music discovery service.
- Google News[7] - a news service that provides comprehensive news coverage aggregated from sources all over the world.

[4] `www.mapquestapi.com/traffic/` [5] `http://api.wunderground.com/`

[6] `www.last.fm/` [7] `www.news.google.com/`

- Yelp[8] - a service for user reviews of local businesses.
- Google Feed[9] - a service for subscribing to web feeds or blogs such as RSS or media feeds from any website.
- Foursquare[10] - a location-based social networking service to search for places and venues around the user's location.
- Angie's list[11] - a service that aggregates verified consumer reviews of service companies in USA.

7.3.2.2 Service Tier Registry

This component contains a list of all the services, that are registered in the Service Tier, and the categories of information they provide. Since the information provided by a service can be described succinctly and coherently in terms of its information categories, we use them to compute the semantic relatedness metric.

In the current version of our system, all the 8 services mentioned earlier are listed in this Registry. Some of these services and their top level categories are shown in Table 7.1. These categories have been retrieved from the websites of these services or generated manually based on their content. Many of these services such as Yelp and Foursquare have a hierarchical category structure where general categories subsume more specific ones. It is evident that some of these services provide similar or overlapping information.

[8] `http://www.yelp.com/` [9] `https://developers.google.com/feed/`

[10] `www.foursquare.com/` [11] `www.angieslist.com/`

7.3.2.3 Service Tier Entry

This component functions as a gateway or an entry point to the TellMe server side system. It receives the list of user's situation items from the Rover II middleware, parses it and sends each item to the Relevant Services and Information Discovery and Ranking Engine. It also receives the ranked list of candidate relevant services and their most relevant categories, for each item, from the engine and sends it to the Service Tier Interface.

7.3.2.4 Service Tier Interface

This component is responsible for interfacing with the services registered in the Service Tier and aggregating the relevant information retrieved from them. It accesses each ranked candidate service, via its public API, in order to retrieve the information for its most relevant categories and personalizes it based on the user's context (such as location or preferences) if available. It aggregates this information and sends it back to the Service Tier Entry.

7.3.2.5 Relevant Services and Information Discovery and Ranking Engine

This is the core component for determining information relevant to a user's situation or task item. Figure 7.3 shows the pipeline for it. It employs four variations of the base Relevant Services and Information Discovery and Ranking algorithm (Algorithm 3). This algorithm processes the situation/task item description, and utilizes the metric of semantic

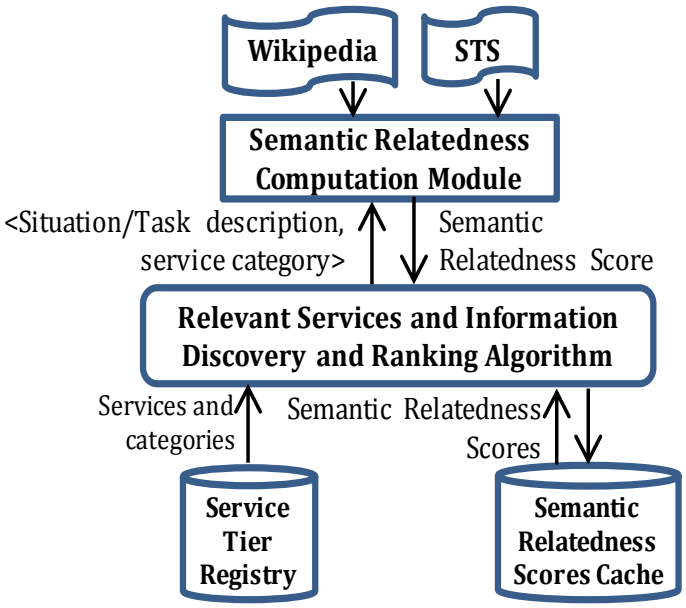


Figure 7.3: Pipeline for the Relevant Services and Information Discovery and Ranking Engine

relatedness and a ranking mechanism to discover and generate a ranked list of candidate relevant services (from among those registered in the Service Tier) for it. For each ranked service, it also determines the three most relevant information categories. Currently, we focus only three relevant categories because the relevant information should be available to the user within 2 or 3 interactions (swipes/clicks) with the client application.

As shown, the algorithm iterates over each service listed in the 'Service Tier Registry' and employs the Semantic Relatedness (SR) Computation Module to calculate the SR score between each of the service categories and the item description. Once the scores have been computed for all the categories of each service, a 'Service Score' is calculated for it by aggregating the scores from one or all of its categories. The services are then ranked based on the service score. For each ranked service, we also determine the top 3 categories which have the highest SR scores. These are the three most relevant categories

for that service. The SR scores are further cached after retrieval to improve response time on subsequent calls.

We implemented four variations of Algorithm 3 by employing two different methods for calculating SR scores and two different ranking mechanisms. The methods for computing SR scores are:

- STS-based SR - This variation employs the Semantic Textual Similarity (STS) System [105] for computing SR. The STS system is based on Latent Semantic Analysis (LSA) along with WordNet knowledge and is trained on LDC Gigawords and Stanford Webbase corpora. Since the STS system incorporates lemmatization, POS tagging and parsing as part of the SR computation, we do not perform these operations on the category and situation phrases.

- Wikipedia-based SR - This variation is a novel approach similar to Explicit Semantic Analysis [106]. It uses the cross-lingual dictionary created by Spitkovsky and Chang [107] along with the lch calculation from Rada et al. [108] and is further evaluated against Wikipedia by WikiRelate [109]. It is described in Section 7.3.4 for a more lucid explanation.

The service score is computed using the following two different mechanisms and is then used to rank the services:

- Highest Semantic Relatedness (HSR) - The service score is the highest SR score for any category of the service.

- Average Semantic Relatedness (ASR) - The service score is the average SR score over all the categories of the service.

Algorithm 3: Base Relevant Services and Information Discovery and Ranking algorithm (The algorithm can vary depending on the method of calculating semantic relatedness and the mechanisms for ranking.)

Input: Situation or Task item description, List of Services and their categories from the Service Tier Registry

Output: Ranked list of candidate relevant services and three most relevant categories for each service

foreach *Service listed in the Service Tier Registry* **do**

  foreach *Category of the Service* **do**

    Calculate the Semantic Relatedness (SR) Score between the Category and the situation/task item description;

    Store the Category with Highest Semantic Relatedness (HSR) Score;

  end

  Calculate the Average Semantic Relatedness (ASR) for the service over all categories;

  Calculate the $ServiceScore$ from the Category Scores using HSR or ASR;

  If $ServiceScore < Score_{threshold}$, replace $ServiceScore$ with 0.0;

end

Rank the services in decreasing order of $ServiceScore$;

For all services that have $ServiceScore$ = 0.0, set rank as 'Not Applicable';

For all ranked services, rank their categories in decreasing order of SR scores ;

return *Ranked list of candidate services and the top three categories with the highest SR scores for each service;*

If the service score is below a threshold, it implies that the information provided by the service is not relevant to the given user situation and hence, it is not ranked. This helps in reducing noise and false positives. Since SR is a cosine similarity measure, a threshold of 0.293 (1 - cos 45°) is generally considered an appropriate threshold and we use that in our implementation.

This ranked list of candidate services, along with their three most relevant categories, is then propagated to the Service Tier Interface component (via the 'Service Tier Entry'). The Interface retrieves the information for each of the 3 categories, for each service, via the service API. It further filters this information based on the user's context, such as location or preferences, if available. This ranked list of services, and the information retrieved from them, is then sent to the TellMe client application which displays it to the user.

7.3.3 Illustrative Use Case

We now illustrate the utility and benefits of the TellMe system via 2 scenarios (including the one described earlier in Section 7.1).

7.3.3.1 Use Case 1

For the user's task 'Get medical tests done', a generated list of discovered and ranked candidate relevant services and their categories by the STS-HSR variation of Algorithm 3 is:

1. 'Laboratory testing', 'Health and medical', and 'Medical centers' from Yelp

Algorithm 4: Algorithm to compute semantic relatedness using Wikipedia-derived data.

Input: A pair of strings

Output: A measure of semantic relatedness.

```
// Calculate the most related concepts from each
   string:
```

foreach *Input String* **do**

> Tokenize the string into unigrams and bigrams;
>
> Map the unigrams and bigrams to associated concepts from the substring dictionary;
>
> Keep the combination of tokens that maximizes the sum of the pairwise lch scores;

end

```
// Find the most related concepts from each
   combination:
```

foreach *Combination of Concepts* **do**

> Calculate lch between the two concepts;
>
> Maintain the concept pair with the highest lch value;

end

return *Largest* lch *value and concept pair;*

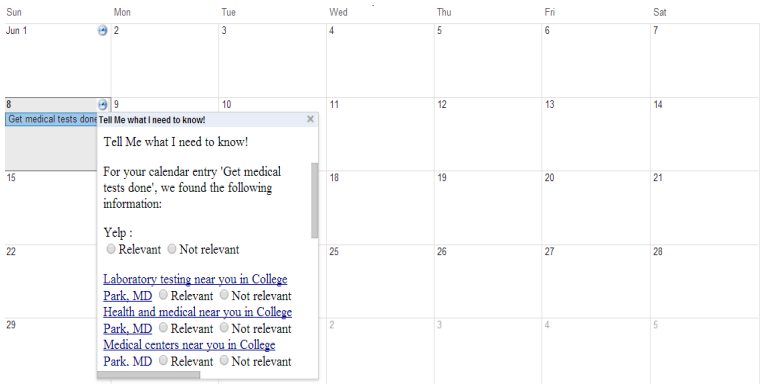


Figure 7.4: Screenshot of the TellMe client application displaying relevant information for a user's task

2. 'Doctor's office', 'Medical school', and 'Hospital' from FourSquare

3. 'Genetic medicine', 'Physical medicine', and 'Alternative medicine' from Angie's List

4. 'Health', 'Science' and 'Legal' from Google News

Figure 7.4 shows a screenshot of the TellMe prototype client application displaying the aggregated ranked list of services, and relevant information retrieved from each service (for each of its 3 most relevant categories), to the user via the calendar event gadget. Since the current client is a web-based system, the server side system utilizes the current geolocation of the user obtained from the user's browser to filter the information for each category.

As other services registered with the Service Tier (such as those for weather, music etc.) do not provide information relevant to this task, they are not discovered or ranked. This list is intuitive and beneficial to the user as it provides him with helpful suggestions

on laboratories, medical centers and hospitals, near his current location, where he can go and get the tests done. Moreover, it enables him to discover new information serendipitously, such as information about alternative medicine or latest news regarding health and science, which further aids him in making an informed decision.

7.3.3.2 Use Case 2

Similarly, consider another situation item - 'Go sailing'. The STS-ASR variation of Algorithm 3 generates the following ranked list of relevant services and categories:

1. 'Surfing', 'Diving', and 'Boats' from Yelp
2. 'Wind', 'Weather', and 'Visibility' from Weather Underground
3. 'Navigation', 'Routes' and 'Maps' from MapQuest

In this case, the STS-ASR algorithm aggregates a wide variety of content which could benefit the user. For instance, the user may want to rent boats for sailing. In addition, he may want to check weather conditions before undertaking any water sport activity.

These two use cases highlight the diversity of the information needs of users according to their situations. It also demonstrates that the TellMe system can aggregate heterogeneous content from various sources, which the user might have been oblivious of but, nonetheless, proves valuable in decision making.

7.3.4 Wikipedia-based Semantic Relatedness

Algorithm 4 outlines the process of calculating semantic relatedness between two phrases based on Wikipedia. In this algorithm, we first tokenize each phrase (which can be a sit-

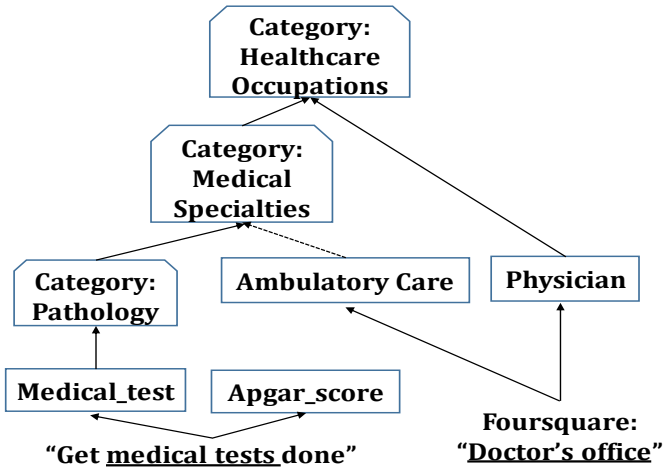


Figure 7.5: Graphical illustration of the lch calculation for the phrases "Get medical tests done" and "Doctor's office".

uation description or a service category in our case) and disambiguate its meaning using the substring dictionary provided by Spitkovsky and Chang [107]. We have processed this dictionary to be of the form: $substring \rightarrow (Concept, p_l, p_c)$ where Concept refers to a Wikipedia article, p_l is the probability of the substring linking to the specific concept and p_c is the probability of the concept itself. These substrings come from link text in Wikipedia and internet crawls, and their tf-idf values are used to generate p_l and p_c. For performance optimization, we have discarded substrings shorter than 2 characters (for instance, really short strings such as 'I') and longer than 32 (for instance, extremely long words). We have also discarded high probability concepts ($p_c > 0.01$) which represent stop words and low probability concepts ($log(p_c) < -17$) which represent very rarely used uncommon terms. Finally, we have discarded substrings that have greater than 10,000 concepts (such as "here").

To disambiguate a phrase, we first consider all combinations of unigrams and bigrams

obtained from it. We then map these to Wikipedia concepts, retrieve the concepts with the highest likelihood and select the concept combination that has the highest inter-concept semantic relatedness. Inter-concept semantic relatedness is a metric to measure how close the concepts are within a phrase. By maximizing this metric within phrases, we are able to correct cases where low-likelihood but high relatedness concepts would be discarded by a context-free statistical calculation. We use the lch function [108] between the concepts to provide a semantic relatedness measure. This function calculates the closest shared ancestor between two articles in Wikipedia category space and has been proved to be a decent approximation of semantic relatedness [109]. For each concept combination, we calculate this function and return the pair with the highest score.

Figure 7.5 graphically illustrates the computation of semantic relatedness between the situation description "Get medical test done" (Use case 1) and category "Doctor's office" from Foursquare using Algorithm 4. First, we map from word phrases to concepts represented by Wikipedia articles. In this case, the bi-gram "medical test" is linked to the Wikipedia article with the same name. It also links to other articles such as "Apgar score" which is a type of medical test. The Foursquare category "Doctor's office" links to both "Physician" and "Ambulatory care". In this case, Ambulatory care has a higher link probability, but Physician is a more probable concept. The next step is to calculate the distance between the two phrases by examining their lch score. If "Physician" were selected, the closest common ancestor is the category "Healthcare Occupations" which leads to a distance of 4. If "Ambulatory care" were selected, the distance between "Medical test" and "Ambulatory care" is 3. We have combined these mechanisms to exploit the execution speed and automatic curation available through crowd sourced platforms such

as Wikipedia.

7.4 Evaluation

7.4.1 Goals and Methodologies

The primary goal of our study is to evaluate the effectiveness of the TellMe system, and its underlying algorithms, in discovering and ranking services and information relevant to a user's situation or task. We test the validity of our hypothesis - that semantic relatedness can be used as a measure of relevance of information to a user's situation and can be further used to rank it. To this end, we model our experiments after those in Information Retrieval (IR). In IR, retrieval correctness usually cannot be proved formally and hence, evaluation often relies on human assessment of result quality [110]. Thus, we analyze the results generated by our algorithms through various standard performance measures commonly used in IR, such as Precision and Recall, in order to assess their effectiveness and compare their performance with human generated relevance rankings. We also evaluate TellMe against several qualitative metrics such as generality, response time and robustness. In order to evaluate our system, we used two methodologies:

7.4.1.1 Live deployment

We recruited 14 (8 male and 6 female) subjects, who were either working professionals or campus community members, to evaluate the TellMe system. None of these subjects had interacted with TellMe or Rover II before and hence, we did not have any usage or interaction logs for them. All the subjects had to import the TellMe calendar event

gadget client application (as shown in Figure 7.4) into their Google calendars and had to authorize it to access their calendar data and current geolocation for a period of 1 month. In addition, the subjects were given specific instructions to use their calendars as the main tool for scheduling appointments, to-do tasks and reminders during the study period.

For each calendar entry, the gadget displayed *in situ* the top 3 ranked services, and information for the top 3 categories for each service, as retrieved by the TellMe server side system, to the subject. Since there maybe personal biases of users towards services, we currently do not remove information for overlapping categories between the ranked services in the results. Also, since the TellMe server side system currently has information about the users' current location but not about their preferences (for food, music etc.), the results are personalized based on current location only. The subjects were asked to provide feedback by marking the information displayed to them as 'Relevant'/'Not relevant' via the gadget.

7.4.1.2 Web-based study for Ground Truth collection

The methodology for live deployment doesn't provide a direct means for evaluating the efficacy of the algorithms via standard IR performance measures. This is because the subjects could not provide *Ground Truth* on what information they considered relevant in comparison to what the system displayed. Moreover, all the subjects had varying calendar entries (with only a few common ones as shown in Table 7.2) and could provide feedback only for their calendar entries. This created sparseness in the results.

Hence, we used another methodology to collect ground truth for relevance compari-

son. After the live deployment, we aggregated the calendar, to-do and reminder entries of all the subjects collected during the live deployment period. We further augmented these with diurnal activities such as 'Driving' to generate 120 unique situation and task items. Table 7.2 shows the most commonly recurring situation and task items collected as part of our user study.

All the 14 subjects who participated in the live deployment further participated in a web-based study. For each of the 120 items, they were asked to rank up to 3 services (from among the 8 services currently registered in the TellMe server side system) that they considered most relevant to the task or situation. For each ranked service, they were asked to select up to 3 of its most relevant information categories. If none of the categories seemed relevant to them, they were allowed to provide their own keywords. This collected data was then used as ground truth and we compared the results generated by our system with it.

7.4.2 Evaluation Metrics and Results

Relevance is a subjective measure and as such there may not be agreement among the users. In our study, there is a small subset of services (such as Yelp, Foursquare and Angie's List) that have some overlap while others provide mostly mutually exclusive information. As a result, the service rankings responses have high agreement. However, among overlapping services there may be slight disagreement among users that can be caused by personal biases towards services (say preferring Yelp to Foursquare).

On the other hand, there are significantly more categories than services. For instance,

Yelp had 698 and Foursquare had 402 categories at the time this research was conducted. As stated earlier, the category structures are hierarchical. For instance, the general category “restaurants” subsumes more specific categories such as “Indian restaurants” or ”Chinese restaurants”. Thus, there is lower agreement for category responses mainly caused by either lack of full knowledge of the categories or by a user’s preferences (causing them to pick more specific categories). To address issues with lack of knowledge of categories, we had provided category auto-suggestions (based on substring matching) as a user provided relevant categories in the web-based study.

Also, service discovery is the first step for retrieval of relevant information in our system. If the appropriate services are not discovered then the relevant information from them can not be retrieved. Due to these inherent differences in the services and categories discovery process and response spaces, we have split our evaluation to independently evaluate service rankings and categories retrieval.

7.4.2.1 Service ranking

Although all subjects were asked to rank up to 3 services, most subjects ranked only 1 and in a few cases, 2. On the other hand, our algorithms ranked 3 or 4 candidate services as relevant for each situation item. This created data that is too sparse for a thorough accuracy evaluation and hence, we use alternative measures to show how the algorithms performed with respect to Service Ranking:

- Friedman Test

  To quantify the consistency of responses, we first applied the Friedman test - a

non-parametric statistical test that is used to detect differences in treatments across multiple test attempts. We applied it with an α (significance) level of 0.05 to the service ranking responses of all subjects for all items in our study. We use this to determine how consistent the subjects were in ranking the services for each item and if there were any significant differences in their responses. The null hypothesis of this test, when applied to our study, is that there is no difference in the rankings provided by each user. When a difference occurs, it may imply a matter of differing personal biases towards services or an ambiguous task such as 'Purchase birthday gift' which has no obvious "correct" answer.

Out of the 120 items, for 85 (70.8 %) the null hypothesis held - there was no significant variation in how subjects ranked the services. 4 (3.3 %) of the situations did not have enough services, ranked by the subjects, to provide a meaningful measurement. This occurred when there were no or very few obvious relevant information providers for the specified situation (for instance, a situation item such as 'Water the plants'). Finally, there were 31 (25.8 %) situations that showed statistically significant variance in subject response. As previously stated, this may imply some underlying difference in personal biases or situations with no "correct" answer.

- Spearman's rank correlation coefficient

  To provide a single user generated baseline for comparison, we applied Borda counting to combine service ranking results from all subjects into a hybrid service ranking for each situation or task item. Borda counting is a type of preference voting which gives varying points for each rank (more for higher, less for lower

Service	Rank			Points	Final Rank
	1	2	3		
Yelp	8	2	0	8x3+2x2+0x1=28	1
Foursquare	3	6	0	3x3+6x2+0x1=21	2
Angie's List	2	1	2	2x3+1x2+2x1=10	3

Table 7.3: Service ranking results for all the 14 users and Calculation of points for each ranked service using Borda counting to generate singe user provided baseline

ranks). Table 7.3 shows an example of how Borda counting is applied to subjects' rankings for the task item 'Get medical tests done' in order to generate a single unified ranking. Each cell value (in clomuns 2 -4) represents the # of users who assigned the rank, represented by the column, to the service represented by the row. These rankings are used to calculate the points awarded to each service (column 5) and generate the user baseline ranking (column 6).

We then applied Spearman's rank correlation coefficient (ρ) to measure similarity between the user generated baseline and the various service rankings, generated by the four algorithms and the subjects, for each item in the study. Spearman's ρ is a nonparametric rank statistic that determines how close two variables are by quantifying the strength of the associations between two vectors. Its value ranges from -1 (perfectly reversed; negatively correlated) to 1 (equal; positively correlated). We measured ρ for each algorithm's service ranking, for each of the 120 situation items, against the user generated baseline produced by Borda counting. We performed the

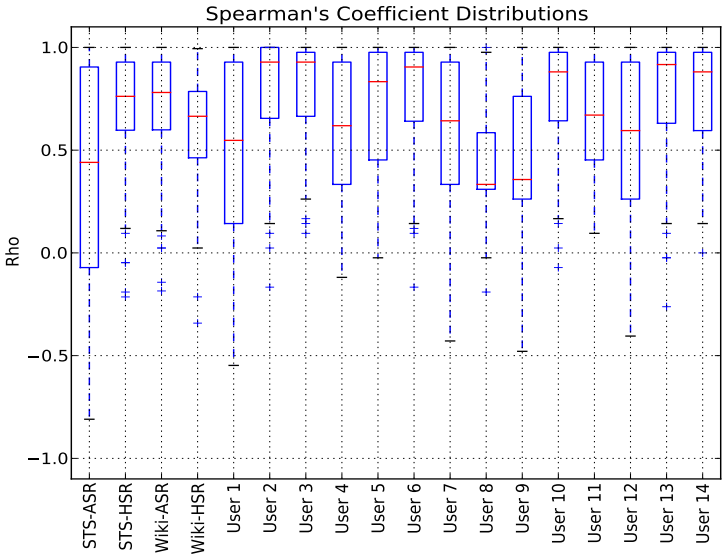


Figure 7.6: Summary of the distribution of ρ between user generated baseline and service rankings of individual algorithms and subjects

same measurement for all the test subjects to show how well they performed against the baseline.

Figure 7.6 shows the boxplots for the distribution of ρ for all algorithms and subjects against the baseline. First, we note that the STS-based algorithms are very sensitive to the service scoring function. The STS-HSR algorithm shows very strong correlation with the baseline in both its median and its interquartile range (IQR). In contrast, STS-ASR has the poorest performance in both measures showing little to no correlation in many cases. The Wiki-based algorithms are more closely matched and show very strong correlation in both median and IQR. Unlike the STS-based algorithms, the Wiki-based algorithms perform better with the ASR variant probably because of the tight range of the lch calculation. Overall, the algorithms (sans STS-ASR) show a strong positive correlation with the user generated baseline, thus,

proving that they perform comparably to human supplied relevance rankings.

We also analyze the subjects' rankings against the baseline to show that it is reasonable. Examining their coefficients, we see similar variation in responses. Almost all subjects have upper quartiles with $\rho > 0.5$ showing a medium to strong positive correlation with the baseline. Subjects 2, 3, 6, 10, 13, and 14 show strong positive corrections in almost all cases. Subject 8 has the worst median correlation but a fairly tight variance whereas subject 1 has slightly stronger median performance but the largest variance. Overall, the subjects demonstrate a strong positive correlation with the baseline.

- Precision and Precision@k

  Spearman's ρ does not map directly to user experience. Hence, to put these measurements into perspective, we measured Precision (P) and Precision@k (P@k) of the service ranking determined by our algorithms where

$$\text{P} = \frac{\text{Retrieved services} \cap \text{Relevant services}}{\text{Retrieved services}}$$

  As stated earlier, most subjects ranked only 1 or 2 services as relevant in comparison to our algorithms which ranked 3 or 4 services for each situation item. Hence, for P@k, we set k =1 i.e. we compute the % of times the subjects' rank #1 service matched with the rank # 1 service returned by each algorithm.

  Table 7.4 shows a summary of performance for the user generated baseline and the algorithms for service ranking. We can see that the baseline agrees with the subjects' ranked # 1 service, 70% of the time and this rank # 1 service would appear

in the baseline's top 3 services, 94% of the time. STS-ASR performs the worst in precision, only agreeing with the subjects' rank # 1 service 33% of the time and only retrieving the # 1 service in its top 3 results, 53% of the time. STS-HSR and Wiki-ASR have very similar correlation profiles (when considering ρ) but Wiki-ASR outperforms STS-HSR (when considering Precision) retrieving the # 1 service in top 3 results, 87% of the time as opposed to 80% for STS-HSR. Finally, Wiki-HSR, which had a slightly weaker median correlation (for ρ) outperforms all the algorithms, ranking the subjects' # 1 service as its # 1 service, 77% of the time (more than the user generated baseline) and retrieving it in its top 3 services, 92% of the time (comparable to the user generated baseline). This again demonstrates that the algorithms (sans STS-ASR) perform comparably to humans supplied rankings.

7.4.2.2 Category analysis

Since there is higher disagreement among category selections by users, mainly because of the huge number of categories to choose from, we do not compute a single user generated baseline for it. Similar to service ranking, we use the metrics of Precision (P), Recall (R), F-score (F) and Precision@k (P@k) to evaluate category retrieval by our algorithms. Here,

$$\mathrm{P} = \frac{\text{Retrieved categories} \cap \text{Relevant categories}}{\text{Retrieved categories}}$$

,

$$\mathrm{R} = \frac{\text{Retrieved categories} \cap \text{Relevant categories}}{\text{Relevant categories}}$$

and

$$\text{F} = \frac{2 * P * R}{P + R}$$

Table 7.4 provides a summary of the results for these metrics. First, we examine the % of categories that were retrieved by each algorithm for each situation item and were considered relevant by the users (P). Next, we evaluate the % of relevant categories that were selected by the users and retrieved by each algorithm (R). F is the accuracy measure obtained by computing the harmonic mean of P and R. Finally, we measure P@ k. Since our algorithms return 3 categories for each situation item, we set k = 1. Thus, we measure % of times the users' rank #1 category matched with the rank # 1 category returned by each algorithm.

As evident, the STS-HSR algorithm had the best performance for all metrics. It retrieved the rank # 1 category of users as its rank # 1 category (P@1) around 73% of the time while STS-ASR retrieved it around 57% of the time. In addition, it retrieved useful and relevant categories in its top 3 categories (P) 81% of the time while STS-ASR retrieved them around 66% of the time. Overall, STS-HSR has an accuracy (F) of 75% as opposed to STS-ASR which has an accuracy of 60%. STS-HSR outperformed STS-ASR mainly because the ASR scoring function did not rank some of the relevant services (as evident by its poor performance in service ranking) and did not retrieve the categories for them.

The Wiki-based algorithms performed much worse with an accuracy of 11%. A reason for their poor performance could be that the lch calculation has a very tight range. For a situation such as "Get medical tests done", many of the medical-based concepts will

Algorithm	Service Ranking		Category Retrieval			
	P	**P@1**	**P**	**R**	**F**	**P@1**
Baseline	0.94	0.7	—	—	—	—
Wiki-HSR	**0.92**	**0.77**	0.13	0.1	0.11	0.1
Wiki-ASR	0.87	0.66	0.13	0.1	0.11	0.1
STS-HSR	0.8	0.51	**0.81**	**0.69**	**0.75**	**0.73**
STS-ASR	0.53	0.33	0.66	0.55	0.6	0.57

Table 7.4: Analysis of P and P@1 for service ranking of user generated baseline and algorithms and Summary of P, R, F and P@1 for category retrieval by the algorithms

have the same score leading them to be returned alphabetically. This helped with service discovery because services such as Yelp (which have several medical categories) would score higher than services such as Google News (which has a single health category). To improve these results, a different tie breaking mechanism, based on category probabilities or user preferences, could be used.

7.4.2.3 Discussion

Our initial results are highly promising. With respect to service ranking, the proposed algorithms performed comparably to humans. However, for category retrieval, the algorithms performed reasonably. A possible reason could be that the current system does not have information on individual user preferences and hence, cannot personalize the categories for users. Similar to service ranking, all subjects were asked to select up to 3

relevant categories for each ranked service. However, most subjects selected only 1 or 2 categories and their selections reflected their individual preferences for some of the situations (such as 'Indian restaurants' for 'Having Lunch' rather than just 'Restaurants'). On the other hand, our algorithms retrieved information for general categories such as 'Restaurants' and not for specific categories, such as 'Indian restaurants'.

However, since the aim of this study is to show the value of our approach for bootstrapped discovery and ranking of heterogeneous services and information relevant to a user's situation, we leave preference tuning for the future. It can be addressed by modeling users' preferences from various sources such as their social media profiles or via preference elicitation. Thus, a hybrid system that combines our approach with user preference modeling for personalizing the results will achieve an even higher accuracy for category retrieval. Please note that a direct comparison of our work with [85] is not possible as they do not present any accuracy evaluation.

Also, we observed that there were some subtle relationships, between a situation item and the relevant information selected by users, which are hard to model using existing NLP techniques. For instance, multiple users selected weather information as relevant information for many of the situations (such as "Mow the lawn"). We believe that these subtleties can be captured by mining terms that commonly co-occur in web search queries and using them to boost the SR scores. We are currently working in this direction.

7.4.2.4 Qualitative Metrics and Results

We further evaluate TellMe against several qualitative metrics:

- Flexibility and Generality - We note that we have used web-based services as a primary and motivating information source in our current implementation of the system. However, our approach is flexible enough to incorporate other sources such as databases, and smartphone and desktop applications. This only requires a list of categories of information provided by the new sources, along with an API or an access mechanism to retrieve information.

  Moreover, the system is not restricted to a domain-specific ontology or static pre-defined rules. Instead, it employs techniques that determine relevant information dynamically from services that provide heterogeneous content. It leverages robust and domain-independent databases such as WordNet and Wikipedia that contain a large amount of world knowledge. As a result, it is generic and can support a wide spectrum of users' tasks and situations from their daily lives.

- Turn around time - The STS and Wikipedia based systems for computing Semantic Relatedness are deployed locally on another server where the language models are held in memory. Hence, the turnaround time is very fast (in the order of ms). Moreover, we cache the SR scores, between categories and situation items, for future use. As a result, any recurring items will have faster turnaround time in subsequent calls.

- Bootstrapped and Robustness to cold start - As evident by our experiments, the TellMe system can provide relevant information to new users and for new unanticipated user situations without requiring any user interaction history. Thus, it is bootstrapped and capable of avoiding a cold start.

Chapter 8: **Unsupervised Modeling of Users' Interests from their Facebook Profiles and Activities**

Once the relevant information has been determined, it should be personalized based on users' individual preferences. These preferences can be either obtained explicitly by asking users to create user interest profiles or by modeling these profiles in an automated manner which is preferable and less burdensome. In today's world, social networks such as Facebook or Twitter provide users with a powerful platform for interest expression and can, thus, act as a rich content source for automated user interest modeling. This, however, poses significant challenges because the user generated content on them consists of free unstructured text. In addition, users may not explicitly post or tweet about everything that interests them. Moreover, their interests evolve over time. In this work, we address these challenges and present a novel unsupervised algorithm and system [11] for modeling users' interests from their Facebook profiles and activities.

8.1 Introduction

As the amount of information available to us increases, a common phenomenon that occurs among users is information overloading. This can be addressed by personalizing information streams and services, and user interfaces and experiences for individual users.

User interest profiles are essential for such personalization as they enable recommender systems to provide users with highly personalized content in several domains (such as sports, music etc.). Automated user interest modeling that involves extracting and inferring user interests without any user input presents a burden free method for generating such profiles.

As a simplest formulation, user interest profiles can be modeled as a *Utility Matrix* where rows represent users, columns represent items and the values represent the users' levels of interest in those items on a scale, say, 1 - 5 [111]. For instance, it is possible to generate a utility matrix for users from their interaction with a movies website and the movie ratings provided by them. Furthermore, these ratings can be leveraged to predict users' preferences for new unseen movies using a technique such as *Collaborative Filtering* [112, 113] that exploits user -user and movie -movie similarity information. Ultimately, this enables a movie recommender system to make personalized movie recommendations based on the users' derived and predicted movie preferences.

User interest profiles can also be developed by utilizing domain knowledge about the users or about the items they have expressed interest in. These approaches are referred to as *Content-based approaches* [113] which model user preferences by representing the item, that the user has expressed interest in, in terms of its attributes and building the user's interest profile based on that (for instance, movies can be represented by genre, cast etc.) They are particularly useful when data from similar users may not be readily available due to privacy reasons. Moreover, they are capable of recommending information that has not been rated before by any user and can also accommodate differences between individual users.

A popular content source for such approaches is users' interaction (such as time spent on web pages or click history) with various websites, for instance, news websites [114]. The websites' content can then be mined to derive their interests and preferences. However, most websites offer very specific content (movie reviews, news articles etc.) and hence, only a limited set of user interests can be modeled from them. To address this, modeling users' interests based on their entire web search and browsing history has been researched extensively [115–117].

In today's world, social networks and media such as Facebook or Twitter provide users with a powerful platform for interest expression across several domains. They have become a popular medium for users to connect, explore, share content and express themselves. For instance, Facebook users can 'like' (described as a way to *"give positive feedback and connect with things a user cares about"*) items such as Facebook pages. They can share URLs and videos, or post status updates and comments about topics that interest them. Thus, user profiles and activities on these social networks can act as a rich content source for automated user interest modeling.

However, a significant challenge in modeling user interests from their social network profiles and activities is that the user generated content on them (such as posts, tweets etc.) often consists of free unstructured text which is far from any form of items and user interest levels. Moreover, in practice, users may not explicitly tweet or post on these networks about everything that interests them. In addition, their profiles and activities evolve over time based on their dynamic interests. As a result, new algorithms and techniques need to be developed in order to build accurate and exhaustive user interest profiles from their activities on social networks.

In this work, we address these challenges and present an approach for building comprehensive hierarchical user interest profiles from their social network profiles and activities. These interest profiles represent the users' levels of interest in over 200 coarse to fine-grained categories covering a wide spectrum of popular interests. To this end, we extract users' *explicit* interests (that have been directly expressed through tweets or posts) from user generated content on a social network platform such as Facebook. We further infer their *implicit* interests in order to augment their interest profiles. Thus, our contributions are:

1. We propose a novel unsupervised algorithm and system, that utilizes several Natural Language Processing (NLP) techniques, to model an individual user's explicit and implicit interests from her social network profile and activities without any user input. The NLP techniques that we have employed include named entity recognition, document categorization, sentiment analysis, semantic relatedness and social tagging.

2. We perform extensive evaluation of our system, and its underlying algorithm, with a dataset consisting of 488 active Facebook users' profiles and demonstrate that it can accurately estimate a user's interests.

3. We compare our algorithm with several baselines and show that it outperforms all of them.

4. We establish the value of using techniques such as Semantic Relatedness for inferring implicit user interests, and using features such as social tags and document

categories for modeling her fine-grained interests. We demonstrate that this enhances interest prediction.

5. We also consider temporal dynamism of users' interests and demonstrate that it improves prediction of their interest levels.

We note that while we have used Facebook profiles as a primary and motivating dataset, our approach can be easily extended to other social network profiles.

8.2 Related Work

In this section, we first briefly discuss related work in user interest modeling from search and browsing history. Since existing literature in user interest modeling from social media and networks is more pertinent to our work, we discuss it in detail after that. We highlight their shortcomings as well as differences with our approach.

8.2.1 User interest modeling from search and browsing history

Modeling users' interests from their browsing history has received significant attention recently. Eichstaedt et al. [115] present a method for generating user interest profiles from structured documents such as browsed web pages. The profiles include a hierarchy of interest categories with a score for each category (representing the user's interest level) assigned based on the text content and classification of the viewed documents. The hierarchy is periodically refined to reflect dynamically changing interests. Kim and Chan [118] model hierarchical user interests by clustering user's bookmarked web pages.

Qiu and Cho [116] personalize search results and improve ranking mechanisms, by

automatically learning a user's preferences based on past click history. Liu et al. [114] present a hybrid approach, that combines a content based approach with collaborative filtering, for personalized news recommendations based on users' click behavior. White et al. [117] map a user's browsing history URLs into Open Directory Project Ontology Concepts[1]. They further utilize the browsing behavior of several users, over a window of time, in order to predict the user's interests. Ramanathan et al. [119] create a hierarchical user profile from Wikipedia pages visited by a user. Min and Jones [120] employ unsupervised clustering of Wikipedia pages to infer a user's coarse-grained interests.

8.2.2 User interest modeling from social media and networks

Social network profiles, such as Facebook profiles, offer a wealth of information for modeling users. Golbeck et al. [121] and Bachrach et al. [122] derive the personalities of users (in terms of the Big Five personality model) based on characteristics of their Facebook profiles such as the size and density of their friendship network, number of uploaded photos, number of events attended, etc. Viswanath et al. [123] examine the evolution of activity and interactions between 60,000 Facebook users over a period of 2 years. Kosinski et al. [124] predict personal attributes of users such as ethnicity, religious and political views, based on the pages that they liked on Facebook. However, these approaches do not attempt to model individual user interests and preferences.

Mislove et al. [125] propose an approach, based on community detection in social networks, to infer user attributes from their friends' attributes. Ho et al. [126] explore how friendships and conversations on Facebook are influenced by users' shared interests.

[1] `http://www.dmoz.org/`

They estimate shared interests based on whether users liked the Facebook pages for four popular interests such as 'Camping' etc. On the other hand, in our work, we employ a content-based approach and utilize an individual user's Facebook profile, instead of profiles of other users in her network, to model a wide spectrum of her interests.

Abel et al. [127] employ a bag of words approach to explore whether a user's professional scientific interests overlap with her social web interactions and if this overlap can be used to recommend relevant publications. Abel et al. [128] utilize a user's Tweets in order to construct different user profiles based on extracted hashtags, entities and topics. They also examine how these profiles perform with respect to personalized news recommendations. They further analyze temporal patterns (such as weekday/weekend differences) in users' tweeting behavior and their effect on personalization. Esparza et al. [129] explore the idea of using user generated content on real time web, such as Twitter, as a source of knowledge for building recommender systems. Esparza et al. [130] present a user profiling approach based on topical categorization of URLs present in Tweets. They achieve a mean profile prediction accuracy of 0.73 for 32 users over 18 coarse-grained interest categories (such as music, food etc.). Pennacchiotti and Popescu [131] proposed a supervised learning approach to classify Twitter users into 3 attribute based categories representing political affiliation, ethnicity and business fans. They employed features such as user's Twitter profile, tweeting behavior and content, and social graph.

Even though our system shares some similarities with the content source being used in these works (Facebook profile and activities etc. vs Twitter profile and Tweets), we propose an unsupervised approach. Moreover, we models users' interests in over 200 hierarchical, coarse to fine-grained categories covering different domains such as food,

entertainment etc. This is a significantly challenging task. In addition, we consider dynamic changes in users' interests with time. Also, our approach employs several NLP techniques in addition to topical categorization and entity recognition. As we demonstrate in our evaluation, these techniques improve interest prediction accuracy.

Ma et al. [132] model users' interests based on terminology specific keywords extracted from their personal webpage and profiles on social networks such as Facebook, LinkedIn, etc. They further infer higher level interests from those keywords using a predefined ontology. This supervised approach has limits as keyword extraction is domain-specific. Moreover, it requires ontology generation for each domain. These limitations motivate a deeper exploration of how a broad range of interests can be modeled in an unsupervised manner from a user's social network profiles and activities.

8.3 Facebook User Profiles

Facebook is one of the most ubiquitous and popular social networks. Recent Facebook statistics [133, 134] show that it has around 1.11 billion active users. It processes 2.5 billion pieces of content and 500+ terabytes of data each day. It has 2.7 billion Likes. A Facebook user creates 90 pieces of content every month on an average.

Though the Facebook social network has a number of features [135], we focus predominantly on the text content of an individual user profile. Each user profile consists of several sections such as basic profile information (personal and professional), *check-ins* which represent all the places/locations that the user has visited, *notes* which are user generated write ups, photos that the user has taken etc. The two most significant sections that

Questions on a Likert Scale of 1 (Highest) to 5 (Lowest)
How often do you listen to music? For the following genres, rate your interest: Asian, Rock, Easy Listening, Rhythm and Blues, Jazz, Hip Hop, Electronic, Pop, Country or Folk, Avant-Garde
How often do you read books? For the following genres, rate your interest: Romance, Travel, Horror, Children's, Nonfiction, Fantasy, Historical/ Realistic/ Science Fiction, Short story, Mystery, Humor, Biography, Poetry etc.

Table 8.1: Sample questions from the survey administered to Facebook users

we analyze are liked *Pages* and *Timeline*. A page can represent any entity and contains detailed information about it. In addition, each page also has a category such as 'Book', 'Movie' etc. assigned to it from among the Facebook Page categories[2]. The Timeline reflects the user's Facebook activity such as her status updates and posts along with comments and likes, posts from another Facebook user etc. Most of the profile sections remain static or expand as the user adds more content or likes more pages[3]. However, the timeline is dynamic and varies temporally depending on how active a user is.

A technique to infer explicit coarse-grained interests for a user would be to utilize the names and categories of liked pages in her profile. However, as we show later in our evaluation, this approach performs fairly in comparison to our approach which infers implicit user interests in addition to such explicit interests. Moreover, we also address modeling fine-grained interests such as specific genres of books and movies etc.

8.4 Data Collection

Since Facebook has no public API, we recruited 728 Facebook users through tasks posted on Amazon Mechanical Turk. The recruitment criteria were: (i) they should be based in

[2] `https://www.facebook.com/pages/create/` [3] We assume that once a user has liked a page, he will rarely 'unlike' it.

the US (so that their profiles would be predominantly in English), and (ii) they should be currently active Facebook users with at least 500 or more Facebook activities such as liked pages and posts (as modeling users with sparse profiles can be error prone). Each task lasted an hour on an average and each user was paid $1 each for successfully completing it. Through the task, the users had to authorize a Facebook application to allow access to their profile. Once authorized, it directed them to a secure server where a deployed web application retrieved their Facebook profile data. They also completed a web-based survey that gathered their demographic, Facebook activity frequency, and interests and preferences information. The survey consisted of 44 quantitative questions which were either open ended, choice based, or Likert scale based. Table 8.1 lists a few survey questions.

Out of the 728 users, 472 (64.9%) were women and 255 (35.1%) were men. The largest group consisting of 266 (36.6%) users was working full time while 174 (23.9%) users had completed university. 704 (96.7%) users listed English as their native language. 417 (57.3%) users admitted visiting Facebook 2-10 times a day, while 375 (49.1%) participants said they spent 10 - 60 minutes per day on Facebook. 295 (40.5%) participants stated that they posted status updates several times a month.

From the collected dataset, we invalidated the data of 240 users. For these users, either the profile data was too sparse (for instance, very few posts and liked pages), incomplete (e.g. all posts missing) or we found missing or contradictory responses in their surveys indicating that they did not fill it out properly (incomplete surveys). Our final dataset consists of 488 users, each of whom had an average of over 2000 pages and posts etc.

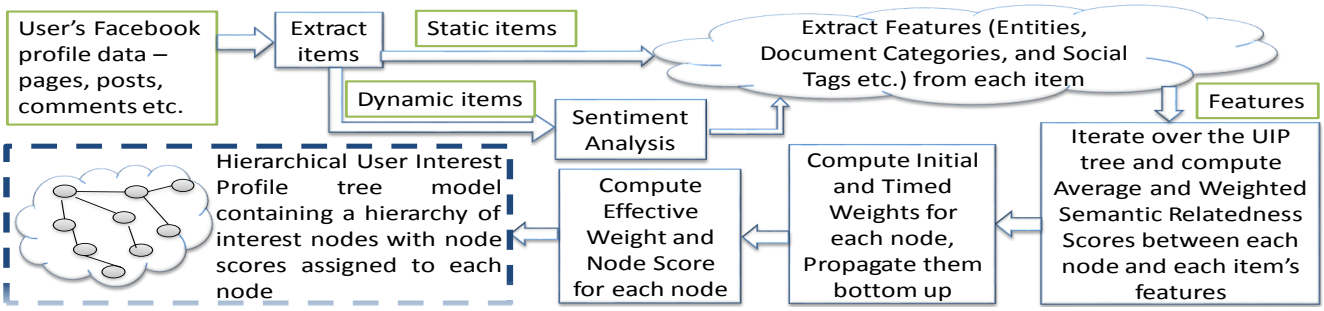


Figure 8.1: User Interest Profile Generation Pipeline

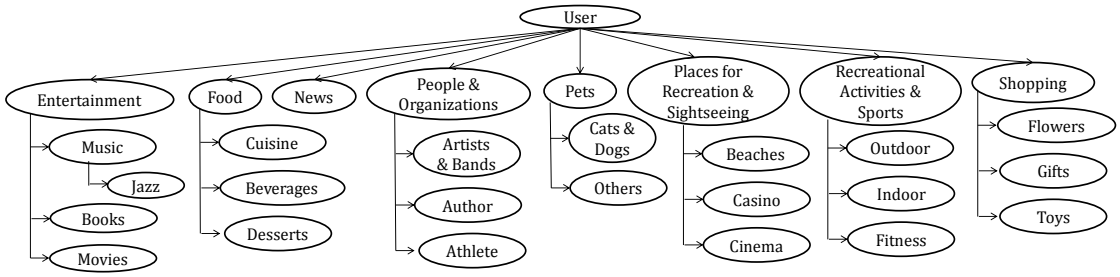


Figure 8.2: Partial view of the User Interest Hierarchy showing all depth 1 nodes and a few depth 2 and depth 3 nodes

8.5 User Interest Modeling

Figure 8.1 shows an overview of the User Interest Profile Generation system pipeline. It generates a hierarchical **User Interest Profile (UIP)** tree model, for each user, from the user's Facebook profile data. Each user's UIP tree is structured as our User Interest Hierarchy (Figure 8.2) with a score assigned to each node, which represents the user's level of interest in the node category.

Algorithm 5: User Interest Profile Generation Algorithm

Input: Facebook profile text data, User Interest Hierarchy (UIH)

Output: User Interest Profile (UIP)

Initialize UIP tree using UIH with Initial Weights = 0.0 and Node Scores = 0.0 for all nodes;

Extract user generated static and dynamic items from a user's Facebook profile such as liked pages, posts etc.;

foreach *item* i **do**

  Extract *features* from *i* such as Named entities and their types, Document categories etc.;

  Calculate Item Weight of *i*: w_i;

  foreach *Node n in the UIP tree* **do**

    Compute ASR_n = Average Semantic Relatedness score between *n* and features of *i*;

    if $ASR_n < SR_{threshold}$ *(threshold SR score)* **then**

      replace ASR_n with 0.0;

    Compute Weighted Average Semantic Relatedness score: $WSR_n = w_i \times ASR_n$;

    if i *is a static item* **then**

      Compute Initial Weight (IW) of n: $IW_n = \arg\max$ (Current IW_n, WSR_n);

      if *n is a leaf node* **then**

        Compute IW of n's parent: IW_{n_parent} = arg max(Current IW_{n_parent},arg max(IW of n and siblings));

      else

        IW_{n_parent} = arg max (Current IW_{n_parent}, Average IW of n and siblings);

    else if i *is a dynamic item with positive or neutral sentiment* **then**

      Compute kth Timed Weight (TW) of n: $TW_{n,k} \leftarrow <T_{i\_LastUpdate}$ (LastUpdate Timestamp of i), $WSR_n>$;

      if *n is a leaf node* **then**

        Compute k'th TW of n's parent: $TW_{n\_parent,k'} = <T_{i\_LastUpdate}$, arg max($TW_{n,k}$ of n and siblings)> ;

      else

        $TW_{n\_parent,k'} = < T_{i\_LastUpdate}$, Average $TW_{n,k}$ of n and siblings>;

  end

  if arg max*(*ASR_n*)* $< SR_{threshold}$ **then**

    Create a new node for i in the UIP tree;

end

foreach *Node n in the UIP tree* **do**

  Compute Effective Weight: $EW_n = IW_n \times \prod_{j=1}^{k}$ (1 - $TW_j \times e^{-\lambda(T_{Current} - T_{j\_LastUpdate})}$);

  Node Score of node $n \leftarrow EW_n$ mapped to a linear scale of 1-5;

end

return *User Interest Profile tree;*

8.5.1 User Interest Hierarchy

Figure 8.2 shows a partial view of our User Interest Hierarchy. It has been derived from the Yelp category list[4] and the Facebook Page categories, which cover a wide range of popular lifestyle and interest categories. The Yelp category list is hierarchical while Facebook Page categories are generic, coarse-grained and flat. As a result, we have attempted to build a hierarchy that would cover a significant range of popular user interests (without being too specific) by merging the two lists and filtering out very fine grained categories.

Our hierarchy has a tree structure with depth 3 and contains 204 nodes including the root node 'User'. There are 8 high-level (depth 1) categories which branch out into 58 coarse-grained categories (depth 2). A few coarse-grained categories at depth 2 are further categorized into 137 fine-grained categories (depth 3). The high-level categories, and some of the coarse-grained and fine-grained categories that they include, are shown in Figure 8.2. For instance, 'Entertainment' (depth 1) subsumes 'Music' (depth 2) which further subsumes 'Jazz' (depth 3).

It is possible that a user may be interested in a category not included in our hierarchy. As explained later, it will be added as a newly discovered interest category in that user's profile.

8.5.2 User Interest Profile Generation

The pipeline for user interest profile generation employs the proposed User Interest Profile Generation Algorithm (Algorithm 5) and consists of the following phases (for some

[4] `http://www.yelp.com/developers/documentation/category_list`

phases, the design choices are explained in 'Selection and Tuning of design schemes and parameters'):

8.5.2.1 Item Extraction

As mentioned earlier, most of the profile sections remain static or expand while the timeline varies dynamically with time. We extract all such user generated *static items* (profile content such as notes, liked pages, etc.) and *dynamic items* (contents of the timeline such as status updates, posts and comments etc.) from a user's Facebook profile. Each dynamic item also has a timestamp indicating when it was generated or last updated.

8.5.2.2 Sentiment Analysis

Each dynamic item is subjected to sentiment analysis (a technique to determine the attitude or emotional state of a speaker or writer with respect to some topic). We determine the item's sentiment polarity, which can be negative (< 0.0), neutral (0.0) or positive (> 0.0), using a well-known tool called Alchemy[5].

In our current implementation, we utilize only those dynamic items for a user that have neutral or positive sentiment as we are estimating her likes rather than her dislikes. This is not required for static items such as liked pages as liking implies positive interest.

8.5.2.3 Feature Extraction

From each item, we extract *features* such as page category (available only for liked pages), and other rich semantic information that includes named entities, document categories and

[5] `http://www.alchemyapi.com/`

social tags. This semantic information is obtained using three NLP techniques:

- Named Entity Recognition - a subtask of information extraction that identifies names of persons, organizations etc. in a given text or sentence
- Document Categorization - a task that classifies the subject or topic of the text, and
- Social Tagging - the practice of generating tags or keywords by users rather than experts to classify and describe online content.

We employ a tool called OpenCalais[6] which can recognize up to 39 entities from the text. It also categorizes the text into one or more 18 document categories such as Finance, Entertainment etc. In addition, it associates one or more social or common sense tags with the text. The use of these techniques ensures that a large amount of world knowledge is exploited for feature extraction. For instance, consider the Facebook page for 'Pride and Prejudice'[7]. Some of the features generated from it:

- Entities (Type): Pride and Prejudice (Book), Elizabeth Bennet (Person), United Kingdom (Country)
- Document Category: Education
- Page Category: Book
- Social Tags: Film, Literature etc.

These features provide a valuable insight into a user's interests. Thus, if a user has liked this particular page, we can infer from the extracted features that the user is interested in books and literature. An important observation to note here is that if we use

[6] http://www.opencalais.com/ [7] http://on.fb.me/1BVX8Yo

only the Facebook page category i.e. 'Book', it can help infer the user's coarse-grained interest (in books). However, additional features such as tags further enable inferring his fine-grained interests in 'Literature' (a specific book genre).

8.5.2.4 Computing Normalized Item Weights

Each item is assigned a normalized Item Weight w_i (explained in the 'Selection and Tuning of design schemes and parameters' section) which represents its contribution, with respect to other items, to the user's interest profile. This normalization also ensures that the scores remain comparable across users.

8.5.2.5 Computing Semantic Relatedness Scores

For each item, we iterate over the UIP tree model (initialized from the User Interest Hierarchy with all node scores set to 0.0) and compute the Semantic Relatedness (SR) scores between each of the the item's features and the node category. Semantic Relatedness[8] [136] is a metric for determining similarity of two documents or phrases based on their semantic meaning. It is often normalized to a value between 0 (no relatedness) and 1 (extremely high relatedness). Harispe et al. [86] present a detailed discussion of various Semantic Relatedness metrics and techniques such as Latent Semantic Analysis (LSA). We employ the Semantic Textual Similarity (STS) system [105] for computing the SR scores. STS is based on LSA along with WordNet knowledge and is trained on LDC Gigawords and Stanford Webbase corpora.

The SR metric enables inferring implicit interests of users which they have not ex-

[8] Semantic Relatedness and Semantic Similarity have often been used interchangeably in literature. We use Semantic Relatedness to refer to the metric.

pressed explicitly (through liking or posting etc.) on Facebook. For instance, the semantic relatedness between 'Movies' and 'Cinema' is 0.9. Thus, if a user has expressed significant interest in the category 'Movies' by liking items pertaining to movies or by posting about movies, it is highly likely that her level of interest in going to 'Cinema' (categorized under 'Places for Recreation and Sightseeing') would be high.

For each node, we average the SR scores calculated between the item's features and the node category to compute the Average Semantic Relatedness (ASR) score for that node. As explained earlier, the ASR score is used to predict the likelihood of the user being interested in the node's category, given her level of interest in that item. If the ASR score for a node calculated from an item is below a certain threshold score, $SR_{threshold}$, we set it to 0.0. This helps in reducing noise and false positives. Since SR is a cosine similarity measure, a threshold of 0.293 (1 - cos 45°) is generally considered an appropriate threshold and we use that in our current implementation.

8.5.2.6 Computing Initial and Timed Weights for each node

Each node in a user's UIP tree has one 'Initial' weight (initialized as 0.0) and zero or more 'Timed' weights assigned to it. The Initial Weight represents the weight obtained from static items in the user's Facebook profile. The Timed Weights represent the weights obtained from dynamic items and also consist of the time at which the weights were generated i.e. the time at which the items were posted or updated in the user's Facebook profile. These Timed Weights signify decaying interests in the dynamic items with the decay being a function of time (decay scheme explained in the next section) that has

passed since the item was generated or updated. Intuitively, if a user has posted recently about a category, for instance 'Cricket', as opposed to a year back, her current level of interest in 'Cricket' would be high.

The ASR score between an item and a node is multiplied with the item's normalized weight to generate a Weighted Average Semantic Relatedness (WSR) score for that node. The WSR score calculated from each static item is compared with the current Initial weight of each node and the maximum of the two values is assigned as the node's current Initial weight. The WSR score calculated from each dynamic item is used to generate a <key,value> pair of Timed weight for each node where the timestamp of the item is the unique key. The generated Timed Weight is added to the existing set of Timed Weights for each node.

8.5.2.7 Propagating weights bottom up in the tree

The Initial and Timed weights assigned to children nodes are propagated bottom up in the UIP tree to their parent node (propagation scheme explained in the next section). The upward propagation of weights ensures that implicit high-level interests are inferred from explicit low-level interests that they subsume. For instance, if a user has explicitly expressed interest in a fine-grained node category such as 'Jazz Music', she has implicitly expressed interest in its parent node category 'Music'.

The Initial weight, propagated from the children nodes, is compared with the existing Initial weight of the parent node and the maximum of the two values is assigned as the parent node's current Initial weight. The propagated Timed weights are added to the

existing set of Timed weights of the parent node.

8.5.2.8 Discovering new category of interest

If the maximum ASR score between an item and any of the nodes in a user's UIP tree is $< \text{SR}_{threshold}$, it implies that the item is not related to any of the categories in the profile. Hence, we add that item as a new node in the user's UIP tree, thus, representing a newly discovered category of interest for that user.

8.5.2.9 Computing Effective Weight from Initial and Timed Weights

To compute an aggregated weight from the Initial and decayed Timed weights assigned to a node, we compute an 'Effective Weight' for it.

8.5.2.10 Computing Node Score from Effective Weight

Since the SR score is a non-linear score, the weights computed from it are non linear. In order to compare them with Likert scale scores, we map the effective weight for each node to a linear scale of 1-5 using the inverse cosine function, round it and assign it as the Node Score.

8.5.3 Selection and Tuning of design schemes and parameters

We experimented with various design parameters and schemes for Algorithm 5 on a validation dataset consisting of about 10% of the users from the entire dataset i.e. 50 randomly chosen users. We then tuned the parameters and schemes based on empirical per-

formance. The performance measure used was the Mean Absolute Error (MAE), which is one of the most commonly used metrics in the literature [113, 137]. We computed MAE over all nodes for all users in the validation dataset.

The Absolute Error for a node *j* in a UIP tree for a user *i* is computed as: $err_{i,j} = |s_{i,j}^{actual} - s_{i,j}^{estimated}|$ where $s_{i,j}^{actual}$ is the actual Likert scale score assigned by user *i* to node *j* and $s_{i,j}^{estimated}$ is the node score estimated by our algorithm for node *j* for user *i*. The MAE for the entire population of users is defined as:

$$\sum_{i=1}^{N} \frac{\frac{\sum_{j=1}^{n} err_{i,j}}{n}}{N}$$

where n is the number of nodes (203) and N is the number of users in the population.

8.5.3.1 Item Weight Normalization Scheme

We experimented with two schemes for item weight normalization for each item i:

1. Constant item weight scheme: The item weight is constant and same for each item i.e. $w_i = 1$.

2. Normalized item weight scheme: The item weight is normalized as,

$$w_i = \frac{\text{\# of similar items in a section of a user profile}}{\text{Total \# of items in that section of the user profile}}$$

   For instance, w_i for an item such as a liked movie page would be the ratio of the number of liked movie pages to the total number of liked pages. This is intuitive because if a user has liked 100 pages on Facebook and 50 of them are movie pages, her interest in movies is significantly high.

Scheme 1 has a MAE of 1.71 while Scheme 2 has a MAE of 1.28. Hence, we implemented scheme 2 in our system.

8.5.3.2 Upward Weight Propagation Scheme

We experimented with three schemes for propagating weights upwards:

1. No weight propagation - In this scheme, there is no bottom up weight propagation.
2. Average weight propagation - In this scheme, the averaged Initial and Timed weights of children nodes are propagated up to the parent node at all depths.
3. Average and maximum weight propagation - In this scheme, if the children nodes are leaves, then the maximum Initial and Timed weights of the children nodes are propagated up to the parent node. At all other depths, the averaged Initial and Timed weights of the children nodes are propagated upwards.

As stated earlier, the upward weight propagation ensures that implicit high-level interests are inferred from subsumed low-level interests. We observed that if a user expresses a significant interest in any of the fine-grained categories, say 'Jazz Music', she often expresses a significant interest in its parent coarse-grained category, 'Music'. Hence, we experimented with propagation of both average and maximum weights. Scheme 1 has a MAE of 1.68, Scheme 2 has a MAE of 1.41 and Scheme 3 has a MAE of 1.21. We have implemented Scheme 3.

8.5.3.3 Temporal Decay scheme and parameters

We experimented with two temporal decay schemes:

	No Decay	Decay with λ =				
		0.002	0.008	0.03	0.14	0.8
MAE	1.4	1.28	1.21	1.32	1.27	1.25

Table 8.2: MAE Comparison of Temporal Decay schemes

1. A no decay scheme - In this scheme, the Timed weights do not decay. The Effective Weight for each node is the maximum of its Initial and Timed weights.

2. An Exponential decay scheme - In this scheme, a decayed Timed weight obtained from item i is

$$TWDec_i = TW_i \times e^{-\lambda(T_{Current} - T_{LastUpdate})}$$

where TW_i is the Timed weight generated from item i, λ is the exponential decay constant, $T_{Current}$ is the time at which the interest score is being calculated and $T_{LastUpdate}$ is the time at which i was updated. The time difference is measured in days. The Effective weight for node n is calculated as

$$EW_n = IW_n \times \prod_{i=1}^{k}(1 - TWDec_i)$$

where IW_n is the Initial weight and k is the number of Timed weights assigned to node n.

Table 8.2 shows the MAE for the two schemes. For the Decay scheme, we varied the values [9] of λ. As shown, the no decay scheme performs worse than the exponential decay schemes, thus, validating our intuition that the timeline represent dynamic interests

[9] We experimented with several values of λ but have shown only a few.

of users that evolve with time. For the exponential decay scheme, $\lambda = 0.008$ has the lowest MAE of 1.21. Hence, we use that in our current implementation. Henceforth, all experimental results will be based on the aforementioned design choices.

8.5.4 Discussion

In order to apply this approach to other social media profiles, such as Twitter, the user's Twitter profile and content of tweets (including URLs etc.) will be utilized as items. These items will be presented as input to the UIP Generation Pipeline which will extract features (such as hashtags, entities, social tags and categories) and employ Algorithm 5 to build the user interest profile.

8.6 Challenges in processing and analysis of Facebook text data

Before we evaluate our algorithm, we first identify some of the challenges that we faced while processing and analyzing text data retrieved from a user's Facebook profile:

- As mentioned earlier, Facebook profile text is unstructured user generated content. Hence, it is noisy and unnormalized. Moreover, it can be grammatically incorrect, and can contain misspellings and slangs. Unlike Twitter, where research has been conducted in syntactic and lexical text normalization [138], Facebook text has not been studied extensively.

- A Facebook user's profile may have content in several languages in addition to English. Hence, most NLP techniques, that have been developed using English as a base language, cannot be directly applied to such multi-lingual content.

- As stated earlier, Facebook page categories are coarse-grained and flat. Moreover, they are not clearly documented and are often inconsistent. Multiple pages representing the same entity are categorized differently. For instance, a sportsperson can have a page with category 'Public figure' and another with category 'Athlete'. A page may have the generic category 'Interest' which doesn't indicate anything useful. Hence, relying solely on the page category feature does not lead to high interest prediction accuracy. The use of additional features such as social tags and document categories, as we have done, improves performance. This is demonstrated later in the Evaluation section.

- Facebook data is retrieved as a JSON object with no defined format. Hence, individual parsers have to be implemented to parse the text data from each section.

8.7 Experimental Evaluation

8.7.1 Methodology and Goals

The goal of this evaluation is to determine how accurately our system, and its underlying algorithm, can estimate a user's interest in a category. As mentioned earlier, we asked the Facebook users participating in our study to fill out a survey and enumerate their levels of interest in the 203 interest categories, in our User Interest Hierarchy, on a scale of 1-5 (1 = To a great extent, 2=Very much, 3=Somewhat, 4=Very little, 5 =Not at all). These values were used as *Ground Truth* for the users and were then compared with the User Interest Profiles generated by our system from their Facebook profiles. Figure 8.5 shows

partial views of the ground truth and generated UIP trees for a randomly chosen user.

8.7.2 Ground Truth Characteristics

The ground truth exhibits some interesting characteristics: The global average score assigned by all 488 users over all category nodes is 3.2. The average score for nodes at different depths increases as depth increases (2.6 at depth 1, 2.98 at depth 2, and 3.3 at depth 3). The most popular categories (average user score < 2.0) were:

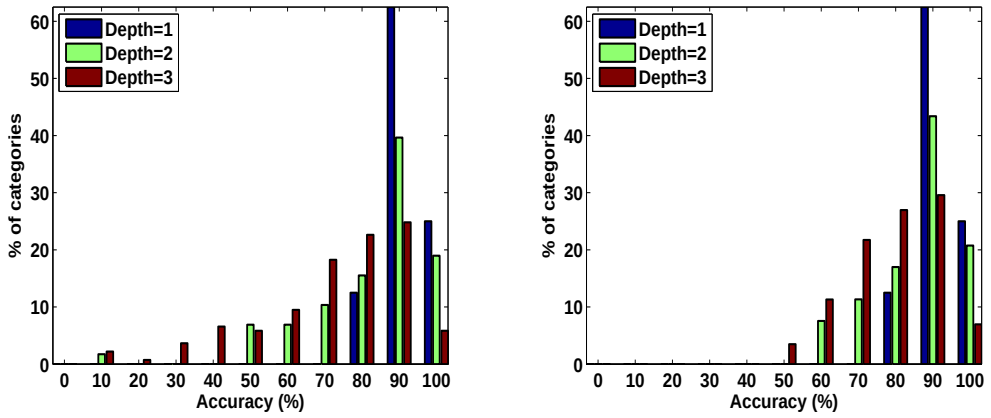
- Depth 1 - 2 categories ('Entertainment' and 'Pets')
- Depth 2 - 10 categories including 'Music', 'Television shows', 'Movies', 'Desserts', 'Fruits', 'Cats and Dogs'
- Depth 3 - 15 categories including 'Comedy Television shows' and 'Comedy movies'

Least popular categories (average user score > 4.0) were:

- Depth 2 - 5 categories including 'Landmarks and Historical Buildings' and 'Financial services'
- Depth 3 - 22 categories including 'Asian Music', 'Avant-Garde Music', 'Religious Television Shows', 'Religious Movies' and 'Chess'

8.7.3 Results

We evaluate the performance of our system, and proposed algorithm, on the entire dataset of 488 users using 3 standard performance measures that have been widely employed in existing literature [113, 137]:



(a) Interest prediction accuracy for all categories (b) Interest prediction accuracy for relevant categories

Figure 8.3: Histogram of interest prediction accuracy for all and relevant categories for all users

8.7.3.1 Interest Prediction Accuracy

This measure evaluates how accurately the proposed algorithm can estimate whether a user is interested in a category or not. As suggested by Herlocker et al. [139], we use the ground truth scores provided by the users to identify whether they are interested in a category or not. We divided the ground truth and estimated node scores into two classes 'Interested' and 'Not Interested' using a classification threshold. Since we use a 5 point Likert scale (which has also been employed in [139]), we set the threshold as t=3.0 as a score $<= 3.0$ represents some interest level. Thus, a node score $<= 3.0$ represents the positive class 'Interested' and a node score > 3.0 represents the negative class 'Not

Categories	Accuracy (%) for	
	all categories	relevant categories
Depth 1	90.19	90.19
Depth 2	81.67	85.53
Depth 3	71.72	78.85
All depths	75.29	81.2

Table 8.3: Accuracy for all and relevant category nodes at individual depths and all depths taken together

Interested'. Accuracy a is defined as

$$\frac{t_p + t_n}{t_p + t_n + f_p + f_n}$$

where t_p is number of true positives, t_n is number of true negatives, f_p is number of false positives and f_n is number of false negatives.

Since our interest categories have a hierarchical structure, we analyze the accuracy at individual depths as well as all depths taken together. Table 8.3 summarizes the accuracy (higher the better) of our algorithm for all categories at different depths and at all depths. Figure 8.3(a) shows the histogram of accuracy for Algorithm 5 for categories at different depths[10]. As evident, the algorithm performed very well when predicting a user's interests in high-level and coarse-grained categories (Depths 1 and 2) and reasonably well for fine-grained categories (Depth 3). Though a direct comparison with CatStream [130] is not possible because of the differences in our approach, dataset, and structure and number of

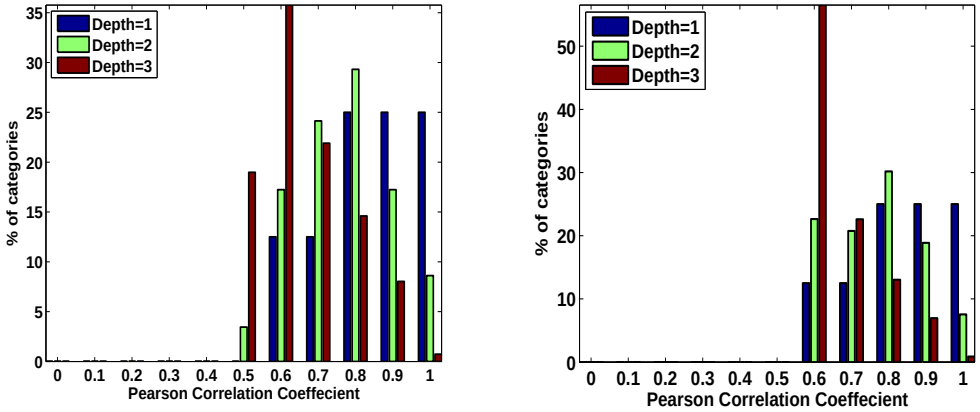
[10] Due to large number of categories in our hierarchy (203), it is not possible to show accuracy for each individual category

categories, our algorithm's overall accuracy is higher.

For most real world recommender systems, the primary concern is to accurately predict user interests in the *relevant* or *good* categories [113]. For our evaluation, we consider irrelevant categories as those which have a high average user assigned ground truth score (> 4.0) as this indicates a low average user population interest (See the Ground Truth Characteristics section for a discussion of popular and unpopular categories.). As mentioned earlier, there were no irrelevant categories at depth 1, 5 (out of 58) irrelevant categories at depth 2, and 22 (out of 137) irrelevant categories at depth 3. Hence, we remove the irrelevant categories and compute accuracy over the relevant categories only. Figure 8.3(b) shows the histogram for accuracy at different depths for only the relevant categories. Table 8.3 summarizes the accuracy for relevant categories at individual depths and at all depths. As evident, the accuracy of our algorithm is high for relevant categories. Thus, in practice, our system and algorithm can predict a user's interests accurately.

8.7.3.2 Pearson's Correlation Coefficient

Figure 8.4(a) presents the accuracy of predicting interest as expressed by the Pearson product-moment correlation coefficient (ρ) between the actual and estimated interest levels for all categories. As shown, more than 75% of depth 1 categories have $\rho > 0.75$ which indicates that the actual and estimated interest levels are highly correlated. At depth 2, 67% of the categories have $\rho > 0.7$ and at depth 3, 63.5% of the categories have $\rho > 0.6$. This too indicates that the algorithm has high accuracy when predicting interests in high-level and coarse-grained categories and reasonable accuracy for fine-grained



(a) Pearson Correlation coefficient between actual and predicted values for all categories

(b) Pearson Correlation coefficient between actual and predicted values for relevant categories

Figure 8.4: Interest Prediction accuracy expressed by the Pearson correlation coefficient between estimated and actual values, for all and relevant categories

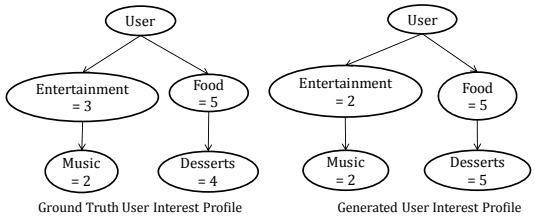


Figure 8.5: Partial view of Ground Truth and Generated UIPs for a randomly chosen user

categories, corroborating the results presented previously.

Similar to interest prediction accuracy measure, we computed the Pearson correlation coefficient for relevant categories. Figure 8.4(b) presents the accuracy at different depths for relevant categories only. As shown, more than 75% of depth 1 categories have $\rho >$ 0.75 which indicates that the actual and estimated interest levels are highly correlated. At depth 2, 70% of the categories have $\rho > 0.7$ and at depth 3, 70% of the categories have $\rho > 0.6$.

Categories	MAE for	
	all categories	relevant categories
Depth 1	0.88	0.88
Depth 2	0.91	0.89
Depth 3	1.0	0.93
All depths	0.97	0.91

Table 8.4: MAE for all and relevant category nodes at individual depths and all depths taken together

8.7.3.3 Mean Absolute Error (MAE)

The Absolute Error evaluates how accurately the proposed algorithm can estimate the level of interest of a user in a category. For instance, for the node with category 'Entertainment' in the ground truth and estimated profiles of a randomly chosen user shown in Figure 8.5, the absolute error will be $|\ 3 - 2\ | = 1.0$.

We use the definition of MAE mentioned earlier and compute it for all categories for all 488 users in our dataset. Table 8.4 shows the MAE (the lower the better) for categories at individual depths and all depths for all users. When considering only relevant categories, the MAE decreases even further. This too demonstrates that, for practical purposes, our system and algorithm can accurately predict a user's level of interests.

Categories	Algorithm 1	Baseline			
		1	2	3	4
Depth 1	**0.88**	1.12	1.17	1.2	1.5
Depth 2	**0.91**	1.21	1.27	1.33	1.66
Depth 3	**1.0**	1.26	1.32	1.59	1.77
All depths	**0.97**	1.24	1.35	1.49	1.74

Table 8.5: MAE Comparison of Algorithm 1 with baselines for all category nodes at individual depths and all depths taken together

8.7.4 Discussion

The MAE is higher and the accuracy is lower as the categories become more specific at each depth. A possible reason for this could be that even though our algorithm employs various techniques to infer fine-grained interests, we do not have enough information to estimate a user's levels of interest in all the fine-grained categories in our hierarchy. For instance, a user can be interested in various genres of books but may post only about a few. A hybrid approach that also employs collaborative filtering techniques (which exploit user or item similarity) can be used to overcome this limitation. Moreover, the challenges mentioned earlier (such as unnormalized text and multi-lingual content) highlight some of the difficulties involved in user interest profiling from Facebook data. We will address these limitations in future work.

8.7.5 Comparison with baselines

To put the accuracy of our proposed Algorithm 5 in perspective, we compare its performance with 4 baselines. These have been constructed using different sets of features or techniques to evaluate and demonstrate their impact on accuracy. These baselines are:

1. Baseline 1 - An algorithm similar to Algorithm 5 which utilizes Facebook page category and page name as the only features to generate the user's interest profile. Since only the liked pages in a user's Facebook profile have a page category, they are the only items used.

2. Baseline 2 - An algorithm similar to Algorithm 5 which performs lexical comparison (instead of using Semantic Relatedness) of each feature with the interest categories to generate the user's interest profile. Here, the SR score = 1 if a match is found and 0 otherwise.

3. Baseline 3 - An algorithm that assigns the global average user score for all categories i.e. 3.2 to each node in the UIP tree for a user.

4. Baseline 4 - An algorithm that selects a random value between 1 and 5 and assigns that as the score to each node in the UIP tree for a user.

Table 8.5 compares the MAE for Algorithm 1 and the baselines (lower MAE indicates superior performance) for all categories for all users in our dataset. As shown, Algorithm 5 outperforms all the baselines at all depths. It is evident from the higher overall MAE for Baseline 1 that even though Facebook page categories are valuable for inferring coarse-grained interests, they are not sufficient for inferring fine-grained interests. Additional

features such as social tags and document categories utilized by our Algorithm proved to be useful in this case and improved performance by almost 6.75% (on a scale of 1-5). Moreover, this also demonstrates that exploiting the temporally dynamic nature of the user's Facebook timeline is beneficial for augmenting her user interest profile and improving interest prediction accuracy. Similarly, we can see from the performance of Baseline 2 that even though lexical equality serves to identify explicit interests, Semantic Relatedness helps infer implicit interests as well which further enhances interest prediction for all categories at all depths by a margin of 9.5%.

Chapter 9: **Multi-Dimensional Collaborative Recommendations Using Tensor Factorization on Sparse User-Generated Data**

The users' sensed context, such as their location and activities, is aggregated over a period of time to generate their context history. The context history of several users can be leveraged to recommend new information to a particular user via collaborative filtering. Previous work has mainly explored recommending interesting locations; however, users would also benefit from recommendations for activities in which to participate at those locations along with suitable times and days. Thus, systems that provide collaborative recommendations involving multiple dimensions such as location, activities and time would enhance the overall experience of users. The relationship among these dimensions can be modeled by higher-order matrices called tensors which are then solved by tensor factorization. However, these tensors can be extremely sparse. In this chapter, we present a system and an approach [12] for performing multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data.

9.1 Introduction

Today's smartphones come equipped with a multitude of sensors such as GPS and increasingly-powerful computational, storage and communication capabilities. These features have

enabled smartphone applications to emerge across a variety of tourism-related areas such as travel recommendations, location-based services, and social suggestions. However, because information overload can be a problem for mobile users, it is important that only relevant and personalized information is presented. As a result, recommender systems that suggest items of interest to mobile users based on context and preferences have become increasingly popular.

Since the GPS embedded in smartphones can be used to accurately localize a user, *location* has become the basis of many recommender systems. These systems recommend interesting places or landmarks for visit to mobile users, particularly tourists and weekend travelers. However, in location recommendations, contextual information such as *time* often plays an important role as certain places are open only between fixed hours or can host certain activities at fixed times of a day or on fixed days in a week. In addition, users who want suggestions or recommendations for places to visit would also benefit from recommendations for *activities* to participate in at the location along with a suitable time of participation. Here, activities refer to human lifestyle and recreational activities such as shopping, dining, surfing etc. Users may even have specific *preferences* for activities such as preferring outdoor rather than indoor activities. For instance, at Pier 39 in San Francisco, a list of all possible things that a tourist can do includes shopping, eating seafood at the various restaurants, riding the Venetian carousel, watching the sea lions at the nearby dock or getting a view of the Fourth of July fireworks display. However, all these activities happen at specific locations in Pier 39 and either in specific months, on specific days or at specific hours. Thus, recommender systems that generate collaborative recommendations involving multiple dimensions such as location, time, activities, and

user preferences etc., would enhance the overall experience of users and provide them with the most helpful recommendations.

As a simplest formulation, recommender systems model preferences of *users* for *items* in the form of a *utility matrix* where rows represent users, columns represent items and the values represent the users' ratings or preferences for those items on a scale of say, 1 to 5. The goal of the recommender system is then to impute the missing values based on observed values in the matrix [113, 140] using a standard approach such as collaborative filtering. User-based collaborative filtering determines a subset of users most similar to the current user and predicts the missing ratings based on a weighted combination of the ratings provided by those other users [113]. Likewise, item-based collaborative filtering [141] focuses on predicting the missing ratings for items based on a weighted combination of the ratings given by the current user to similar items, where similarity between each pair of items is determined by the similarity of the ratings of those items provided by the users who have rated both items. Once the matrix has been completed (that is, all missing values have been imputed) by these neighborhood-based methods, a user can be given a recommendation list of ranked items ordered by descending predicted ratings.

An alternative powerful methodology that has been used with positive results in recommender systems is the latent factor model [142], an approach that emerged from research fueled by the Netflix prize[1]. Unlike neighborhood-based methods, latent factor models assume that similarity between users and items is simultaneously induced by some hidden lower-dimensional structure in the data. Some of the most successful re-

[1] `http://www.netflixprize.com/`

alizations of latent factor models are based on matrix factorization, where users and items are simultaneously represented as unknown feature vectors (column vectors) along k latent dimensions. These methods have become prominent in recent years because they combine scalability with high predictive accuracy. In addition, they offer more flexibility for modeling practical scenarios where the data is very sparse.

In case of recommendations along a single dimension, the 2-dimensional user $\times$ item matrix factorization model can be applied successfully. For instance, location recommendations can be modeled as a user $\times$ location matrix and can be solved using standard approaches to matrix factorization. However, to model collaborative multi-dimensional recommendations involving location, activity, time and other contextual information, the relationship among the various dimensions is represented by higher-order matrices called *tensors*. To address them, standard matrix factorization approaches need to be generalized to tensor factorization.

Another major challenge in recommender systems is that, in practice, most users provide interest ratings for only a subset of the recommended items. For instance, a user will provide ratings for only a subset of all possible locations. Moreover, as the number of dimensions increases, data sparsity increases and becomes a major concern for systems that generate multi-dimensional recommendations from real-world datasets.

In this work, we address these challenges and present a system and an approach for performing multi-dimensional collaborative recommendations for Who (User), What (Activity), When (Time) and Where (Location), using tensor factorization on sparse user-generated data. Our contributions are:

- We address n-dimensional collaborative recommendations

  (where $n \geq 4$ and includes user, location, activity, time and any other contextual information) by fusing data from multiple sources such as Flickr (a photo sharing website), Foursquare (a location based social network), Yelp (a crowd-sourced reviews based website) and Viator (a travel website). While our approach can be extended to any number of dimensions, we make the work in this chapter concrete by focusing on 4 dimensions and provide detailed discussion on how additional dimensionality can be addressed.

- While most of the prior efforts in multi-dimensional recommendations have been attempted using a single tensor, they suffer from tensor sparsity. We present a novel solution to this sparsity problem by formulating an objective function that simultaneously factorizes coupled tensors and matrices constructed from heterogeneous data sources. We then minimize this function using gradient descent.

- We evaluate our system and approach on large-scale real world data sets consisting of 588,000 Flickr photos collected from three major metro regions in the USA — the San Francisco Bay Area, Las Vegas and Chicago. From this data, we extracted over 4900 users, 6100 locations, 120 activities, and 96 time slots. The tensors constructed from these datasets are 99.999% sparse.

- We compare our approach with several state-of-the-art baselines and demonstrate that it outperforms all of them.

- Our approach also demonstrates an improvement in runtime without the need for

sacrificing performance.

The rest of the chapter is organized as follows: Section 9.2 reviews related work in location and activity recommendations and tensor factorizations applications to recommender systems. In Sections 9.3 and 9.4, we describe our datasets and the information inferred from them. We explain our approach in Section 9.5 and present its evaluation and comparison with baselines in Section 9.6.

9.2 Related Work

Since our work involves multi-dimensional recommendations for location, activity and time using tensor factorization on extremely sparse user-generated data, we have categorized the related work into different sections. The first section covers existing literature in location and activity recommendations while the second covers existing work in recommender systems using tensor factorization. We differentiate our work from them and identify their shortcomings.

9.2.1 Location and Activity recommendations

Most of the existing literature has focused on recommendations along one dimension (for instance, location). A few researchers have explored collaborative location and activity recommendations. To the best of our knowledge, none of the works have attempted collaborative location, activity and time recommendations.

9.2.1.1 Photos as a data source

One of our main data sources for user locations are Flickr[2] photos. Many of the photos uploaded by users are geotagged, thereby providing a wealth of geospatial data. These photos have been used for many purposes, such as finding Point-of-Interest (POI) clusters [143], identifying the location of photos from visual, textual, and temporal features [144], determining when tourism is in season [145], and creating routes that are pleasing to the user [146].

9.2.1.2 Location/Landmark/Venue recommendations

Previous researchers have investigated the problem of making recommendations for geolocations that may be interesting to the user [147–149]. Such approaches use large bodies of collected geospatial data along with user preferences and then apply low-dimensional recommendation algorithms.

9.2.1.3 Activity Recommendations

Even though users seek recommendations on what activities they can engage in when they visit a place or at a given time, the area of activity recommendation has not been researched extensively. Recent research has focused on diurnal activity recognition from smartphone sensory data [46] or from location [150]. Belotti et al. [151] explored the idea of serendipitous activity based discovery of venues and activities via the Magitti Mobile Leisure Guide. Ducheneaut et al. [152] experiment with several models such as

[2] `https://www.flickr.com/`

collaborative filtering, preference-based, distance-based, and a weighted combination of these to provide activity recommendations via Magitti.

9.2.1.4 Collaborative Location and Activity recommendations

Co-occurring location and activity collaborative recommendations were proposed by Zheng et al. in [153] where they addressed location recommendations given an activity, and activity recommendations given a location. They used Collective Matrix Factorization (CMF) [154] to complete a sparsely populated 2-dimensional Location $\times$ Activity matrix and evaluated their approach on 162 users with recommendations for 5 activities. CMF takes advantage of correlations and sharing of information between data sets from multiple sources and simultaneously factorizes coupled matrices. It is shown to have achieved higher prediction accuracy than individual matrix factorization. Sattari et al. [155] used the same dataset as Zheng et al. in [153] but employed Singular Value Decomposition (SVD) in place of CMF to complete the Location $\times$ Activity matrix. Since SVD requires the matrix that needs to be decomposed to be fully populated, they padded it with zero values. They demonstrated an improvement in performance over [153].

Zheng et al. [156] modeled user, location and activity data as a 3-dimensional tensor and applied a regularized tensor and matrix factorization approach for location and activity recommendations. They formulated a CANDECOMP/PARAFAC (CP) [157, 158] decomposition style objective function and minimized it using gradient descent. This decomposition factorizes a tensor into a linear combination of component rank-one tensors. They evaluated their approach on a dataset of 164 users, 168 locations and 5 activities.

Our work stands out from the existing literature in several ways:

- We use large user-generated datasets which involves several thousand users and locations, and over a hundred popular lifestyle, recreational and tourist activities.
- We incorporate dimensions of user's context such as time in addition to location and activities.
- As opposed to the manual approach for activity inference from user comments that was employed in [153, 156], we propose an automated and unsupervised Natural Language Processing (NLP) based algorithm (Section 9.4.2) which is more robust and scalable to large real-world datasets.
- Moreover, a major limitation of using CP decomposition for tensors is that it is not suitable for very sparse tensors and demonstrates a sharp increase in error especially if more than 80% of the data is missing [159, 160]. Sparsity is a non trivial issue for us as our multi-dimensional data is 99.999% sparse. Hence, we propose our joint analysis and factorization based approach to solve the problem.

9.2.2 Other applications of Tensor Factorization to Recommender Systems

Symeonidis et al. [161] and Nanopoulos [162] applied Higher Order Singular Value Decomposition (HOSVD) [163] to a 3rd order tensor which represents users, items and tags in social tagging systems such as Last.fm and Bibsonomy. Karatzoglou et al. [164] address multi-dimensional recommendations by incorporating contextual information to

model a User-Item-Context tensor. They utilize a sparse HOSVD style method that decomposes a D dimensional sparse tensor into D matrices and a D dimensional tensor. HOSVD is a generalization of the matrix SVD to a tensor. It assumes a dense tensor and is not suitable for very sparsely populated tensors. Moreover, unlike SVD, HOSVD may not provide the best low rank approximation of a tensor [165].

Hidasi and Tikk [166] apply an Alternating Least Squares (ALS) [165, 167] based tensor factorization approach for context-aware recommendations. ALS consists of three steps, each one being a conditional update of one of the factor matrices, given the others. However, it suffers from several drawbacks: It has poor convergence for sparse data [168] and is not scalable to large-scale data sets [159].

To address these limitations and perform multi-dimensional recommendations on large-scale and sparse user-generated datasets, we formulate our recommendation model via the Coupled Matrix and Tensor Factorization (CMTF) framework [160, 169]. CMTF is an approach similar to CMF and proposes joint analysis of a matrix and an N^{th}-order tensor with a common mode or dimension, where the tensor is factorized using an R-component CP model and the matrix is factorized by extracting R factors using matrix factorization. CMTF is shown to have achieved better performance than standard CP decomposition especially if more than 80% of the data is missing [159, 160]. A variant of the CMTF approach [159, 169] performs the joint analysis of the tensor and matrix by ignoring the missing entries and fitting the tensor and/or the matrix model to the known data entries only. This approach has been shown to easily scale to handle very large data sets with up to 99% missing entries [159].

To this end, we model our multi-dimensional recommendation problem as a joint

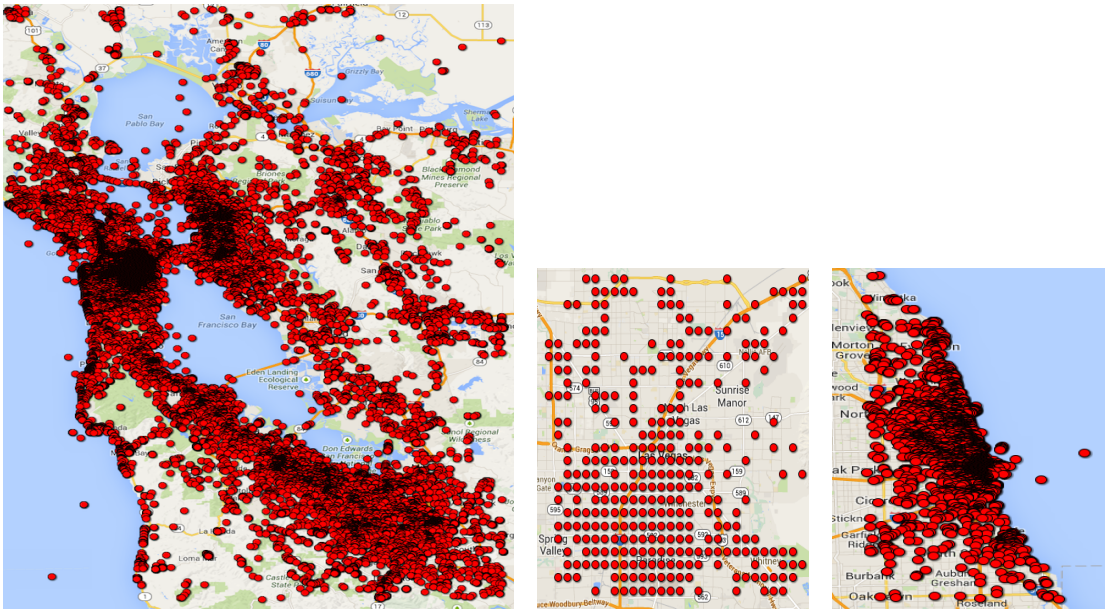
Data Source	Type of Data	Dimensions extracted	Volume of raw data extracted
Flickr	Geotagged and Timestamped Photos	User, Location, Activity, Time	588,000 photos with 9 million words of text
Foursquare	Location and POI database	Location, Activity, Venue	274,000 locations with POI or venue information
Yelp	Business and service reviews	Activity	1060 service categories
Viator	Things to do in tourist spots	Activity	60 things to do categories

Table 9.1: Data fusion from heterogeneous data sources

analysis of a sparsely populated tensor with several matrices which share one or more common modes with the tensor. These tensors and matrices are constructed by fusing data for the various dimensions (users, locations, activities and time) from multiple data sources. This is a challenging task since data sets are often incomplete and heterogeneous. We then factorize these tensors and matrices simultaneously using a gradient descent-based algorithm. As we show later in Section 9.6.5, our approach outperforms standard CP decomposition, HOSVD and ALS.

9.3 Data

Table 9.1 summarizes the data sources, types of data, extracted dimensions and the volume of raw data from each source. Our primary dataset consists of 588,000 geotagged and timestamped publicly-available photos from Flickr.com, a popular photo-sharing website hosted in the USA. To obtain the data set, we searched specifically for photos taken with smartphones so that we could get the most accurate geospatial traces. Further, these photos spanned the time period of September 1, 2009, to September 1, 2013 and were taken from three major metro regions in the USA – the San Francisco Bay Area, Las Vegas and Chicago. Table 9.2 shows the characteristics of this dataset, while Figure 9.1 shows the



(a) San Francisco Bay Area (b) Las Vegas (c) Chicago

Figure 9.1: Geographical distribution of photos from San Francisco Bay Area, Las Vegas and Chicago core regions. Map image tiles were provided by Google, and waypoint placement was performed using `http://www.gpsvisualizer.com/`.

geographical distribution of the photographs over the three regions. To improve legibility of the figure so that individual photo locations can be discerned, we sampled the number of photos down.

We extracted photos and their meta-content in JSON format using Flickr's public REST based API. Each photograph is marked with the user ID of the user who took it, geo-location in latitude and longitude format representing where it was taken, and an epoch timestamp representing when it was taken. We also extracted about 9 million words of user-generated text (including title, description, tags or comments) for these photos. Note that less than 50% of the photos are marked with any text. In addition, we also obtain POI or Venue information for 274,000 locations in the three geographical regions

Region	Photos	Users	Land Area (km^2)
San Francisco Bay Area	280,045	7895	6053
Las Vegas	23,350	578	1818
Chicago	284,751	2423	1412

Table 9.2: Flickr.com raw dataset characteristics

from Foursquare[3]. Finally, we obtained information about popular activities from Yelp[4] and things to do in tourist spots from Viator[5]. We explain the inference and extraction of the various dimensions from the collected data in Sections 9.4 and 9.5.

9.4 Inferring various dimensions of information from Flickr photos

As mentioned earlier, photos on Flickr have meta-data that includes a user ID, timestamp, location and text. We now infer the dimensions of user, location, activity and time from this meta-data.

9.4.1 Location Hashing

Our system relies on unique and discrete locations, but the Flickr geotags are stated as continuous floating-point latitude and longitude geocoordinate pairs. To discretize these values, we applied Cartographic Sparse Hashing, our O(1) algorithm (shown in Algorithm 6) that hashes a latitude and longitude pair into one of many virtual rectangular grid bins formed throughout the geocoordinate space. This algorithm takes as input (i) latitude and longitude as 64-bit floats and (ii) a bin resolution size r in meters as an integer. It then

[3] www.foursquare.com/ [4] www.yelp.com/ [5] www.viator.com/

Algorithm 6: Cartographic Sparse Hashing (CASH) algorithm

Input: $latitude$ as 64-bit float, $longitude$ as 64-bit float, grid resolution r in meters

Output: Hash value $hashResult$ as 64-bit integer

Trim $latitude$ digits past 5th decimal position;

$sigFig \leftarrow 10^5$;

$latitudeInt \leftarrow (int)(latitude \times sigFig)$;

Round down $latitudeInt$ to be divisible by r;

Repeat with $longitude$ to produce $longitudeInt$;

$hashResult \leftarrow shiftHigh(longitudeInt) + latitudeInt$;

return $hashResult$

outputs a 64-bit integer key representing the final virtual bin. The algorithm leverages the fact that the latitude and longitude are expressed in decimal degrees, with the fifth decimal place corresponding roughly to 1 meter. Since this precision was acceptable to us, we truncated each value to five decimal places. The function then produces the resulting integer key with the longitude and latitude ending up in the high and low bits, respectively. This key identifies a virtual bin approximately *r* meters per side, although the bin will be elongated north-to-south for regions further away from the equator due to the Earth's curvature.

We experimented with 4 different location grid sizes - 300m, 500m, 700m and 1000m. These were determined based on the venue density in each grid as well as human walking distance (as people may often prefer to walk or use public transport). If the grid size is too large, recommendations beyond a certain walking distance will not be helpful to the user. On the other hand, if the grid size is too small, there may not be any venue or

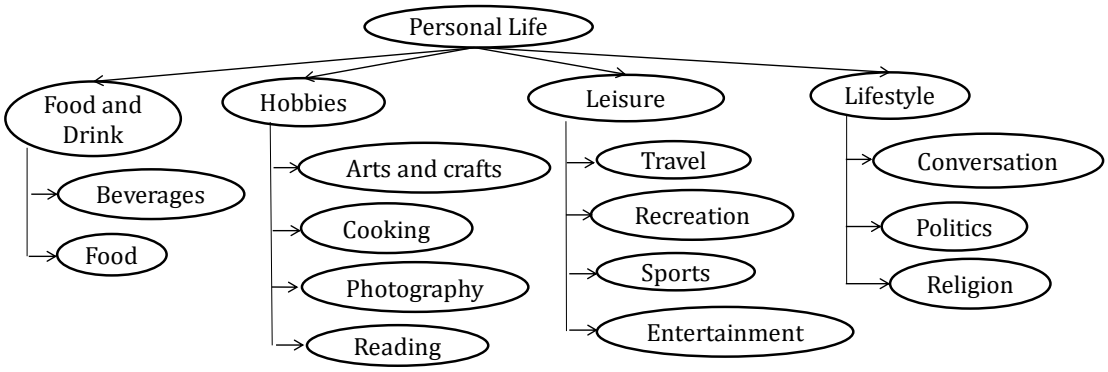


Figure 9.2: Partial view of the Activity Hierarchy showing all depth 1 nodes and a few depth 2 nodes

place to recommend in the grid. We explain the selection of the appropriate size based on performance in Section 9.6.3.

9.4.2 Activity Inference

Previous research in activity recognition [46, 150] focused on recognizing daily activities such as 'walking', 'driving', 'biking' etc. from smartphone sensors such as the accelerometer and GPS. However, a real world recommender system should be capable of recommending a diverse set of activities to users in addition to such diurnal activities. To address this, we employ an activity hierarchy which consists of lifestyle, recreational and tourist activities for activity recommendations.

Figure 9.2 shows a partial view of our Activity Hierarchy. We generated it by manually combining popular activities from the Yelp category list, Viator things to do list, and the FourSquare venue category hierarchy. Our hierarchy has a tree structure of depth 4 and contains 120 activity nodes including the root node 'Personal Life'. There are 4 high-level (depth 1) activities which branch out into 15 coarse-grained activities (depth 2).

All the high-level and some of the coarse-grained activities that they include are shown in Figure 9.2. Some of the coarse-grained activities at depth 2 are further categorized into 28 fine-grained activities (depth 3) which are further categorized into 128 leaf node activities. For instance, the root node 'Personal Life' branches out into coarse-grained activities including 'Leisure' which is further categorized into fine-grained activities such as 'Entertainment' and Recreation' etc. 'Entertainment' is further categorized into leaf node activities such as 'Music', 'Movies' and 'Dance'. It is possible that a user is engaged in an activity which is not present in our hierarchy. To address this, we have a generic category 'Other' at each depth.

Inferring users' activities from our data set is non-trivial and challenging. Unlike previous research, we do not have labeled sensory data from users' phones. Though all the photos in our dataset are geotagged, a user could be engaged in several probable activities at a location. Also, less than 50% of the photographs are annotated with user-generated photo content such as a detailed name or a description, and even fewer photographs have tags or comments which can provide some indication of the activity occurring when the photograph was taken. In addition, crowd-sourcing the labeling of activities (as done in [153, 156]) is not feasible for 588,000 photos.

To address these challenges, we propose an automated and unsupervised NLP based algorithm (Algorithm 7) to infer a user's activity from user-generated text such as the photo content items. As shown, we first remove all stop words from each photo content item and concatenate the items. We then perform a web search query to elucidate the meaning of the concatenated items and retrieve the content of the top-most web search result. From this content, we extract features such as named entities, document categories

Algorithm 7: Activity inference from user-generated text

Input: Photo content items such as name, description, tag and comments and Activity Hierarchy

Output: Inferred Activities

Remove stop words from each photo content item;

Concatenate the content items to generate a search query;

Perform a web search using the search query and retrieve the text content of the top-most web search result;

Extract features such as named entities and types, document categories and social tags from the text content;

foreach *Activity in the Activity Hierarchy* **do**

    Compute SR scores between the features and the Activity;

    MaxSRScore for each activity $\leftarrow \arg\max$ (SR Score between any of the features and the activity);

    if *MaxSRScore* $< SR_{threshold}$ **then**

        MaxSRScore $\leftarrow$ 0.0;

end

MAXMaxSRScore $\leftarrow \arg\max$ (MaxSRScore);

if *MAXMaxSRScore* $\neq$ *0* **then**

    MaxActivities $\leftarrow$ Activities with MAXMaxSRScore;

if *MaxActivities* = $\emptyset$ **then**

    MaxActivities $\leftarrow$ Propagated photo labels based on distance and time;

return *MaxActivities;*

and tags. These features are extracted using three NLP techniques:

- Named Entity Recognition - a subtask of information extraction that identifies names of persons, organizations etc. in a given text or sentence
- Document Categorization - a task that classifies the subject or topic of the text, and
- Social Tagging - the practice of generating tags or keywords by users rather than experts to describe online content.

For this feature extraction, we employ a tool called OpenCalais[6] which can recognize up to 39 entities from the text. It also categorizes the text into one or more 18 document categories such as Finance, Entertainment etc. In addition, it associates one or more social tags with it. The use of these techniques ensures that a large amount of world knowledge is exploited for feature extraction.

We then compute the Semantic Relatedness (SR) scores between each activity and each feature extracted from the web content. SR [86] is a metric for determining the similarity of two documents or phrases based on their semantic meaning. It is normalized to a value between 0 (little to no relatedness) and 1 (extremely high relatedness). While there are several techniques and systems available for computing SR, we employ the Semantic Textual Similarity (STS) system [105] for computing SR scores. STS is based on Latent Semantic Analysis (LSA) along with WordNet knowledge and is trained on the LDC Gigawords and Stanford Webbase corpora.

For each activity, we store the maximum SR score (MaxSRScore) between any of the features and the activity. If the MaxSRScore for an activity is less than a threshold, we set

[6] `http://www.opencalais.com/`

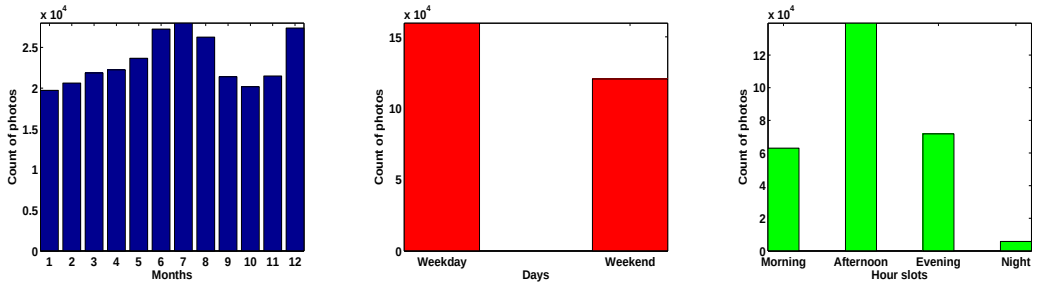
it to 0.0, thereby reducing noise and false positives. Since SR is a cosine similarity measure, a threshold of 0.293 (1 - cos 45°) is generally considered an appropriate threshold, and we use that in our current implementation. Finally, we iterate over all the activities and select those with the highest MaxSR scores. If a photo has no labeled activities, we apply a simple label propagation technique to label it based on its nearest neighbors (with respect to location and time). This approach is intuitive because if two consecutive photos are close in time and location, it is highly probable that the user was performing the same activity in both.

To illustrate Algorithm 7 better, consider a photo which has been tagged as 'Brazen Wildcat Half' by the user. This phrase by itself does not convey any meaningful information about the user's activity, but a web search for it reveals content such as 'Brazen Wildcat Half Marathon Racing...'. From the web content text, we get features such as 'Trail Run', 'Athletics' and 'Recreation' etc. Algorithm 7 then maps these features to the activities: Sports, Running and Recreation. From this, we can infer that the user was engaged in these activities when the photo was taken.

9.4.3 Time hashing

Since each photo has a unique timestamp, the number of timestamps in our dataset is huge. To address this, we perform feature hashing to hash the value of each timestamp to a timeslot. In order to determine the granularity of timeslots, we analyzed the monthly, weekly, daily and hourly distribution of photographs.

Figure 9.3(a) shows the distribution of photographs in the San Francisco Bay Area



(a) Monthly distribution (b) Weekly distribution (c) Hourly distribution

Figure 9.3: Monthly, Weekly and Hourly distribution of photographs in the San Francisco Bay Area dataset

dataset for months in a year (1= January until 12 = December). As evident, the number of photos varies with each month. Clearly, July and December have the highest photo counts as these months have the typical vacations of July 4th and Christmas. We also analyzed the differences in weekdays and weekends. As shown in Figure 9.3(b), weekdays have a higher photo count.

We further considered the distribution of photographs at different hours in a day. To this end, we divided a day into 4 hourly slots:

- Morning - hours between 6 am and 12 pm
- Afternoon - hours between 12 pm and 6 pm
- Evening - hours between 6 pm and 12 am
- Night - hours between 12 am and 6 am

As seen in Figure 9.3(c), the highest count of photographs is taken in the afternoon. Intuitively, the least number (5834) were taken at night. Based on this analysis, we gen-

Grid Size *r* (m)	# of unique hashed locations		
	San Francisco	**Las Vegas**	**Chicago**
300	7869	2747	5082
500	5565	1932	3665
700	4222	1475	2855
1000	2999	1058	2070

Table 9.3: Number of unique hashed locations in our 3 datasets

erated buckets of hashed time slots. Since there are 12 months in a year, 2 types of days in a week - weekday and weekend, and each day has 4 hourly slots, the total number of hashed timeslots is 96. For each photo, we first convert its epoch timestamp to a standard date time format which is then hashed to a timeslot. For instance, a photo taken on August 19, 2013 at 2 pm will be hashed to 'WeekDay_8_Afternoon'.

9.5 Our Joint analysis and factorization based Approach

9.5.1 Data Modeling of various dimensions

After inferring the user, location, activity and time information from the Flickr photos, the data in each dataset needs to be modeled along the various dimensions of our multi-dimensional tensor. These dimensions are:

- Location dimension - We first apply data filtering to reduce noise. We retain only those unique hashed locations that have been visited by at least u_t users (where u_t

= 5), Thus, for each unique hashed location present in our 3 datasets, we determine the number of unique users from that dataset who visited that location. We remove all locations that have ≤ 5 unique users. We perform this thresholding mainly to eliminate locations that can be residences of users or random spots such as roads. Table 9.3 shows the final count of locations in our datasets. The number of locations for each dataset varies with the hashed grid size, where we note that we discern at least 6127 locations in all with the largest grid size (1000m). These correspond to the Location dimension of our tensor.

- User dimension - Once the final set of locations for each dataset has been obtained, we use only those photographs which have been taken at these locations. The corresponding user IDs for these photographs represent the User dimension of our tensor. Our final set of users consists of 2498 unique users in the San Francisco Bay Area, 573 unique users in Las Vegas and 1850 unique users in Chicago.

- Activity dimension - The 120 activities in our activity hierarchy represent the Activity dimension of our tensor.

- Time dimension - The 96 hashed time slots represent the Time dimension of our tensor.

9.5.2 Constructing the tensors and matrices

We now construct a 4 dimensional tensor from these dimensions.

9.5.2.1 User × Location × Activity × Time tensor

The four dimensional (User, Location, Activity and Time) data modeled from each of the Flickr datasets can be represented as a sparse tensor $\mathrm{X} \in \mathbb{R}^{u \times l \times a \times t}$ where u is # of users, l is # of locations, a is # of activities and t is # of time slots. The ratings placed into this tensor should represent the user's interest for performing a certain activity at a certain location at a certain time. However, in our data set, users do not provide any explicit ratings. Hence, we derive an implicit feedback [170] value normalized over [0.0, 1.0]. Each cell value of the tensor represents the frequency of the current user being at the current location performing the current activity at the current time. The counts are further normalized based on the total number of data points (photographs) for each user.

Moreover, each user typically visits a small subset of the possible locations at only a few of all the possible times and performs a fraction of all the activities possible. Hence, our tensor is very sparse and any given fiber will have only a few non-empty entries. After this construction, the tensors from the 3 datasets have a density of the order of $10^{-3}\%$. Thus, the tensors are 99.999% sparse.

To supplement the sparse 4 dimensional tensor, we further construct various 2-dimensional matrices which are *coupled* with the tensor i.e. they involve one or more of these dimensions and thus, share at least one mode in common with the tensor. We construct them using data obtained from various other data sources.

9.5.2.2 Location × Activity Matrix

Knowing what activities occur in a given location can enable inference of the activity a user is engaged in when he is at the location. As mentioned earlier, even though all the photographs have a geolocation, more than 50% of the photographs do not have any photo content to indicate what activity the user could be engaged in. Hence, this location-activity information can enable inference of the most likely activity a user could be engaged in at a location.

To obtain this relationship for each activity in our hierarchy, we query the Foursquare location database to find all the locations, in each of our datasets, where that particular activity can occur. Thus, for each location l_i in a dataset, we get an n-dimensional frequency vector c_i = [c_1, c_2 ... c_n] for n activities (where n = 120). Each $c_{i,j}$ is normalized as $\frac{c_{i,j}}{\sum_{j=1}^{n} c_{i,j}}$. From this information, we construct a Location × Activity matrix $Y \in \mathbb{R}^{l \times a}$ which contains normalized counts for the activities that occur in each location.

9.5.2.3 Location × Venue Matrix

Activities typically occur at a venue or a POI such as a restaurant, shopping mall, etc. For instance, a user would 'Eat' at a 'Restaurant' or 'Shop' at a 'Shopping Mall' in a location. Hence, the knowledge of venues that are present in a location can be leveraged to enable the inference of the user's activity.

For each location in each of our datasets, we also obtain the counts of different venues (from the Foursquare location database) that are present in it. The venues belong to the Foursquare venue hierarchy that includes 470 different types of venues such as restau-

rants, movie theaters etc. Thus, for each location l_i in a dataset, we get an m-dimensional frequency vector v_i = [v_1 ... v_m] for m venues (m = 470). Each $v_{i,j}$ is normalized as $\frac{v_{i,j}}{\sum_{j=1}^{m} v_{i,j}}$. From this information, we construct a Location $\times$ Venue matrix J $\in \mathbb{R}^{l \times v}$ which contains normalized counts for venues in each location.

9.5.2.4 Location $\times$ Location Similarity Matrix

The locations that have similar type and count of venues will host similar activities. Hence, we employ the Location $\times$ Venue matrix to compute the location similarity information. For each pair of locations l_i and l_j in each dataset, we calculate the cosine similarity between the venues vectors as sim(l_i,l_j) = $\frac{v_i \cdot v_j}{\|v_i\| \|v_j\|}$ where $0 \leq$ sim(l_i,l_j) ≤ 1. Using the location similarity information, we construct a symmetric Location $\times$ Location matrix Z $\in \mathbb{R}^{l \times l}$.

9.5.2.5 Activity $\times$ Activity Correlation Matrix

Users may often have preferences for activities that are similar and correlated. For instance, a user who likes sports might engage in several different types of outdoor sports such as basketball, tennis, etc. Hence, this knowledge of correlation between activities can be exploited easily to further boost the information about the kinds of activities a user could be interested in.

We use the SR metric (from Section 9.4.2) to compute correlation between all the activities present in our hierarchy. For each pair of activities a_i and a_j in our Activity hierarchy, we calculate the SR score between their descriptions to get sim(a_i,a_j). For instance,

SR score between 'Sailing' and 'Surfing' is 0.63, indicating that they are correlated. This is intuitive as both are water sports. From this correlation information, we construct a symmetric Activity $\times$ Activity matrix S $\in \mathbb{R}^{a \times a}$.

9.5.3 Objective function formulation

We now perform joint analysis of the constructed tensor and matrices. We formulate an objective function that simultaneously factorizes the main 4-dimensional tensor and the four 2-dimensional coupled matrices that contain additional information. The tensor is factorized using a CP model while the matrices are factorized using matrix factorization. As discussed earlier, such an approach achieves better performance than standard CP decomposition for extremely sparse tensors. The objective function for our multi-dimensional recommendation problem is:

$$F = \frac{1}{2}\|W \times (X - U \circ L \circ A \circ T)\|^2 + \frac{\lambda_1}{2}\|Y - LA^\top\|^2 + \frac{\lambda_2}{2}\|S - AA^\top\|^2 + \frac{\lambda_3}{2}\|Z - LL^\top\|^2 + \frac{\lambda_4}{2}\|TR\|^2 + \frac{\lambda_5}{2}(\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$$

Thus, the objective function comprises six summands and can be written in the form F = F_1 + F_2 + F_3 + F_4 + F_5 + F_6. The six summands and the terms and symbols that they include are:

- F_1 - The weighted least squares error term for the decomposition of the 4 dimensional tensor X into the factor matrices U $\in \mathbb{R}^{u \times k}$, L $\in \mathbb{R}^{l \times k}$, A $\in \mathbb{R}^{a \times k}$ and T $\in \mathbb{R}^{t \times k}$ where *k* is the number of factors. W is a weight tensor $\in \mathbb{R}^{u \times l \times a \times t}$ and indicates the missing entries in X such that:

$$w_{u,l,a,t} = \begin{cases} 1 & \text{if } x_{u,l,a,t} \text{ is known} \\ 0 & \text{if } x_{u,l,a,t} \text{ is missing} \end{cases}$$

This term tries to minimize the loss in only the known entries of the tensor.

- F_2 - The least squares error term for the decomposition of the 2-dimensional matrix Y (containing the Location - Activity information) into factor matrices L $\in \mathbb{R}^{l \times k}$ and A $\in \mathbb{R}^{a \times k}$.

- F_3 - The least squares error term for the decomposition of the 2-dimensional symmetric matrix S (containing the Activity - Activity correlation information) into the factor matrices A $\in \mathbb{R}^{a \times k}$ and its transpose $A^\top \in \mathbb{R}^{k \times a}$.

- F_4- The least squares error term for the decomposition of the 2-dimensional symmetric matrix Z (containing the Location - Location correlation information) into the factor matrices L $\in \mathbb{R}^{l \times k}$ and its transpose $L^\top \in \mathbb{R}^{k \times l}$.

- F_5 - Regularization term for temporal smoothing. It leverages the fact that human behavior in successive time periods will be similar and will have a gradual variation. Hence, it tries to reduce the error between consecutive time slots. R is a bi-diagonal matrix $\in \mathbb{R}^{k \times k}$ with 1 on the main diagonal and -1 on the diagonal above it.

- F_6 - Regularization term to avoid overfitting.

- λ_1 - λ_5 are model parameters.

- $||\,||^2$ denotes the Frobenius norm, $\circ$ denotes the outer product

In general, there is no closed form solution for F, so we use numerical methods, such as gradient descent, to solve this problem. By using the representations in [167], we take the first order derivatives of F with respect to each of the factors to get the following:

∇_UF= ($\mathrm{W}^{(1)}$ - $\mathrm{X}^{(1)}$)(T $*$ A $*$ L) + λ_5U

∇_LF= ($\mathrm{W}^{(2)}$ - $\mathrm{X}^{(2)}$)(T $*$ A $*$ U) + λ_1($\mathrm{LA}^\top$ - Y)A + λ_3(- Z - $\mathrm{Z}^\top$ + 2L$\mathrm{L}^\top$)L + λ_5L

∇_AF= ($\mathrm{W}^{(3)}$ - $\mathrm{X}^{(3)}$)(T $*$ L $*$ U) + λ_1($\mathrm{LA}^\top$ - Y)$^\top$L + λ_2(- S - $\mathrm{S}^\top$ + 2A$\mathrm{A}^\top$)A + λ_5A

∇_TF= ($\mathrm{W}^{(4)}$ - $\mathrm{X}^{(4)}$)(A $*$ L $*$ U) + λ_4TR + λ_5T

where $\mathrm{W}^{(i)}$ and $\mathrm{X}^{(i)}$ denotes the mode-i tensor unfolding or matricization [7] of W and X such that $\mathrm{W}^{(1)}$ and $\mathrm{X}^{(1)} \in \mathbb{R}^{u \times lat}$, $\mathrm{W}^{(2)}$ and $\mathrm{X}^{(2)} \in \mathbb{R}^{l \times uat}$, $\mathrm{W}^{(3)}$ and $\mathrm{X}^{(3)} \in \mathbb{R}^{a \times ult}$, and $\mathrm{W}^{(4)}$ and $\mathrm{X}^{(4)} \in \mathbb{R}^{t \times ula}$, and $*$ denotes the Khatri-Rao product.

9.5.4 Minimizing the objective function

We employ Algorithm 8, which uses gradient descent, to minimize F and its gradient G. We implemented it in MATLAB using the Tensor Toolbox [171]. As input, the algorithm takes the incomplete sparse tensor (X), the low dimensional matrices (Y, Z, and S), the number of factors (*k*) as well as the stopping or convergence criteria. The individual components (U, L, A, T) are initialized using the n-mode singular vectors of X which span the subspace of the mode-n fibers i.e. the left singular vectors of the n-mode matricization of X. In each iteration, we first compute the step length using the More-Thuente line-search method [172]. We then compute the values of the gradients for the objective function and the components, and update the objective function value by taking a step in the direction of the gradient. The convergence criteria are set as:

[7] Matricization is generation of the matrix representation of a tensor in which all column or row matrices are stacked one after another.

Algorithm 8: Gradient descent based algorithm

Input: Sparse tensor X$\in \mathbb{R}^{u\times l\times a\times t}$, 2 D matrices Y $\in \mathbb{R}^{l\times a}$, Z $\in \mathbb{R}^{l\times l}$, S $\in \mathbb{R}^{a\times a}$,

k and convergence criteria

Output: Complete tensor M $\in \mathbb{R}^{u\times l\times a\times t}$

for *n = 1:size(X)* **do**

Initialize U, L, A, T;

end

Initialize F, ∇_UF, ∇_LF, ∇_AF and ∇_TF;

Set *i* = 0;

while *not converged* **do**

Compute step length α_i;

Compute the gradients $\nabla_U F_i$, $\nabla_L F_i$, $\nabla_A F_i$, $\nabla_T F_i$;

$U_{i+1} = U_i - \alpha_i \nabla_U F_i$, $L_{i+1} = L_i - \alpha_i \nabla_L F_i$, $A_{i+1} = A_i - \alpha_i \nabla_A F_i$, $T_{i+1} = T_i - \alpha_i \nabla_T F_i$;

Compute F_{i+1};

end

M $\leftarrow$ U $\circ$ L $\circ$ A $\circ$ T;

return *M;*

- Relative change in Function Value i.e. $\frac{F_{i+1}-F_i}{F_i} \leq 10^{-10}$
- Relative change in Gradient Value i.e. $\frac{G_{i+1}-G_i}{G_i} \leq 10^{-10}$
- Number of iterations $i \leq 10^5$

Finally, when the algorithm converges, we obtain the complete tensor M $\in \mathbb{R}^{u \times l \times a \times t}$ by taking the outer product of the individual components U, L, A and T. The computation complexity of this algorithm is O (N*k*J) where N is the number of dimensions of the tensor, *k* is the number of factors and J is $\prod_{n=1}^{N} I_n$.

9.5.5 Extending to N $>$ 4 Dimensions

We note that while we have focused on N=4 dimensions for concreteness, we can extend the model to accommodate additional dimensions, without loss of generality, by performing data engineering and modifying the objective function. For instance, if we add another contextual dimension such as users' purchases, we can supplement the sparse tensors and matrices with User $\times$ Purchases or Location $\times$ Purchases matrix obtained from another data source. We can then modify the objective function to include the purchases dimension and employ Algorithm 8 to minimize it.

9.6 Evaluation

9.6.1 Methodology

To evaluate the recommendations produced by our system, we used the following methodology: we randomly split each of the three datasets (from San Francisco Bay Area, Las

Vegas and Chicago) into training and testing sets with a 7:3 ratio, where we use the training set for training and tuning the model parameters. We use the held-out test set for computing the performance metrics over the predicted and ground truth values. More formally, we define *P* to be a test dataset containing n values. For each held out value $\in$ *P*, y_i denotes ground truth value and $\hat{y_i}$ denotes predicted value.

9.6.2 Metrics

We use three standard performance metrics [137] for evaluating the performance of our approach on a test set *P*:

- Root Mean Squared Error (RMSE) - RMSE is computed as

$$\sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}.$$

  However, RMSE can be susceptible to large errors and often places more emphasis on them. Hence, we compute Mean Absolute Error (MAE) as well to evaluate the performance of our approach.

- Mean Absolute Error (MAE) - MAE is computed as

$$\frac{\sum_{i=1}^{n}|\hat{y_i} - y_i|}{n}$$

  However, both RMSE and MAE may be less appropriate for tasks where a ranked result is returned to the user, who then only views items at the top of the ranking. For this, we compute Normalized Discounted Cumulative Gain (nDCG).

- Normalized Discounted Cumulative Gain (nDCG) - nDCG is commonly used in information retrieval to measure a search engine's performance. A higher nDCG value for a list of search results indicates that more relevant items were ranked higher in the list. In particular, nDCG@p measures the relevance of top p results and is defined as:

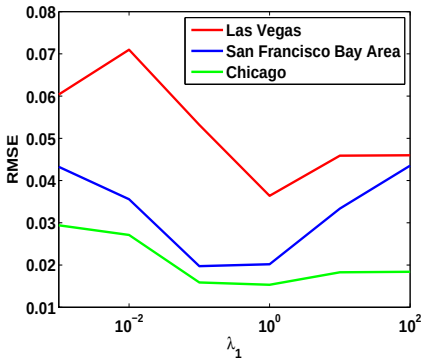
  nDCG@p $= \frac{\text{DCG@p}}{\text{iDCG@p}}$ where DCG@p $= rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 rel_i}$, iDCG@p is the DCG@p value of ideal ranking list and rel$_i$ is a relevance value. nDCG ranges from 0 to 1. The higher the nDCG value is, the better a ranking result list is.

9.6.3 Parameter Tuning

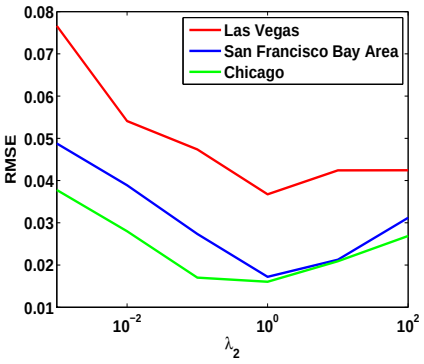
We performed parameter tuning via parameter-sweeping experiments on the different training sets. We randomly split each training set into training and validation sets with a 4:1 ratio. We held out the validation set and constructed the model using the training set with different values for the model parameters, λ_1 - λ_5, # of factors (k) and location grid size r. We computed RMSE on the held out validation sets and picked the parameter values that minimized it.

9.6.3.1 Impact of model parameters

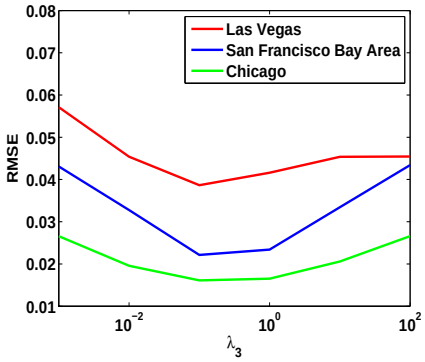
For tuning each individual parameter, we set the remaining parameters as 0.00001 in order to reduce their impact on the model performance. We then ran the parameter-sweeping experiments 5 times for each parameter value and averaged the RMSE. Figure 9.4 shows the variation of RMSE for different values of the model parameters for the 3 validation



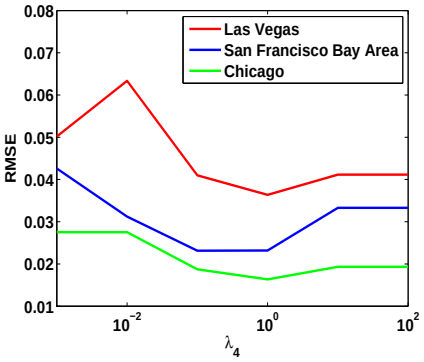
(a) RMSE variation for λ_1



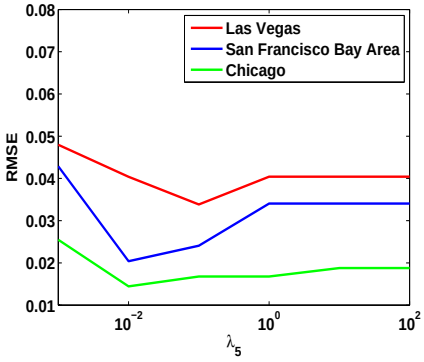
(b) RMSE variation for λ_2



(c) RMSE variation for λ_3



(d) RMSE variation for λ_4



(e) RMSE variation for λ_5

Figure 9.4: Variation of RMSE for different values of λ_1 - λ_5 on the validation sets

sets. As evident, the RMSE first increases and later decreases as value of each parameter (λ_1 - λ_5) increases. This is because when a parameter value is too small, the model cannot fully utilize the information from the corresponding matrix. On the other hand, if it is too large, then the information from the matrix will dominate the objective function. After this tuning, we set $\lambda_1 = 1.0$, $\lambda_2 = 1.5$, $\lambda_3 = 0.5$, $\lambda_4 = 1.0$ and $\lambda_5 = 0.02$.

9.6.3.2 Impact of number of factors

We varied the number of latent factors (k) from 10 to 50. However, we observed that the RMSE did not exhibit significant variation which implies that changing the number of factors did not have a significant impact on performance, as also reported by Zheng et al. [153, 156]. For our experiments, we set $k = 30$.

9.6.3.3 Impact of location grid size

Table 9.4 shows the RMSE for the 3 validation datasets for different location grid sizes. The grid size of 500m has the lowest RMSE for all the validation sets and hence we use that for our experiments. This is also intuitive because recommended locations within a distance of 500m can be easily reached on foot.

Henceforth, all experiments will use the tuned parameter values.

9.6.4 Comparison with Baselines

We compare our approach with 7 state-of-the-art baselines.

Grid Size r (m)	RMSE		
	San Francisco	Las Vegas	Chicago
300	0.0217	0.0378	0.0170
500	**0.0213**	**0.0376**	**0.0162**
700	0.0222	0.0392	0.0175
1000	0.0224	0.0437	0.0186

Table 9.4: RMSE on validation set for various location grid sizes

9.6.4.1 Collaborative Filtering baselines

These baselines exploit similarity on each of the individual dimensions to complete the tensor. They take only the 4 dimensional tensor as input. We implemented 4 CF baselines:

- User-User Collaborative Filtering baseline (UCF) - This baseline exploits the user-user similarity information to fill in the missing entries of the sparse tensor X. In particular, for UCF, we consider CF on each user $\times$ location matrix with respect to each activity and each time slot, on each user $\times$ activity matrix with respect to each location and each time slot, and on each user $\times$ time matrix with respect to each location and each activity independently. To this end, we matricize X in Mode 1 to generate matrix $X^{(1)} \in \mathbb{R}^{u \times lat}$. We then use Pearson correlation coefficient between the vectors in the matrix to compute pairwise user similarity information. For each user, we compute the weighted average of the top N similar users to predict the missing values.

- Location-Location Collaborative Filtering baseline (LCF) - Similarly, the LCF baseline exploits the location-location similarity information to fill in the missing entries of X. We matricize X in Mode 2 to generate matrix $X^{(2)} \in \mathbb{R}^{l \times uat}$. We use Pearson correlation coefficient between the vectors in the matrix to compute pairwise location similarity information. For each location, we then compute the weighted average of the top N similar locations to predict the missing values.

- Activity-Activity Collaborative Filtering baseline (ACF) - The ACF baseline exploits the activity-activity similarity information to fill in the missing entries of X. We matricize X in Mode 3 to generate matrix $X^{(3)} \in \mathbb{R}^{a \times ult}$. We use Pearson correlation coefficient between the vectors in the matrix to compute pairwise activity similarity information. For each activity, we then compute the weighted average of the top N similar activities to predict the missing values.

- Time-Time Collaborative Filtering baseline (TCF) - Finally, the TCF baseline exploits the time- time similarity information to fill in the missing entries of X. We matricize X in Mode 4 to generate matrix $X^{(4)} \in \mathbb{R}^{t \times ula}$. We use Pearson correlation coefficient between the vectors in the matrix to compute pairwise time similarity information. For each time slot, we then compute the weighted average of the top N similar time slots to predict the missing values.

In these experiments, we set N = 10, since the prediction results do not depend on N significantly as suggested by Zheng et al. [156]

Approach	RMSE		
	San Francisco	Las Vegas	Chicago
Our approach	**0.0197**	**0.0339**	**0.0153**
UCF	0.0324	0.0486	0.0293
LCF	0.0336	0.0430	0.0311
ACF	0.0333	0.0433	0.0317
TCF	0.0319	0.0484	0.0315
Standard CP	0.0224	0.0427	0.0178
HOSVD	0.0227	0.0405	0.0175
ALS	0.0222	0.0389	0.0173

Table 9.5: RMSE for our approach and baselines on the 3 test sets

9.6.4.2 Model based baselines

We implemented the standard CP decomposition model which, when applied to our multi-dimensional recommendation problem, has an objective function of the form:

$F = \frac{1}{2}\|X - U \circ L \circ A \circ T\|^2 + \frac{\lambda_5}{2}(\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$ Thus, it takes only the 4 dimensional tensor as input and its objective function has only the tensor decomposition term along with the regularization term. This can be obtained by setting λ_1 - λ_4 = 0 in our objective function (see Section 9.5.3) and replacing the weighted least squares error term with the standard least squares error term in summand F_1. We then minimize this objective function using the gradient descent based Algorithm 8.

9.6.4.3 Algorithmic baselines

The 2 algorithmic baselines that we employ are:

- Higher Order Singular Value Decomposition (HOSVD) - We implement the HOSVD approach proposed by Lathauwer et al. [163]. This approach also takes only the 4 dimensional tensor as input. It first matricizes the 4 dimensional User $\times$ Location $\times$ Activity $\times$ Time tensor X along each of the 4 modes to get the matrices $X^{(1)}$, $X^{(2)}$, $X^{(3)}$ and $X^{(4)}$. On each matrix, SVD is applied to compute the low rank approximation: $X^{(i)} = U^{(i)} \cdot S^{(i)} \cdot {V^{(i)}}^{\top}$ where $1 \leq i \leq 4$.

  The core tensor S is then constructed as:

  $$S = X \times_1 {U_{c_1}}^{(1)^{\top}} \times_2 {U_{c_2}}^{(2)^{\top}} \times_3 {U_{c_3}}^{(3)^{\top}} \times_4 {U_{c_4}}^{(4)^{\top}}$$

  where ${U_{c_1}}^{(1)^{\top}}$, ${U_{c_2}}^{(2)^{\top}}$, ${U_{c_3}}^{(3)^{\top}}$, and ${U_{c_4}}^{(4)^{\top}}$ are the transpose of the c_1-dimensionally reduced $U^{(1)}$, c_2 - dimensionally reduced $U^{(2)}$, c_3-dimensionally reduced $U^{(3)}$, and c_4-dimensionally reduced $U^{(4)}$ matrices respectively.

  Finally, the completed matrix M is obtained as:

  $$M = S \times_1 {U_{c_1}}^{(1)} \times_2 {U_{c_2}}^{(2)} \times_3 {U_{c_3}}^{(3)} \times_4 {U_{c_4}}^{(4)}$$

  where c_1, c_2, c_3 and c_4 are set empirically. Based on the experiments of Nanopoulos [162], we preserve 30% of the information in each matrix.

- Alternating Least Squares (ALS) - In ALS, the objective function is a standard CP formulation and the idea is to solve for each factor matrix, leaving all other factors fixed. We implement the ALS algorithm proposed by Kolda and Bader [165, 167]

Approach	MAE		
	San Francisco	Las Vegas	Chicago
Our approach	**0.0076**	**0.0179**	**0.0049**
UCF	0.0104	0.0314	0.0062
LCF	0.0102	0.0310	0.0066
ACF	0.0105	0.0323	0.0069
TCF	0.01	0.0354	0.0065
Standard CP	0.0095	0.0228	0.0058
HOSVD	0.0098	0.0239	0.0061
ALS	0.0093	0.0218	0.0054

Table 9.6: MAE for our approach and baselines on the 3 test sets

for the standard CP decomposition (see Section 9.6.4.2) of our multi-dimensional recommendation problem.

9.6.5 Results

For testing experiments, we held out the test dataset and generated the completed tensor using the training dataset with the tuned parameter values, number of factors k (30) and for the optimal grid size (500m). We then computed the RMSE and MAE between the predicted and the ground truth held out values of the test set.

Since we do not have human supplied relevance rankings, we compute nDCG on the held-out known values. We calculated nDCG with respect to each recommendation dimension by fixing the remaining dimensions and computing nDCG on the current di-

Approach	nDCG@5 for location		
	San Francisco	Las Vegas	Chicago
Our approach	**0.898**	**0.835**	**0.821**
UCF	0.691	0.523	0.693
LCF	0.71	0.514	0.682
ACF	0.702	0.582	0.695
TCF	0.735	0.593	0.701
Standard CP	0.685	0.453	0.651
HOSVD	0.890	0.815	0.798
ALS	0.798	0.722	0.748

Table 9.7: nDCG for location for our approach and baselines on the 3 test sets

mension. Thus, for each of the held out values in the test set, we first fixed user, location and activity and ranked the time slots in the completed tensor. This ranking was used to calculate DCG (refer to Section 9.6.2) for time. To compute iDCG for time, we determined the ranking for the ground truth time values. We then calculated nDCG@p (with p = 5) for time. Finally, we averaged the values for all user, location and activity combinations to generate an averaged nDCG@5 for the time dimension. Similarly, we fixed user, location and time values and ranked activities to compute nDCG@5 for the activity dimension and fixed user, activity and time to compute nDCG@5 for the location dimension. To ensure statistical significance, we ran all algorithms 5 times on each test dataset and averaged the result for each metric.

Approach	nDCG@5 for activity		
	San Francisco	Las Vegas	Chicago
Our approach	**0.869**	**0.827**	**0.798**
UCF	0.452	0.465	0.467
LCF	0.457	0.477	0.485
ACF	0.440	0.458	0.412
TCF	0.391	0.34	0.401
Standard CP	0.341	0.314	0.393
HOSVD	0.829	0.817	0.714
ALS	0.770	0.737	0.692

Table 9.8: nDCG for activity for our approach and baselines on the 3 test sets

Tables 9.5, 9.6, 9.7, 9.8 and 9.9 show the results achieved by our approach as well as the baselines on the 3 test datasets for the 3 performance metrics: RMSE, MAE and nDCG@5. For RMSE and MAE, a lower value signifies superiors performance while for nDCG, a higher value signifies superior performance. Clearly, our approach outperforms these baselines.

9.6.6 Discussion of Results

As evident from the results, the neighborhood-based Collaborative Filtering baselines exhibit poor performance with respect to all the 3 performance metrics. There are two possible reasons for this result. First, these approaches employ only the User $\times$ Location

Approach	nDCG@5 for time		
	San Francisco	Las Vegas	Chicago
Our approach	**0.833**	**0.832**	**0.807**
UCF	0.581	0.612	0.605
LCF	0.563	0.605	0.567
ACF	0.541	0.609	0.575
TCF	0.593	0.632	0.582
Standard CP	0.578	0.605	0.563
HOSVD	0.812	0.82	0.798
ALS	0.732	0.725	0.656

Table 9.9: nDCG for time for our approach and baselines on the 3 test sets

$\times$ Activity $\times$ Time tensor. Since the tensor is extremely sparse, computing similarity along any dimension will be error prone as most of the entries are missing. Hence, our approach which supplements the sparse tensor with additional information from external sources overcomes this hurdle. Second, each of these baselines predicts the missing values based on similarity along one dimension only, ignoring the other dimensions. Since our problem involves collaborative recommendations along multiple dimensions, it is important to employ all the dimensions for recommendations as we do.

The model based and algorithmic baselines such as Standard CP, HOSVD and ALS also utilize only the sparse tensor as input. As pointed out in Section 9.2, the standard CP approach demonstrates an increase in error if the data is extremely sparse. Similarly, ALS

has poor convergence if the data is sparse and does not scale to large datasets. Moreover, HOSVD and ALS have high space complexity. On the other hand, our approach overcomes the sparsity of the data by supplementing the sparse tensor with coupled matrices and also scales to large datasets. Hence, it outperforms these baselines.

Also, we note that the nDCG values for location varies greatly for each dataset for each algorithm while the nDCG values for activity and time do not exhibit such a high variation. This is possibly because the number of locations in each of the dataset varies significantly while the number of activities and time slots are constant for each dataset. In addition, the nDCG values for location are higher in general for most of the algorithms and datasets, as compared to activity and time. This indicates that location is the most important dimension for recommendation, followed by time and activity. This is also intuitive because in our datasets, each photo has a location and timestamp but $< 50\%$ photos have meta data to enable activity inference. Hence, the activity data is sparser than for location and time. This would affect the performance of all the baselines since they use the sparse tensor as the only input. However, in our approach, we supplement this sparse tensor with additional matrices involving Location, Activity and Time. We also perform regularized temporal smoothing. Hence, the nDCG for Activity and Time are higher and comparable to that of Location for our approach.

9.6.7 Runtime Comparison

Since our data is very sparse, we performed a weighted decomposition of the tensor which minimizes the error in only the known entries of the tensor. We also experimented without

Approach	Time (in seconds)		
	San Francisco	Las Vegas	Chicago
Without weighting	12000	2465	10500
With weighting	9600	1180	8580

Table 9.10: Runtime comparison

the weighting imposed on the tensor i.e. minimize the error in all the entries of the tensor. In this case, our objective function is:

$$F = \frac{1}{2}\|X - U \circ L \circ A \circ T\|^2 + \frac{\lambda_1}{2}\|Y - LA^\top\|^2 + \frac{\lambda_2}{2}\|S - AA^\top\|^2 + \frac{\lambda_3}{2}\|Z - LL^\top\|^2 + \frac{\lambda_4}{2}\|TR\|^2 + \frac{\lambda_5}{2}(\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$$

We minimized it using Algorithm 8. Both approaches have the same performance with respect to RMSE, MAE and nDCG@5. However, they differ in their runtime. As shown in Table 9.10, the weighted approach is faster as it minimizes the loss on only the known entries of the tensor. All experiments were run on a 64 bit Windows machine with core i7 processor and 16 GB RAM.

Chapter 10: **Modeling Users' Behavior from Large Scale Smartphone Data Collection**

Users' sensed context can be aggregated over a period of time to generate their context history. The Learning Engine of the Rover II middleware framework (Figure 3.1) can then mine this history for building user behavioral models using a combination of machine learning and data mining algorithms such as Decision Trees, k Nearest Neighbor (kNN) classifier, k Means clustering, Hidden Markov Models, and Association Rule Mining. These algorithms are trained on the users' context history to induce probabilistic associations and patterns.

This chapter explains the design and implementation of the Learning Engine of the Rover II middleware framework [13]. This engine implements several novel approaches and algorithms that employ various contextual features and state of the art machine learning techniques for building diverse behavioral models of users.

10.1 Introduction

While discrete observations of an individual's behavior can appear almost random, typically there are repetitive and easily identifiable patterns or routines in every person's life. For many people, a typical weekday routine consists of leaving home in the morning and

traveling to work, going for lunch in the afternoon, and returning home in the evening. These daily routines are often coupled with routines across other temporal scales, such as weekly (e.g. going hiking or running on weekends) or monthly (e.g. visiting family during holidays) patterns.

In today's world, smartphones are the most ubiquitous devices and have become an integral part of people's everyday lives. People carry them around everywhere and use them as their primary medium for many day to day activities. These devices can collect a variety of data about users such as their locations (from GPS), sensory data (from various sensors), call and sms logs etc. As a result, they can act as a rich content source for users' contextual information. However, a large volume of research in mobile and ubiquitous systems has been devoted to using this collected information for inferring users' current high level context such as their environmental context (whether they are indoors or outdoors), the activities they are engaged in (walking, driving etc.) and how many people are around them [9, 45, 46]. On the other hand, mining of users' diverse longitudinal behavioral patterns from rich smartphone data, which can enable exciting new context-aware applications and services, has not received much attention.

Figure 10.1 shows our broader long-term vision. As part of this vision, we plan to collect large-scale data from users' smartphones and employ it to infer diverse frequent patterns that capture different aspects of their behavior. We plan to explore the utility of each type of pattern in improving the users' quality of life by proactively taking actions on their behalf. As shown, we envision a client agent application sensing the user's temporal multi-dimensional context and activity information (e.g. location, call logs and app usage). This information is aggregated over a period of time in the form of behavioral

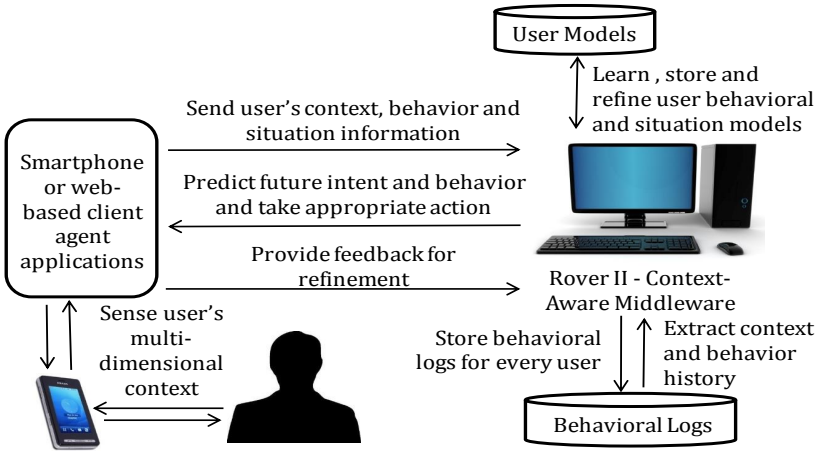


Figure 10.1: Long term vision for building diverse user behavioral models, from users' sensed data, in order to take proactive actions

logs. A context-aware middleware extracts the user's *context history* (for instance, call or location history) from these logs. It utilizes this history to learn and store the users' *behavioral models* (behavioral patterns that users exhibit in similar context or situations over a period of time) for predicting future behavior. Ultimately, this enables the middleware to act proactively on the users' behalf in anticipation of their future goals and intentions without explicit requests from them. The system also refines these models periodically based on users' feedback.

In this work, we take a step towards achieving this vision. We develop an infrastructure for learning diverse patterns, from large-scale data collected from users' smartphones, and utilizing these patterns to help identify a variety of their behaviors, habits, and daily life places and activities. A key aspect of our research is how to mine this massive amount of data, captured over long durations of time, for inferring users' high level behavioral models. In particular, we try to answer questions such as:

- What are the various places frequently visited by the user? Which among them is his home, workplace, recreational or known places etc.?
- Will the user receive an incoming call in his current situation? Who will be the next person that he will call?
- When and where does the user usually charge his device? How is his device battery usage and can we predict his future device battery level?
- What applications (referred to as 'apps') does he use most frequently in the morning as opposed to night?

These behavioral models would enable the context-aware system to predict a user's behavior and proactively take actions on his behalf. For instance, the system can leverage a user's past communication behavior to predict his availability to receive an incoming call and proactively reject it if he is unavailable (say, if he is in a meeting at work). It could periodically order the user's contacts based on who will be the most likely contact that will be called next. Similarly, it can employ his device charging behavior to prompt him if he forgets to charge his device. It can even load his favorite apps ahead of time based on his app usage history. Ultimately, such a system can save the user's time and energy and improve his quality of life.

Our key contributions in this work are:

- We design and implement a unified infrastructure for modeling a diverse set of users' behaviors from large-scale data collected from their smartphones.
- We design and implement novel approaches and algorithms that employ users' con-

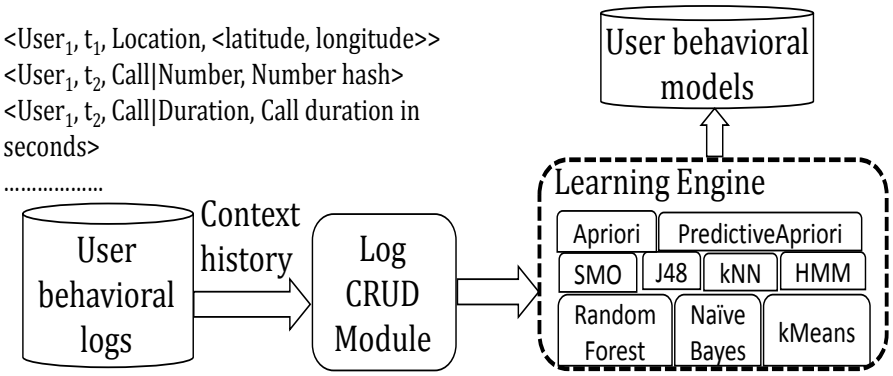


Figure 10.2: The Learning Engine pipeline of Rover II

textual features and state of the art machine learning techniques for building various behavioral models of users. Examples of generated models include classifying users' semantic places (such as Home, Work etc.) and mobility states (Stationary or Moving), predicting their availability for accepting calls and inferring their device charging behavior.

- We evaluate our work on large-scale real-world smartphone data of 200 users, from the DeviceAnalyzer [51] dataset, consisting of 365 million data points.

- We show that our algorithms and approaches can model user behavior with high accuracy and demonstrate improved performance over existing approaches.

10.2 Rover II context-aware middleware

The Rover II context-aware middleware [16, 17] is a generic middleware, which serves as an integration platform for mobile and desktop applications. It can store and retrieve contextual information, as well as learn and store user behavior models. It consists of several components including a main Controller module (which controls the flow of informa-

tion among the various components), an Activity Manager (which defines what activities the system can perform on the user's behalf), a Learning Engine (which learns patterns from user's behavior) and a Relevant Information Discovery and Ranking Engine (which determines what information will be relevant to the user's current situation). Here, we describe the core component responsible for learning user behavioral models - the Learning Engine.

Figure 10.2 shows the pipeline for the Rover II Learning Engine. A user's behavioral log is collected from his smartphone and stored in a relational database in the form of timestamped <key,value> pairs where each key is unique and represents the type of contextual data that is logged e.g. location (represented in latitude and longitude format) or call data including the number called, duration etc. The data is stored in chronological order. The Log CRUD Module extracts the user's context history (location trace, sensor log, call history etc.) over a certain window of time, from these logs, and sends it to the Learning Engine.

The Learning Engine implements several state of the art supervised and unsupervised machine learning techniques including classifiers such as J48 Decision Tree, *k* Nearest Neighbors (kNN), Naive Bayes, Random Forest, and Sequential Minimal Optimization (SMO) [173] (an algorithm used for training support vector machines), clustering algorithms such as *k* Means and DBSCAN, association rule mining algorithms such as Apriori and PredictiveApriori as well as Hidden Markov Models (HMM) that model causal relationships. In the current version of our system, we employ the Weka [53] and ELKI [174] libraries for implementing these techniques.

The engine extracts various contextual features from the users' context history and

applies these techniques to them in order to generate user behavior models. These models are either held in memory for immediate use or persisted to disk for subsequent usage in prediction of users' behavior.

10.3 Data

The data used in this research comes from two sources:

10.3.1 DeviceAnalyzer dataset

DeviceAnalyzer [51] is a free smartphone based Android application that runs continuously in the background and collects a user's data from his smartphone. It was developed to collect a large-scale research data-set of phone usage. It has collected usage information from 17,000 Android devices, over the course of nearly 3 years, and contains 100 billion data points. It captures a rich and highly detailed time-series log of approximately 300 different events[1] including sensory information, Wi-Fi and bluetooth scans, call and sms logs, running processes and applications etc. This data is stored in the form of timestamped key-value pairs.

Since not all users in the DeviceAnalyzer dataset consented to sharing location and sensory information, we pre-processed the data to include only those users who have shared all information from their phone. In order to have enough training and testing instances, we also set the criteria that only those users who have > 30 days of data should be included. Our final dataset consists of 200 users and includes over 365 million data

[1] A complete description of the logged event types is available at http://deviceanalyzer.cl.cam.ac.uk/keyValuePairs.htm

Feature	Description or Representation
Average acceleration	Average acceleration of the device
Variance of acceleration	Variance of acceleration of the device

Table 10.1: Movement related features

points logged over 100,000 days for all users. We use about 20% of this data as Validation data for parameter tuning.

10.3.2 Field study

Despite the richness of the DeviceAnalyzer dataset, a disadvantage of it is that it does not have ground truth labels for semantic places and for mobility states. To address this, we conducted a field study using the DeviceAnalyzer app to collect labeled data over a period of 1 month. This data was collected by 10 members of our lab (including the first author). All the participants who collected the data also annotated it carefully to provide ground truth values for place labels (see Table 10.4) and mobility states (see Table 10.2). This dataset consisted of around 500,000 data points. We split this dataset into Training and Testing data with a 1:1 ratio. We used the training dataset for building some of the models and the testing set for their evaluation.

Label	Description or Representation
Stationary	Whether user is stationary
Moving	Whether user is moving

Table 10.2: Mobility state labels

Accuracy	kNN	J48	Random Forest
	0.81	0.72	**0.85**

Table 10.3: Comparison of accuracies of various classifiers on the training set for Mobility State Classification

10.4 Feature engineering and Algorithm Design for User behavior modeling

10.4.1 Mobility State Classification

The DeviceAnalyzer app samples the data from the accelerometer of a device at a frequency of approximately 0.003 Hz (once per 5 minutes). Moreover, this data is not logged in its raw format but as aggregate measurements (such as count, variance, average etc.) computed over all magnitudes of sensor values captured over a 1-second window. Hence, fine-grained activity recognition as performed in [9, 45, 46] is not possible with this dataset.

Instead, we classify the user's *mobility state* into two classes - Stationary and Moving. We extract movement related features such as average and variance of acceleration

(see Table 10.1) to build this model. We experimented with 3 classifiers - kNN (k=3), Random Forest (10 trees) and J48 decision tree on the training data collected as part of the field study (see Section 10.3.2) with 10 fold cross validation. As shown in Table 10.3, Random Forest had the best performance. Hence, we implement it for Mobility State Classification.

10.4.2 Timestamp hashing

Since the number of timestamps in our dataset is huge (as it spans nearly 100,000 days of data for all users), we perform feature hashing to hash the value of each timestamp to a timeslot in order to reduce the data dimensionality. We generate buckets of hashed time slots based on the day of the week and time of the day. This also enables us to capture daily and hourly patterns in behavior which further help in user behavior modeling.

To this end, we divide a day into 4 hourly slots:

- Morning - hours between 5 am and 10 am
- Noon - hours between 10 am and 5 pm
- Evening - hours between 5 pm and 10 pm
- Night - hours between 10 pm and 5 am

We generated these slots using commonsense knowledge that most people tend to sleep between 10 pm and 5 am. They usually commute between 8 and 9 am and reach work between 9 and 10 am. In addition, most places have work hours are till 4:30 or 5 pm.

Since there are 2 types of days in a week - weekday and weekend, and each day has 4

Label	**Description or Representation**
Home	Home location of the user
Work	Work location of the user
IndoorRecreation (InRec)	Place for indoor sports and recreation (e. g. gym)
OutdoorRecreation (OutRec)	Place for outdoor sports and recreation (walking, hiking, running etc.)
Transport	Place related to transportation (e.g. road, bus stop, train station, parking lot)
IndoorKnown (InKnown)	Known indoor place (e.g. friend's home or a coffee shop)
OutdoorKnown (OutKnown)	Known outdoor place (e.g. park)
NearHome	Place near home location
NearWork	Place near work location
Other	Any other place

Table 10.4: Semantic place class labels

hourly slots, the total number of hashed timeslots is 8. Each timestamp is first converted to a standard date time format which is then hashed to a timeslot. For instance, a timestamp of 2013-05-17 14:07:00 will be hashed to 'WeekDay_Noon'.

10.4.3 Semantic Place Classification

Semantic Place Classification involves association of appropriate meaningful semantic labels with a user's location[2]. Knowing the semantics of a location can enable a context-aware system to provide the user with information relevant to his current location e.g. work related notes at work or grocery list when he is at the grocery store. Moreover, this knowledge can also help predict other behaviors of the user such as his availability to attend a call or his intention to charge his phone.

As mentioned earlier, human behavior usually follows a regular pattern in practice e.g. people sleep at home at night, move continuously when in the gym or running outside, and charge their phones indoors. We exploit these patterns to label the semantic places of users. Table 10.4 shows the 10 semantic place labels which we use to classify the locations in a user's location history. Some of these overlap with those used in the Nokia Mobile Data Challenge (MDC) dataset [175, 176].

The location data in the DeviceAnalyzer dataset has been obtained via the Android network provider, instead of GPS, due to privacy constraints. This provider determines user location using cell tower and Wi-Fi signals. As a result, the obtained location is coarse-grained and not very accurate. Moreover, the sampling is duty cycled to conserve power. Hence, we rely on several additional features to label a user's locations. Table 10.5 shows the 14 spatial and temporal features that we extract from a user's context history and employ for semantic place classification. These are:

- Place visit count: This is computed as the number of unique visits to a place. This

[2] We use place and location interchangeably throughout the chapter.

Feature	**Description or Representation**
Place visit count	Number of unique visits to a place
Relative place visit frequency	Number of visits (per day) to a place
Place stay duration	Total stay duration at each place
Average place stay duration	Average stay duration (per visit) at each place
Place - Time slot visit frequency	Number of unique time slots for each visited place
Time slot - place visit frequency	Place visit count for each time slot
Time slot - place time frequency	Place stay time for each time slot
Bluetooth count	Average number of people around at each place
Bluetooth diversity	Change in the people for consecutive visits to each place
Wi-Fi count	Average number of unique Wi-Fi APs heard at each place
Wi-Fi RSSI	Average Wi-Fi RSSI at each place
Wi-Fi connectivity	Wi-Fi connectivity status at each place
Charging state frequency	Charging state of the device at each place - connected (via ac/usb) or disconnected
Mobility state frequency	Mobility state of the device at each place - stationary or moving

Table 10.5: Spatial and temporal features used for Semantic Place Classification

feature helps in identifying places such as ‘Home’ which a user would visit the most.

- Relative place visit frequency: This is the number of visits to a place per day and is computed as

$$\frac{\text{Number of unique visits to a place}}{\text{Total number of days in the user's location history}}$$

  This feature too helps in identifying places such as ‘Home’ which would be visited almost everyday.

- Place stay duration: This is computed as the total time spent at a place. This again should be high for ‘Home’ as most people spend a bulk of their day at home.

- Average place stay duration: This is the average stay duration at a place (per visit) and is computed as

$$\frac{\text{Total stay duration at a place}}{\text{Number of visits to the place}}$$

  .

- Place - Time slot visit frequency: This is computed by counting the unique time slots at which a place has been visited by the user. This feature helps in discriminating between places such as ‘Home’ and ‘Work’ as most people visit their homes at all possible time slots but usually visit their work place on weekdays only.

- Time slot - place visit frequency: This is computed by calculating the count of a user’s unique visits to a place in each time slot.

- Time slot - place time frequency: This is computed by calculating the total time spent at a place, by the user, in each time slot. These temporal features too help identify 'Home' and 'Work' as most people are at home during the night and at work during the noon time slots.

- Bluetooth count: As suggested by our previous work in [9], bluetooth count can be utilized to determine the number of people around. We compute an average bluetooth count for each unique place in the user's location history.

- Bluetooth diversity: To compute this, we determine the set of bluetooth devices scanned at each unique visit to a place. For every two consecutive visits, we compute the ratio of the intersection to the union of the device sets and average all the values to generate a final value. The bluetooth features are useful for identifying places such as recreational spots which several people may visit and at different times.

- Wi-Fi count: This is computed by counting the number of visible Wi-Fi Access Points (APs) heard at a location for each unique visit and averaging the values.

- Wi-Fi received signal strength (RSSI): This is computed by recording Wi-Fi RSSI values heard at a location for each unique visit and averaging them. These Wi-Fi related features enable distinguishing between indoor and outdoor places.

- Wi-Fi connectivity status: This is generated as the count of times the user's device is connected to Wi-Fi at a place. This feature helps in identifying places that are known to a user as his/her device would connect to the Wi-Fi network at a known

place only and if it has access to it.

- Charging state frequency: This represents the charging state (ac, usb or disconnected) frequencies of the device at a place. This feature too helps in distinguishing between indoor and outdoor environments as ac charging can occur indoors only.

- Mobility state frequency: This represents the mobility state (stationary/moving) frequencies of the user at a place. This feature helps identify recreational and transportation places where the user would exhibit more movement.

The Learning Engine of Rover II employs Algorithm 9 for identifying the semantic labels of all the places visited by a user. The algorithm takes as input the user's context history consisting of the 14 spatial and temporal features described in Table 10.5. It returns the list of all locations, as well as their semantic place labels, in the user's location history.

As shown, we first hash each timestamp in the user's context history to a time slot. We then discretize each location (expressed in latitude and longitude), in the user's location history, into virtual rectangular bins that are formed throughout the geographical coordinate space. This helps in reducing redundancy in the locations. Each bin is created through a location hashing function which takes as input: (i) latitude and longitude as 64-bit floats and (ii) a bin size r in meters. The function leverages the fact that latitude and longitude are expressed in decimal degrees, with the fifth decimal place corresponding roughly to 1 meter. Since this precision was acceptable to us, we truncated each value to five decimal places. The function produces the resulting integer key with the longitude and latitude ending up in the high and low bits, respectively. This key identifies a virtual

Algorithm 9: Algorithm for Semantic Place Classification

Input: User's context history data consisting of the spatial and temporal features described in Table 10.5

Output: List of locations visited by user and their semantic place labels

foreach *Timestamp for which there is a location logged* **do**

    Hash each timestamp to a time slot;

    Hash each location (in latitude and longitude format) to a discrete virtual rectangular bin of size 500m;

end

Cluster the hashed locations based on distance;

foreach *hashed location in user's location history* **do**

    Compute total stay duration, visit count, time slot - place visit frequency, average stay duration, relative visit frequency;

    Compute the time slot at which the location is visited most and the count of unique time slots at which it is visited;

    Compute the ratio of time spent on weekends to time spent on weekdays for the location;

    Compute the average count of visible Wi-Fi APs and the average Wi-Fi RSSI for the location;

    Compute the average count and diversity of bluetooth devices for the location;

    Compute the normalized mobility state frequencies, charging state frequencies and Wi-Fi connectivity status;

end

foreach *time slot* **do**

    Compute the total stay duration and visit counts for each visited place, in the user's location history, for this time slot;

    Compute the location that is visited the most, and the location at which the most time is spent, during this time slot;

end

Label 'Home' as location that has highest stay duration, highest relative visit frequency, highest visit count and has been visited at all time slots;

Label 'Work' as the place that is visited most or where most time is spent during the Noon time slot, which is not 'Home', for which the ratio of weekend to weekday time < 1.0 and which has not been visited on all time slots;

foreach *hashed location in user's location history (other than 'Home' and 'Work')* **do**

    Label as 'IndoorRecreation', 'OutdoorRecreation' or 'Transport' based on its features such as normalized mobility state of the user, average count of visible Wi-Fi APs, average Wi-Fi RSSI, average bluetooth count and diversity ;

end

foreach *unlabeled hashed location in user's location history* **do**

    Label as 'IndoorKnown' or 'OutdoorKnown' based on features such as visit count, # of time slots at which visited, average count of visible Wi-Fi APs, average Wi-Fi RSSI, and Wi - Fi connectivity status;

    if *location not labeled as 'IndoorKnown' or 'OutdoorKnown'* **then**

        if *location belongs to the same cluster as Home* **then**

            Label as 'NearHome';

        else if *location belongs to the same cluster as Work* **then**

            Label as 'NearWork';

        else

            Label as 'Other';

end

return *List of locations visited by user and their semantic place labels;*

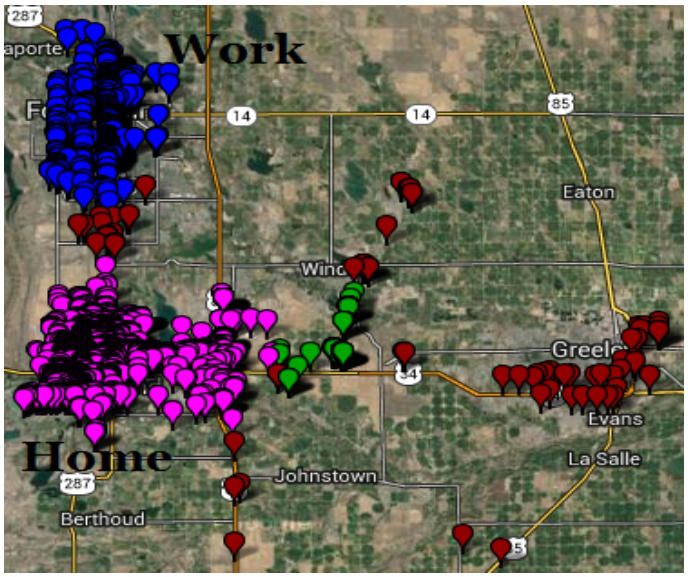


Figure 10.3: Location clusters for a randomly chosen user in our dataset (best viewed in color).

bin approximately *r* meters per side, although the bin will be elongated north-to-south for regions further away from the equator due to the Earth's curvature. Based on empirical observations, we set *r* as 500m in our current implementation.

We then cluster the hashed locations using the DBSCAN [177] algorithm which identifies clusters in large spatial datasets by looking at the local density of database elements. DBSCAN takes two parameters as input - *MinPts* which is the minimum number of points in the vicinity of a point in order for it to be the cluster center and *Eps* which is the vicinity radius. We set *MinPts* as 100 and *Eps* as 2400m in our current implementation. Figure 10.3 show the location clusters generated for a randomly chosen user in our dataset. To improve legibility of the figure, so that individual locations can be discerned, we sampled the number of data points down. The user's 'Home' cluster (colored in pink) and the 'Work' cluster (colored in blue) are clearly evident. The locations colored in maroon are

outliers or ‘Noise’ as determined by DBSCAN as they do not belong to a cluster but are other places that are visited by the user.

We now extract the spatial and temporal features described earlier, for each hashed location, and use them for labeling the locations. We first identify the user’s ‘Home’ location. To achieve this, we exploit the fact that ‘Home’ is the place where we spend the most time, which we visit the most and on all days and at all possible time slots. Once the ‘Home’ is obtained, we identify his ‘Work’ location. Most people follow a day time schedule and hence we classify ‘Work’ as the place which is not ‘Home’, and which is visited most and where most time is spent during the noon time slot. We also utilize the fact that ‘Work’ will usually not be visited on weekends (hence, ratio of time spent on weekends to weekdays will be < 1) and will not be visited at all possible time slots.

We then classify all the other locations in the user’s location history into the 8 remaining classes. For labeling ‘IndoorRecreation’, ‘OutdoorRecreation’ and ‘Transport’ places, we use features such as the average number of visible Wi-Fi APs and RSSI, average number and diversity of bluetooth devices, and average movement of the user (as observed from his normalized mobility state). This approach is intuitive because recreational and transportation places will involve more movement, will typically be visited by many people and by different people during different times. Also, indoor places will have a higher count of APs and higher average RSSI.

Locations which do not get classified into these 3 classes are then categorized as ‘IndoorKnown’ and ‘OutdoorKnown’ based on features such as their visit count, number of unique time slots at which they are visited, average number of visible Wi-Fi APs and average Wi-Fi RSSI, and Wi-Fi connectivity status of the device. This approach is also

Accuracy	kNN	J48	Naive Bayes
	0.78	**0.85**	0.63

Table 10.6: Comparison of accuracies of various classifiers on the training set for Semantic Place Classification

intuitive as known places (such as a friend's home or a coffee shop) will be visited more often by the user and possibly at different times of day. Moreover, a user's device will be connected to Wi-Fi at a known place that he would have visited before. Places which do not fall into the 'Known' class are then classified based on the clusters they belong to. All unlabeled places in the 'Home' cluster of a user are labeled as 'NearHome' and all places in the 'Work' cluster are labeled as 'NearWork'. Finally, all remaining unlabeled places are labeled as 'Other'.

We experimented with kNN (k=3), J48 decision tree and Naive Bayes classifiers for classifying the places into the 8 classes as mentioned above. Table 10.6 shows the accuracies for these 3 classifiers on the training data collected as part of the field study (see Section 10.3.2) with 10 fold cross validation. J48 had a superior performance and faster training time than kNN and Naive Bayes. Hence we have implemented it for Semantic Place Classification.

10.4.4 Call Acceptance Prediction

A user's call log, as generated via the DeviceAnalyzer application, consists of timestamped events such as calls made and received, call duration, properties of the number

Feature	**Description or Representation**
Hashed time slot	Time of day and Day of week at the time of the call
Semantic Place	Semantic location of the user (Home/Work/Transport etc.)
Bluetooth count	Number of people around
Mobility state	Whether user is moving or stationary
Average call frequency	Average number of calls made (per hour) during each time slot
Average ring frequency	Average number of calls received (per hour) during each time slot
Average missed call frequency	Average number of calls missed (per hour) during each time slot
Average missed call rate	% of the calls received that are missed during each time slot
Average call duration	Average call duration during a time slot
Average call time difference	Average time difference (in minutes) between calls made during a time slot
Average ring time difference	Average time difference (in minutes) between calls received during a time slot
Average SMS out frequency	Average number of SMS sent (per hour) during a time slot
Average SMS in frequency	Average number of SMS received (per hour) during a time slot

Table 10.7: Spatial and temporal features used for call acceptance prediction

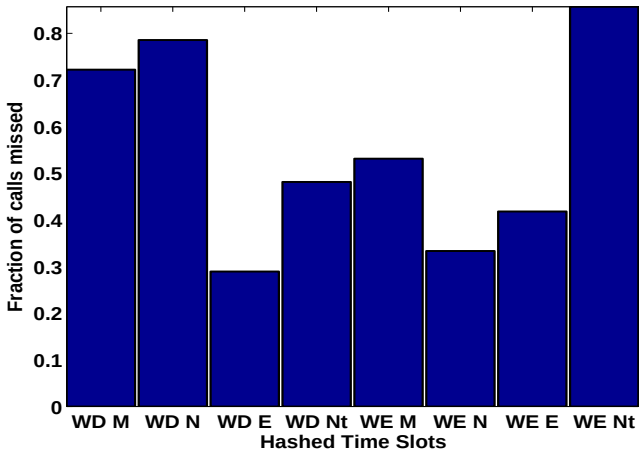


Figure 10.4: % of calls missed in different time slots for a randomly chosen user from our dataset

called etc. This log can be used to mine a user's communication behavior such as his availability or willingness to communicate (for instance, the user could be in a meeting at work and unavailable to receive a call) or the next contact that he would call. As discussed earlier, building such behavioral models for a user can enable a context-aware system to predict and anticipate his behavior and take proactive actions on his behalf without an explicit request from him. For instance, if a user is unlikely to accept an incoming call, the system can take appropriate actions such as messaging the callee after rejecting the call. In addition, the system can periodically provide shortcuts on the device screen for the next called contact.

In this work, we focus on one of these communication behavior models - predicting whether a user would accept an incoming call based on his/her communication behavior history. To this end, we employ several spatial and temporal features, device related features, and as well as features of the contact with whom the communication is occurring.

Table 10.7 shows the 13 spatial and temporal features we extract from a user's context history. We used practical knowledge to select these features as they define the usual communication behavior of the user at a place and during a time slot. Typically, people communicate very little during the night if they are sleeping at home or during the afternoon if they are at work. Also, many people return calls during their evening commute or when they reach home. Thus, the features that we extract are:

- Hashed time slot (Time of day and day of week) of the time of call - This feature is important as the time and day heavily influence a user's availability to attend a call. Figure 10.4 shows the fraction of calls missed (from all the received calls) in different time slots for a randomly chosen user from our dataset (WD = Weekday, WE = Weekend, M= Morning, N = Noon, E = Evening and Nt = Night). Clearly, week day morning and noon time slots have a high rate of missed calls. This is intuitive as a user could be at work on weekday afternoons and hence, reluctant to attend calls.

- Semantic Place - This represents the semantic location of the user at the time of call and is generated using the Semantic Place Classification model described in Section 10.4.3. This feature too is important. For instance, a user could be at work and unable to attend calls.

- Bluetooth count - This represents the number of people around the user at the time of call. For instance, if the user is in a meeting, he/she might not be able to attend calls.

- Mobility state - This represents whether the user is moving or stationary at the time

of call. For instance, if a user is driving, it would be difficult for him to attend a call.

- Average call frequency - This represents the average number of calls made by the user (per hour) during the time slot and is computed as

$$\frac{\text{Total number of calls made during the time slot}}{\text{Total number of time slot hours (for that slot) in user's call history}}$$

- Average ring frequency - This represents the average number of calls received by the user (per hour) during the time slot and is computed as

$$\frac{\text{Total number of calls received during the time slot}}{\text{Total number of time slot hours (for that slot) in user's call history}}$$

- Average missed call frequency - This represents the average number of calls missed by the user (per hour) during the time slot and is computed as

$$\frac{\text{Total number of calls missed during the time slot}}{\text{Total number of time slot hours (for that slot) in user's call history}}$$

- Average missed call rate - This represents the % of calls, received by the user, that are missed during a time slot and is computed as

$$\frac{\text{Total number of calls missed}}{\text{Total number of calls received}}$$

during the time slot

- Average call duration - This represents the average call duration for the user during a time slot and is computed as

$$\frac{\text{Total call duration}}{\text{Total number of calls}}$$

during the time slot

Feature	**Description or Representation**
Ringer mode	Device ringer mode status (normal/silent/vibrate)
Roaming state	Whether the device is on roaming or not
Service state	Whether the device is in service, out of service, receiving emergency calls only
Battery %	Battery level (%) of the device remaining at that time
App frequency	Average number of applications running on the device
Process frequency	Average number of processes running on the device

Table 10.8: Device related features used for call status prediction

- Average call and ring time difference - These represent the average time difference (in minutes) between calls made or calls received by the user during the time slot.

- Average SMS out frequency - This represents the average number of SMS sent by the user (per hour) during the time slot and is computed as

$$\frac{\text{Total number of SMS sent by the user during a time slot}}{\text{Total number of time slot hours (for that slot) in user's call history}}$$

- Average SMS in frequency - This represents the average number of SMS received by the user (per hour) during the time slot and is computed as

$$\frac{\text{Total number of SMS received by the user during a time slot}}{\text{Total number of time slot hours (for that slot) in user's call history}}$$

In addition, the device state at the time of communication can also determine a user's availability. For instance, if a device is on silent mode then the user may be in a meeting. If the device is on roaming, the user could be traveling. Similarly, if the user's device is

Feature	**Description or Representation**
Frequency of communication	Total number of calls or SMS exchanged with this contact
Normalized Frequency of communication	Fraction of the total number of calls or SMS that are with this contact
Relative Frequency of communication	Average number of calls or SMS (per day) exchanged with this contact
Average call duration	Average call duration for this contact
Missed call frequency	Total number of missed calls for this contact
Normalized missed call frequency	Fraction of the total number of calls missed that are with this contact
Relative missed call frequency	Average number of calls missed (per day) for this contact
Average missed call rate	% of the calls received, that are missed, for this contact
Number type	Whether number is unknown, toll free, mobile, fixed line etc.
Number validity	Whether number could be parsed and is local or foreign
Number country code source	Whether the number is from the same country, international etc.
Average communication time difference	Average time difference (in hours) between consecutive communication (such as call or SMS) with this contact

Table 10.9: Contact related features used for call prediction

running out of battery, then he may want to charge his device before any communication. These factors too can influence a user's decision to attend calls. Table 10.8 shows the 6 device related features that we employ. These are:

- Ringer mode - This represents the ringer mode status (normal/silent/vibrate) of the device at the time of call.
- Roaming state - This represents whether the device is on roaming or not at the time of call.
- Service state - This represents the service state of the device i.e. whether it is in service, out of service or receiving emergency calls only at the time of call.
- Battery % - This represents the battery level of the device remaining at the time of call.
- App frequency - This represents the average number of apps running on the device at the time of call.
- Process frequency -This represents the average number of processes running on the device at the time of call.

Moreover, who the the caller is can also influence a user's decision to attend a call. If the call is from an important contact, then the user might attend it irrespective of his current situation. Similarly, if a user communicates very frequently with a caller, he/she may be more willing to take the call. On the other hand, if the caller is unknown, then the user may decide to not take the call. Hence, we extract several features for the caller/contact who is calling. Table 10.9 shows the 12 contact related features we use:

- Frequency of communication - This represents the strength of communication (in the form of total number of calls or SMS exchanged) with this contact.

- Normalized Frequency of communication - This represents the % of the total communication, of the user, that is with this contact (computed as the fraction of the total number of calls or SMS exchanged that are with this contact).

- Relative Frequency of communication - This represents the average number of calls or SMS exchanged per day with this contact and is computed as

$$\frac{\text{Frequency of communication with the contact}}{\text{Total number of days in user's communication history}}$$

- Average call duration - This represents the average call duration for this contact and is computed as

$$\frac{\text{Total call duration}}{\text{Total number of calls}}$$

- Missed call frequency - This represents the total number of missed calls for this contact

- Normalized missed call frequency - This represents the % of calls missed, by the user, that are for this contact (computed as the fraction of the total number of calls missed).

- Relative missed call frequency - This represents the average number of calls missed per day for this contact and is computed as

$$\frac{\text{Missed call frequency for the contact}}{\text{Total number of days in user's communication history}}$$

Accuracy	J48	kNN	Naive Bayes	Random Forest	SMO
	0.87	0.77	0.67	**0.88**	0.83

Table 10.10: Comparison of accuracies of various classifiers on the DeviceAnalyzer validation set for Call Prediction

- Average missed call rate - This represents the % of the calls received from this contact that are missed and is computed as

$$\frac{\text{Total number of calls missed}}{\text{Total number of calls received}}$$

for this contact

- Number type - This represents the number type i.e. whether it is unknown, toll free, mobile, fixed line etc.

- Number validity - This represents the number validity (whether it could be parsed) and is local or from another country.

- Number country code source - This represents the country code source i.e. whether the number is from the same country, international etc.

- Average communication time difference - This represents the average time difference (in hours) between consecutive communication (such as a call or SMS) with this contact.

To build this model, we extracted these 31 features (13 spatial and temporal, 6 device related and 12 contact related) from the context history of all the users in the DeviceAn-

Feature	Description or Representation
Hashed time slot	Time of day and Day of week
Semantic Place	Semantic Location of the user (Home/Work/Transport etc.)
Mobility state	Whether user is moving or stationary
Battery Level status	Whether battery is very low, low etc.
Charging state	Device charging state - connected (via ac/usb) or disconnected

Table 10.11: Device charging behavior related features

alyzer validation set (see Section 10.3.1). For each user in the validation set, we labeled each instance of an incoming call as either of the 2 classes - 'Call Taken' if the call is accepted or 'Call Missed' if the call is not accepted by the user. Since the call log data can be unbalanced for many users (with more instances for one class than the other), we applied a resampling filter to introduce a bias towards a uniform class distribution. This filter produces a random subsample of a dataset using either sampling with replacement or without replacement.

We experimented with 5 machine learning techniques for this model - J48 Decision Tree, kNN (k=3), Naive Bayes classifier, Random Forest (10 trees) and SMO. These algorithms were run on the validation set with 10 fold cross validation. Table 10.10 shows the comparison of accuracies for these 5 algorithms. Random Forest had the best performance on the data and hence, we have implemented it for Call Acceptance Prediction model.

10.4.5 Device Charging behavior modeling

Ferreira [178] et al., analyzed the device charging behavior (such as charging time periods, usual battery levels, levels at which charging is initiated, and preferred mode of charging) of 4035 participants over a course of 4 weeks. Their study showed that most users follow a pattern in charging their devices. The daily average battery level across all users was 67%. The 2 major charging schedules for most users were between 6 and 8 pm in the evening, when the battery level was at 40%, or between 1 and 2 am in the night when the battery level was at 30%. Also, users preferred to charge their phones via ac for longer periods and usb for shorter periods.

As suggested by this work, we attempted to build such charging behavior models for all users in our DeviceAnalyzer dataset. Table 10.11 shows the 5 features we employ for modeling users' device charging behavior. These include:

- Hashed time slot (Time of day and day of week): This feature is important as the time or day can influence a user's inclination to charge their device. For instance, many users prefer to charge their devices overnight.

- Semantic Place: This represents the semantic location of the user and is generated using the Semantic Place Classification model described in Section 10.4.3. This feature is intuitive as most people charge their devices at 'Home', 'Work' or an indoor location.

- Mobility state: This represents whether the user is moving or stationary. Often, people charge their devices in their cars and hence, this feature is important.

- Battery Level: This is one of the most important features. As suggested by [178], most users avoided extremely low battery levels and there is a significant correlation between the battery level and initiation of charging. Since the device battery level is a numeric value, we discretized it into certain ranges and used the range as a feature. For instance, a battery level of 0 - 20% maps to a range of 'VeryLow', from 20 - 40 % is 'Low', from 40 - 60% is 'Average', from 60 - 80% is 'High' and from 80 - 100% is 'VeryHigh'.

- Charging state: Finally, this represents the device's charging state - if it is getting charged (via ac/usb) or is disconnected.

These features were used to generate the device charging behavior instances for all the users in the DeviceAnalyzer dataset. To address any imbalance in the dataset, we applied a resampling filter to introduce a bias towards a uniform class distribution. We also filtered out duplicate instances.

To model the charging behavior of users, we employed association rule mining to mine associations between the various features - time period, semantic place, mobility state, battery level and charging status of the device. A significant benefit of association rules is that they can determine strong associations between different attribute values. Thus, they can predict any attribute, not just the class, which gives them the freedom to predict combinations of attributes [179]. We employed two state of the art association rule mining algorithms - *Apriori* [180] and *PredictiveApriori* [181]. Apriori iteratively reduces the minimum support until it finds the required number of rules with the given minimum confidence. We set the minimum confidence level as 0.6 and the number of

rules to generate as 5. The minimum support is varied from 1.0 to 0.1. PredictiveApriori combines confidence and support into a single measure of predictive accuracy and finds the best *n* association rules in order. We set *n* as 5 in our experiments. Some of the sample rules generated for the users in our dataset are:

- If 'Time slot = Weekday Night' then 'Charging State = charging'.
- If 'Battery Level = Very Low' and 'Semantic Place = Home' then 'Charging State = charging'.
- If 'Charging Status=charging' and 'Time slot=Weekend Noon' then 'Mobility State=Stationary'.
- If 'Time slot = Weekend Night' and 'Battery Level=Low' then 'Charging State = charging'

Clearly, these rules are intuitive and support the findings in [178]. As evident, users in our dataset charge their phones during weekday or weekend nights, when the battery levels are low and they are at home. Learning such rules, from a user's charging behavior, can enable the design of intelligent prompting mechanisms to proactively remind users to charge their devices.

10.5 Evaluation

10.5.1 Methodology and Goals

As stated earlier, the Learning Engine of the Rover II middleware is responsible for building diverse user behavioral models in order to predict a user's behavior and enable the

Model	Accuracy (%)
Semantic Place Classification	89.8
Mobility State Classification	88.02
Call Acceptance Prediction	89.1

Table 10.12: Accuracy of various algorithms (%)

middleware to take proactive actions on the user's behalf. Hence, the primary goal of our evaluation is to determine how *accurately* the various algorithms and approaches, implemented as part of the engine, model users' behavior. To this end, we evaluate the approaches that we have implemented currently (for Mobility State Classification, Semantic Place Classification and Call Acceptance Prediction) on the entire dataset of 200 users from the DeviceAnalyzer data collection as well as the dataset collected from the field study (see Section 10.3) with 10 fold cross validation. We perform an accuracy analysis of these approaches and report our results.

10.5.2 Accuracy measures used

Since these models involve binary and multi-label classification, we define accuracy for them as:

$$a = \frac{T \cap P}{T \cup P}$$

where T is the set of ground truth and P is the set of predicted labels for all instances.

Accuracy	Semantic Place Label									
	Home	Work	InRec	OutRec	Transport	InKnown	OutKnown	NearHome	NearWork	Other
	1.0	0.95	0.93	0.78	0.81	0.94	0.9	0.96	0.95	0.76

Table 10.13: Individual place accuracies for Semantic Place Classification

10.5.3 Results

Table 10.12 shows the accuracy results for all the approaches that have been implemented as part of the Learning Engine in Rover II. As evident, they achieve high accuracy.

10.5.3.1 Semantic Place Classification

Table 10.13 shows the accuracies of our Semantic Place Classification algorithm for individual place labels. The 'Home' location is labeled correctly for all users. Also, places such as 'Work', 'IndoorKnown and 'OutdoorKnown' are classified with high accuracy. Similarly, location based clustering allows us to label places that are 'NearHome' and 'NearWork' accurately. While 'IndoorRecreational' places are labeled correctly with high accuracy, 'OutdoorRecreational' and 'Transport' related places are classified with reasonable accuracy. A possible reason for this could be that at both these places, a user exhibits similar behavior - more movement and being surrounded by several people at different times. Hence, we observed that the correct labels for some of these places were interchanged. Overall, our algorithm has an accuracy of 89.8 %.

Ground Truth	Predicted labels	
	Stationary	Moving
Stationary	0.85	0.15
Moving	0.07	0.93

Table 10.14: Confusion matrix for Mobility State Classification

Ground Truth	Predicted labels	
	Call Taken	Call Missed
Call Taken	0.91	0.09
Call Missed	0.13	0.87

Table 10.15: Confusion matrix for Call Acceptance Prediction

10.5.3.2 Mobility State Classification

Table 10.14 shows the confusion matrix for Mobility State Classification. Overall, this approach has an accuracy of 88.02% and can accurately classify the user's mobility state.

10.5.3.3 Call Acceptance Prediction

Table 10.15 shows the confusion matrix for Call Acceptance Prediction. Our approach achieves an accuracy of 89.1% and can accurately determine whether a user would accept an incoming call.

10.6 Related Work

Since our work involves building several user behavior models for semantic place classification, mobility state classification, call acceptance prediction etc. we have categorized the related work into different sections. We differentiate our work from them and identify their shortcomings. Please note that while most of these works have focused on building a single isolated user model, we have developed a unified infrastructure for building several such models, from large-scale smartphone data, as part of the generic Rover II context-aware middleware.

10.6.1 Semantic Place Classification

Previous research in Semantic Place Classification has used the MDC dataset [175, 176] for classifying semantic places into 10 categories specified by the challenge organizers (Home, Home of a friend, Work, Transport, Indoor sports, Outdoor sports etc.). Huang et al. [182] used 54 features and experimented with classifiers such as kNN, Support Vector Machines (SVM) and J48 along with an ensemble of the three to achieve a final accuracy of 65.77% with 10 fold cross validation. Montoliu et al. [183] employed a multi-coded class based multiclass evaluation rule that combines classification results of the binary classifiers such as kNN and SVM. They achieve an accuracy of 73.26% with 2 fold cross validation. Zhu et al. [184] compare the performance of Logistic Regression, SVM, Gradient Boosted Trees (GBT), and Random Forest to achieve an accuracy of 65.3% with GBT and 10 fold cross validation. Lex et al. [185] used 32 features and employed Random Forest as well as SVM. They achieve an accuracy of 85.4% with RandomForest and 10-

fold cross-validation. While some of the features employed by these works overlap with ours, a direct comparison with them is not possible due to difference in the datasets, quality of location information, and semantic place labels. However, the overall accuracy of our Semantic Place Classification algorithm is higher.

More recently, Bao et al. [186] have experimented with the DeviceAnalyzer dataset and employed features such as Wi-Fi visibility and connectivity, and cell locations to identify semantic places such as Home, Work and Commute as well as the transitions between them. However, they do not present any evaluation or results.

10.6.2 Mobility State

While previous work in activity recognition (including SenseMe [9], CenceMe [45] and Jigsaw [46]) has focused on fine-grained activity recognition such as Walking, Running, Driving etc. , it is not possible to do that with the DeviceAnalyzer dataset because of the limited accelerometer data available and its sampling rate (see Section 10.4.1). Hence, due to this inherent limitation of the used dataset, our work focuses on determining two mobility states for the user - Moving and Stationary.

10.6.3 Call Acceptance Prediction

Husna et al. [187] and Zhang and Dantu [188] have proposed 'quantifying' the presence of users and their availability for phone calls. They used 3 basic features such as time of the day, day of the week and location, and predicted the user's availability and likelihood of accepting phone calls based on a weighted average of these feature values. They evaluated

their approach on the data of 10 users from the Reality Mining dataset [189] and appear to have achieved an average accuracy of around 50%. This makes their approach unsuitable for real world applications. While our approach shares similar goals with these works, we use 31 varying contextual features for call acceptance prediction. In comparison to these works, our approach has a much higher accuracy (89%) for 200 users.

10.6.4 Device charging behavior

Existing literature [190–192]has explored correlations between device power consumption and a user's context (such as location, time and device usage) as well as device related features (such as CPU utilization, wireless state, IO and data transfer) in order to perform context-aware and personalized device battery lifetime and level prediction. However, majority of this research is driven by the need for effective energy management on devices. To the best of our knowledge, no other work has proposed modeling a user's charging behavior to generate patterns and rules in order to design intelligent prompting or reminder mechanisms.

10.6.5 User modeling from mobile phone data

The Reality Mining project [189] was one of the first attempts at mobile phone data collection. This project collected data from 100 users over the course of 9 months. However, a large amount of data collected in this study was through self reported surveys. Eagle and Pentland [193] applied principal component analysis to users' location data from this dataset to identify primary routines or eigenbehaviors (such as sleeping late on weekends

or going out on weekend nights) for individuals and their social circle. They also inferred community affiliations by clustering individuals. Altshuler et al. [194] used the same dataset for predicting individual traits of users such as their nationality, gender and social links such as life partners. Farrahi [195] explored the use of topic modeling for discovering location driven routines such as 'going to work' or 'staying home at evening' and experimented with this approach on a subset of the Reality Mining dataset. In contrast, we attempt to build diverse behavioral models that capture different aspects of users' behavior (such as place visitation patterns, calling patterns and device charging behavior) from large-scale data collected over several years.

Srinivasan et al. [196] proposed an association rule mining based algorithm to mine co-occurring context patterns of users on their devices. A limitation of their work is that though they determine correlations between inferred contexts of users such as 'AtHome' and 'ReadComics', they do not explain how the inference is performed. Moreover, their approach is restricted to pattern mining only and cannot predict or classify user behavior which is often more important and accurate. On the other hand, we employ several machine learning techniques including classifiers and association rules to build user behavioral models.

Chapter 11: **Enabling Proactivity in Context-aware Middleware Systems by means of a Planning Framework based on HTN Planning**

The context-aware middleware can utilize the behavioral models of users in order to predict their future behavior and proactively take actions on their behalf. These actions could include tasks such as sending an email on the user's behalf, calling a phone number, changing the device mode (say from silent to ringer) based on his situation, installing or starting an application on the user's device etc. To this end, we present a new paradigm for enabling proactivity in context-aware middleware systems by means of a Planning Framework based on HTN planning. In this chapter, we present the design and implementation of such a framework [14] within the infrastructure of the Rover II context-aware middleware.

11.1 Introduction

Today's context-aware systems tend to be 'reactive' or pull based - the user requests or queries for some information and the system responds with the requested information. These systems provide personalized and relevant information by filtering the information

retrieved based on the user's preferences or limited context such as time, location, or web history. Such systems, once queried, could return a list of restaurants ordered by food preferences and sometimes recent browsing history. However, none of the systems anticipate the user's intent and behavior, or take into account his current events and activities to act 'proactively' and push relevant information to the user.

The initial notion of proactive computing, as proposed by Tennenhouse [197] and Want et al. [198], focused on human -supervised operations where the user stays out of the loop as much as possible until he is required to provide guidance in critical decisions. Tennenhouse [197] also stated that a fundamental goal of proactive computing is to enable autonomy in ubiquitous systems. However, Want et al. [198] specified key differences between proactive and autonomic computing and outlined several principles underlying proactive computing. Some of these are anticipation, context-awareness and statistical reasoning. Salovaara and Oulasvirta [199] discussed the general concept of proactive computing and suggested that a system can act proactively if it can hypothesize what its user's goals are.

Thus, for a ubiquitous or context-aware system to be effectively proactive, it is crucial that it tracks and predicts user intent [200] in order to take actions on the users' behalf without explicit requests from them. We term these systems as *Proactive context-aware systems*. These systems continuously sense and anticipate users' behavior - they acquire data from multiple sources and sensors, and then analyze the data in order to learn and predict users' behavior. Once the user behavior has been predicted, the system can proactively take actions on behalf of the users without an explicit request. These actions can include sending an email on the user's behalf, calling a phone number, changing the device

mode (say from silent to ringer) based on his situation, or even booking movie tickets for the user. From this discussion, two fundamental capabilities of proactive context-aware systems emerge: *prediction* and *autonomy*.

In our previous work [13], we have addressed the first aforementioned capability of a proactive context-aware system - modeling and predicting user behavior, as part of the Rover II context-aware middleware [16,17]. In this work, we address the second key capability required by a context-aware system to act proactively - acting autonomously without an explicit user request. To address it, we present a new paradigm for enabling proactivity in context-aware middleware systems by means of a Planning Framework based on HTN planning. This framework is based on a predicate model of ubiquitous computing where the state of the context-aware system is represented as a set of variable bindings. It receives information about a task or activity that needs to be performed on the user's behalf (which the user may have requested explicitly or implicitly) and generates a plan to achieve it. It utilizes the current context of the user, and internal and external information sources available to the system in order to determine the sequence of actions that should be performed in order to achieve the task.

The use of AI planning enables the system to decide, dynamically, how best to achieve user goals. It relieves users from the burden of having to know exactly what actions or tasks can or cannot be performed by the system and how to perform those actions. More importantly, since the system plans the sequence of actions to achieve the user's goals dynamically, it can adapt more easily to changing context and availability of resources. This also allows the system to handle faults that may occur while achieving the user's goals gracefully and with minimum user intervention.

Thus, our contributions in this work are:

- We propose the paradigm of enabling proactivity in context-aware middleware systems by means of HTN Planning.
- We present the design of a Planning Framework within the infrastructure of our intelligent context-aware middleware called Rover II.
- We implement this framework and evaluate its utility with several use cases.
- We also highlight the benefits of using such a framework in dynamic ubiquitous systems.

The rest of the chapter is organized as follows: Section 11.2 describes a scenario to motivate the need of enabling proactivity in a context-aware system. Section 11.3 provides an overview of HTN Planning. Section 11.4 describes the design of our planning framework and algorithm while Section 11.5 describes the implementation details and use cases. We discuss related work in Section 11.7.

11.2 Motivating Scenarios

In order to motivate the proactivity paradigm, we describe a simple scenario.

> A user is running late for a meeting with a colleague. The context-aware system infers this delay from his current location and from his meeting schedule (date, time, location, agenda, participants) marked in his calendar. It proactively communicates a message to the colleague with whom the meeting is scheduled to take place informing him/her of the delay.

There may be multiple ways of performing this task and some ways may be better than others because of the user's context or availability of information sources. In addition, the best way of achieving the goal may also change with time because of dynamically changing context. Also, depending on the current state of the system, different actions may need to be taken to achieve it. Hence, it is not easy to statically specify how the task is to be performed. Thus, new techniques are required that can dynamically perform such tasks for users.

Our proposed planning framework helps address this complexity and dynamism of context-aware systems. The framework analyzes the different ways in which a task can be achieved based on the available sources and user's context. It then determines the most feasible way of performing the task and generates a sequence of actions required to achieve it. In the aforementioned scenario, there are two possible ways of communicating with the user's colleague:by SMS or by email. SMS might be a faster way to reach the colleague as he/she may be driving or walking and may not check email soon. However, if the system doesn't have access to the phone number of the colleague then email is the only solution to reach him/her. Hence, the planning framework has to take all these factors into consideration and generate a sequence of actions that will be executed in order to perform this task. From the scenario, it is evident that an AI Planning technique such as HTN Planning can be applied to facilitate proactivity.

11.3 Hierarchical Task Network Planning

Hierarchical Task Network (HTN) planning is an Artificial Intelligence (AI) planning technique [201]. The objective of an HTN planner is to produce a sequence of actions that perform some activity or task. The description of a planning domain includes a set of operators or primitive tasks and also a set of methods, each of which is a prescription for how to decompose a compound or non-primitive task into subtasks (smaller primitive tasks). Given a planning domain, the description of a planning problem will contain an initial state and a partially ordered set of tasks to accomplish.

HTN Planning proceeds by using the methods to decompose tasks recursively into smaller and smaller subtasks, until the planner reaches primitive tasks that can be performed directly using the planning operators. Thus, for each non-primitive task, the planner chooses an applicable method, instantiates it to decompose the task into subtasks, and then chooses and instantiates methods to decompose the subtasks even further if required. If the plan later turns out to be infeasible, the planning system will need to backtrack and try other methods.

HTN Planning requires well-conceived and well-structured domain knowledge. Such knowledge contains rich information and guidance on how to solve a planning problem. This structured and rich knowledge gives a primary advantage to HTN planners in terms of speed and scalability when applied to real-world problems. Examples of HTN Planners include Nets Of Action Hierarchies (NOAH) [202], System for Interactive Planning and Execution (SIPE) [203], Universal Method Composition Planner (UMCP) [204] and Simple Hierarchical Ordered Planner (SHOP) [205].

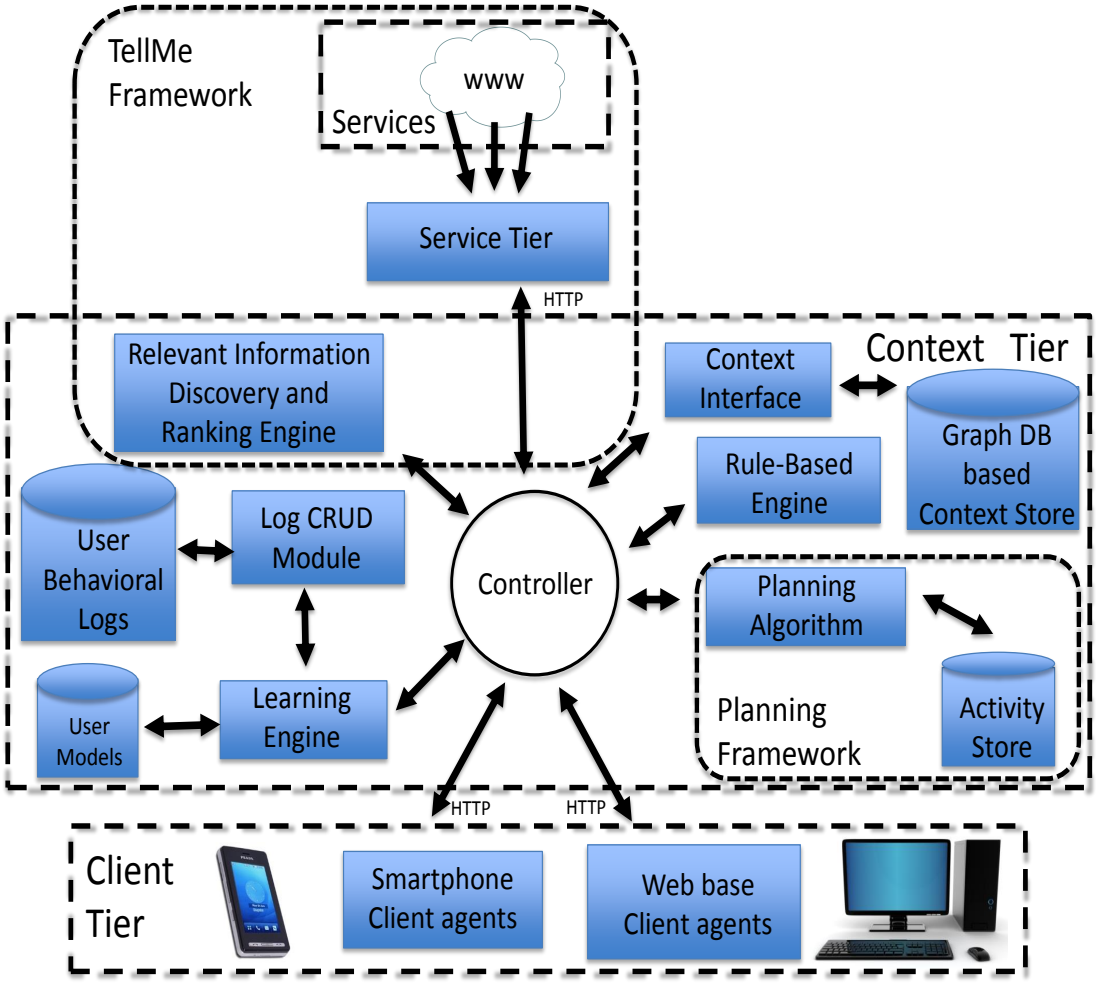


Figure 11.1: Architecture of the Rover II context-aware middleware

11.4 Planning framework of the Rover II context-aware middleware

The Rover II context-aware middleware [16, 17] is a generic middleware, which serves as an integration platform for mobile and desktop applications. It can store and retrieve contextual information, as well as learn and store user behavior models. Figure 11.1 shows the architecture of Rover II context-aware middleware It consists of several components including a main Controller module (which controls the flow of information among the various components), a Learning Engine (which learns patterns from user's behavior in order to predict a user's intent or goal), a Relevant Information Discovery and Ranking Engine (which determines what information will be relevant to the user's current situation), an Activity Store (which defines what activities the system can perform on the user's behalf) and a Planning algorithm (which generates the sequence of activities that should be performed in order to accomplish a task). A complete description of this system and its architecture is beyond the scope of this chapter. Here, we focus on the planning framework of Rover II. This framework (as shown in Figure 11.1) consists of two components:

11.4.1 Planning Algorithm

We employ the Pyhop HTN planning algorithm[1], which is a Python implementation of SHOP [205]. Pyhop is a HTN planner that uses hierarchical decomposition of tasks for planning. It is widely used in hundreds of projects worldwide with applications in in-

[1] `https://bitbucket.org/dananau/pyhop`

dustry, academia and government labs. Like other HTN planners, Pyhop is configurable i.e. its planning engine is domain-independent, but the HTN methods may be domain-specific, and the planner can be customized to work in different problem domains by giving it different sets of HTN methods. As mentioned earlier, this ability to use domain-specific problem-solving knowledge can dramatically improve a planner's performance, and sometimes make the difference between solving a problem in exponential time and solving it in polynomial time.

In Pyhop, a task is a symbolic representation of an activity to be performed in the real world, for instance, 'Book a flight'. Instead of spending time on each individual operator, Pyhop uses its in-built hierarchical structure to avoid exponential explosion. Rather than searching through the entire state-space to find the plan, it aims at performing certain tasks that meets predefined conditions. As commonly done in HTN Planning, Pyhop uses abstract tasks to start a plan and then decomposes them into smaller sub tasks. A task can be primitive or non-primitive. A primitive task corresponds to a basic action that can be directly performed in the real world. On the other hand, a non-primitive or compound task is composed of other primitive tasks and cannot be performed directly to the real world. It first needs to be decomposed into simpler tasks until primitive tasks are found.

Algorithm 10 shows the pseudo code for the Pyhop HTN Planning algorithm. It takes the following as input:

- An initial state - This is a description of the current situation.
- List of tasks - These describe the activities to perform
- Methods - These are parameterized descriptions of possible ways to perform a com-

Algorithm 10: HTN Planning algorithm

Input: Initial state S_0, list of tasks T, methods M and operators O

Output: Plan P

Initialize tasklist T to contain the toplevel task in the hierarchy;

Initialize plan P = $\emptyset$;

Initialize state S = S_0;

procedure SEEKPLAN(state S, tasklist T, plan P)

if $T = \emptyset$ **then**

return plan;

else

Task t ←first task from tasklist;

if $t \in O$ **then**

if *S satisfies the pre conditions of t* **then**

S' ← t applied to S;

T ← T - {t};

P ← P + {t};

SeekPlan(S', T, P);

return P (if found);

else

return failure;

else if $t \in M$ **then**

foreach *relevant method* **do**

τ ← subtasks of t;

SeekPlan(S, τ + T - {t}, P) ;

return P (if found);

end

else

return failure;

return *Plan;*

pound task by performing a collection of subtasks. There may be more than one method for the same task.

- Operators - These are parameterized descriptions of what the primitive actions can achieve.

The Pyhop algorithm makes use of backtracking. Backtracking is a general algorithm for finding all (or some) solutions to a computational problem, that incrementally builds candidates to the solutions, and abandons each partial candidate c (”backtracks”) as soon as it determines that c cannot possibly lead to a valid solution.

The algorithm recursively checks if it can find a plan for a given set of goal tasks (the planning problem). First it checks if any tasks are left. It is done when no tasks are left, but needs to continue planning if any tasks remain. It then selects the next task and checks if an operator matches the task. If no operator matches the task, the planner looks at the methods. There can be multiple HTN methods to accomplish the same task. If the planner found either an operator or method for a given task, and they did not fail (e.g. when preconditions are not met), the search method is called again for the next task (and thus it is recursive) until a full plan is either found or not. If no plan is found, failure is returned.

11.4.2 Activity Store containing Domain Description

As mentioned in Section 11.3, HTN Planning requires domain knowledge. This domain knowledge is specified in our Planning Framework in the form of a domain description consisting of tasks, methods, and operators. The tasks are specified in the form of activi-

ties that needs to be performed on the user's behalf such as 'Book Movie'.

The simplest version of a method has three parts: the task for which the method is to be used, the precondition that the current state must satisfy in order for the method to be applicable, and the subtasks that need to be accomplished in order to accomplish that task. For instance, one method to accomplish the task 'Book Movie' for a user is 'Book movie via smartphone app'. This involves sub tasks such as 'searching for the desired movie', 'checking availability of desired date and time', 'checking availability of the required number of seats', 'booking the seats' and 'paying for the tickets'.

Each operator indicates how a primitive task can be performed. Each operator description includes the operator's name and a list of parameters, a precondition expression indicating what should be true in the current state in order for the operator to be applicable and the effects of the operator on the current state if it is applied. For the above mentioned example, the primitive task for 'checking for seats availability' would involve checking if the number of available seats for the desired movie is $>$ the number of seats required by the user.

11.5 Implementation and Use Cases

In this section, we present the implementation of our Planning Framework (developed as part of the Rover II context-aware middleware) and its components using specific technologies. The Pyhop planning algorithm is implemented in Python. We have implemented the Activity Store containing domain knowledge as a python module which contains different methods and operators to achieve tasks on the user's behalf. This

framework has been integrated with the Rover II middleware using Jython[2], which is an implementation of Python seamlessly integrated with the Java platform.

Some sample use cases that we have implemented are:

11.5.1 Communicating with a user's contact

Recall the scenario mentioned in Section 2. The system can communicate with the colleague of the user via two possible media: SMS and Email. This use case is implemented as follows:

11.5.1.1 Task

The high-level task that needs to be achieved in this use case is 'Communicate'.

11.5.1.2 Methods

There are two methods to achieve this task:by SMS or by email.

```
def communicate _by _sms (state, sender, recipient, subject, text, status):
    if ((sender in state.contacts) and (recipient in state.contacts)):
    return [('create _sms', sender, recipient), ('search_phonenum', sender, recipient), ('set _sms _subject', subject), 'set _sms
_text',text), ('send _sms', status)]
return False

def communicate _by _email (state,sender,recipient,subject,text,status):
    if ((sender not in state.contacts) or (recipient not in state.contacts)) and ((sender in state.emaillist) and recipient
in state.emaillist)):
    return [('create _email', sender, recipient), ('search _emailadd', sender, recipient), ('set _email _subject', subject), ('set
```

[2] http://www.jython.org/

_email _text', text), ('send _email', status)]

return False

As shown, if the phone number of both the sender i.e. the user and the recipient i.e. his colleague are in the contacts list, the system would send an SMS to the colleague. Otherwise, it would send an email.

11.5.1.3 Operators

The method to send an SMS (communicate _by _sms) can be decomposed into the following primitive sub tasks or operators:

```
def create _sms(state, sender, recipient):
    if not sender or not recipient:
        return False
    else:
        state.sms['sendername']=sender
        state.sms['recipientname'] = recipient
        return state

def search _phonenum(state, sender, recipient):
    state.sms['senderphone']=state.contacts[sender]
    state.sms['recipientphone'] = state.contacts[recipient]
    return state

def set _sms _subject(state, subject):
    if not subject:
        return False
    else:
        state.sms['subject'] = subject
        return state
```

```
def set _sms _text(state, text):
    if not text:
        return False
    else:
        state.sms['text'] = text
        return state


def send _sms(state, status):

    state.sms['status'] = status

    return state
```

Similarly, the method to send an email (communicate _by _email) can be decomposed into the following primitive sub tasks such as:

```
def create _email(state, sender, recipient):
    if not sender or not recipient:
        return False
    else:
        state.email['sendername']=sender
        state.email['recipientname'] = recipient
        return state


def search _emailadd(state, sender, recipient):
    state.email['senderphone']=state.emaillist[sender]
    state.email['recipientphone'] = state.emaillist[recipient]
    return state


def set _email _subject(state, subject):
    if not subject:
        return False
```

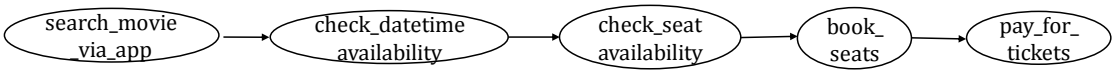


Figure 11.2: Plan generated by the Planning framework for booking a movie for the user

```
        else:
            state.email['subject'] = subject
            return state


def set _email _text(state, text):
    if not text:
        return False
    else:
        state.email['text'] = text
        return state


def send _email(state, status):
    state.email['status'] = status
    return state
```

These primitive subtasks involve creating the SMS or email, searching for both the sender's and recipient's phone numbers or email addresses, setting the SMS or email subject and text and finally, sending it to the recipient.

11.5.1.4 Initial state

The initial state to the Planning algorithm includes the contacts and the email list of the user, as well as the subject and text of the message.

11.5.1.5 Generated plan

The planning algorithm uses the defined domain knowledge (consisting of methods and operators) and current context to generate the appropriate and feasible sequence of actions that need to to be executed in order to accomplish this task. Figure 11.3 shows a sample generated plan in the case where the phone numbers of both the sender and recipient are present in the contacts list of the user. For legibility, the parameters to these operators have not been shown.

11.5.2 Booking a movie

We now consider another scenario where the context-aware system needs to book movie tickets for the user. The system may detect from a user's calendar that he/she desires to watch a particular movie on a certain day and after a certain time, say Friday evening. It then proceeds to perform the task of booking the movie proactively. It could also be the case that the user explicitly requests the system to book the movie tickets for a particular date and time.

11.5.2.1 Task

The high-level task that needs to be achieved in this use case is 'Book a movie'.

11.5.2.2 Methods

There can be two ways of achieving this task: booking the movie via an app or via a website. The methods representing these two ways are:

```
def book_movie_through_app(state,a,movie,dateandtime, seats):
    if ('Fandango' in state.apps):
    return [('search_movie_via_app', a, movie),('check_datetimeavailability', movie, dateandtime),('check_seatavailability',
movie, seats),('book_seats', a, movie, seats),('pay_for_tickets', a, movie)]
    return False

def book_movie_through_website(state,a,movie,dateandtime, seats):
    if ('Fandango' not in state.apps):
    return [('search_movie_via_browser', a, movie),('check_datetimeavailability', movie, dateandtime),('check_seatavailability',
movie, seats),('book_seats', a, movie, seats),('pay_for_tickets', a, movie)]
    return False
```

11.5.2.3 Operators

These methods can be decomposed into the following primitive sub tasks or operators:

```
def movie_tickets_cost(state,movie,seats):
    return state.ticket[movie]*seats

def search_movie_via_app(state,a,movie):
    if (movie not in state.movies['Fandango'] or state.ticket[movie]¿state.money[a]):
        return False
    else:
        return state

def search_movie_via_browser(state,a,movie):
    if (movie not in state.theater['AMC'] or state.ticket[movie]¿state.money[a]):
        return False
    else:
        return state
```

```
def check _datetimeavailability(state,movie,dateandtime):
    if state.moviedatetimes[movie]¿dateandtime:
        return state
    else:
        return False

def check _seatavailability(state,movie,seats):
    if state.movieseats[movie]¿seats:
        return state
    else:
        return False

def book _seats(state,a,movie,seats):
    state.movieseats[movie] = state.movieseats[movie] - seats
    state.owe[a] = movie _tickets _cost(state,movie,seats)
    return state

def pay _for _tickets(state,a,movie):
    if state.money[a] ¿= state.owe[a]:
        state.money[a] = state.money[a] - state.owe[a]
        state.owe[a] = 0
        state.booking[movie]='Booked'
        return state
    else: return False
```

11.5.2.4 Initial State

The initial state to the Planning algorithm includes the user's current location, movies showing in theaters near the user's current location, the scheduled dates and times of their shows, number of seats available, and the cost of their tickets. An important point to note here is that even though this is domain knowledge, the information is extracted

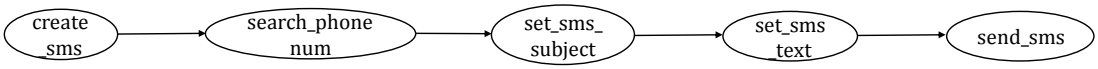


Figure 11.3: Plan generated by the Planning framework for communicating with the user's colleague

from external sources such as web search engines or other websites. In addition, the user request must contain the name of the movie, the preferred date and time and the number of tickets to be booked.

11.5.2.5 Generated Plan

The planning algorithm uses the defined domain knowledge (consisting of methods and operators) and current context to generate the appropriate and feasible sequence of actions that need to to be executed in order to accomplish this task. Figure 11.2 shows a sample plan generated by our framework for booking a movie on the user's behalf.

11.6 Advantages of the Planning Framework

Our planning framework offers a number of advantages:

- Minimal user intervention - The framework automatically generates a sequence of actions required to achieve a user's task based on available information and resources without requiring any user intervention. Human guidance is required only at the crucial step when some information is needed from the user. Thus, the users do not have to worry about knowing how exactly to perform certain kinds of tasks in a ubiquitous system, what kinds of services and applications are present in the

system and how to interact with them. They can leave the intricate details of performing tasks as well as handling failures to the planning framework.

- Fault-tolerance - If an action fails, the framework detects it and backtracks by retrying actions or by replanning and taking another path to achieve the same goal. The replanning approach to failure recovery works because ubiquitous environments are dynamic and constantly changing. Moreover, there are usually several ways of achieving a task.

- Adaptable to varying context - The planning framework generates the plan to achieve a task taking into consideration the current context of the user and environment. It finds out what information and resources are currently available in the system and then tries to achieve the goal using these resources.

11.7 Related Work

While AI Planning has been successfully applied in several domains such as robotics and games, it has not been employed in context-aware systems or ubiquitous systems. Most of the existing work in this domain has focused on web services composition using AI Planning [206–210].

Ranganathan et al. [211] developed a STRIPS-based planning framework that used state space planning for a meeting room domain. The framework was based on a predicate model of pervasive computing where the state of the environment and its elements were represented as a set of predicates. The users were allowed to specify the goals that had to be achieved such as starting a presentation. The actions are either an invocation of a

method on a service, device or application and were represented in terms of their pre-conditions and effects. A utility function is used to determine the best goal state.

Chapter 12: **Conclusion and Future Work**

We believe we have provided a major push in the direction of achieving the paradigm of **Proactive Context-aware Computing**. As described in this dissertation, this paradigm entails the following capabilities for a context-aware system:

1. Determining relevant information - This capability is the cornerstone of context-aware computing. In today's world, with the abundance of information available to us, information overload can easily happen. Hence, it is imperative that the system retrieves and displays only that information which is relevant to the user's task at hand.

2. Personalization - This is achieved by acquiring a user's context (needs, preferences, etc.) through implicit or explicit means and using it to filter the relevant information.

3. Timeliness - The system can achieve timely information delivery by providing the personalized and relevant information to the user at a time when he needs it and can act upon it.

Further, this paradigm poses several key challenges pertaining to:

- Recognizing and anticipating a user's current and future context, activities and sit-

uations,

- Determining relevant information for those situations, and further personalizing this relevant information,

- Providing the user with the personalized and relevant information in a timely manner.

12.1 Summary of Contributions

This dissertation made the following contributions in order to realize the vision of **Proactive Context-aware Computing** and address the challenges posed by it:

1. It presented the design and implementation of a proactive context-aware middleware which has been developed by enhancing an existing context-aware middleware, Rover II, and which exhibits the capabilities required by a proactive context-aware system.

2. It presented various systems and approaches which we have developed, some as part of the enhanced Rover II middleware, in order to realize the vision of proactive context-aware computing. These form the building blocks of a proactive context-aware system and include:

    - Locus [7, 8] - a robust and calibration-free indoor localization, tracking and navigation system for multi-story buildings,

    - SenseMe [9] - a system that leverages a user's smartphone and its multiple sensors in order to perform continuous, on-device, and multi-dimensional context

and activity recognition,

- TellMe [10] - a novel, general and flexible framework for bootstrapped discovery and ranking of heterogeneous relevant services and information in context-aware systems,
- An unsupervised algorithm and system [11], that models users' interests from their Facebook profiles and activities, for personalizing the relevant information,
- A system and an approach [12], that performs multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data, for recommending new information to users,
- A Learning Engine, integrated with the Rover II middleware, which implements several novel approaches and algorithms that employ various contextual features and state of the art machine learning techniques for building diverse behavioral models of users,
- A Planning Framework [14], within the infrastructure of the Rover II middleware, which employs HTN Planning and demonstrates the feasibility of enabling proactivity in a context-aware system.

12.2 Future Work

We have taken firm initial steps, paving a new direction, in the field of context-aware computing. It has been a challenging and audacious endeavor, comprising of multiple research problems that cannot be all addressed in a single PhD dissertation. We are con-

vinced that we have laid a strong foundation and a lot can be built upon it. We have set a good starting point for many more dissertations to come in the future. However, there are still many areas that need to be specifically researched and addressed. We discuss them here:

1. The results presented in Chapter 4 for Locus give us confidence that a calibration-free system can achieve a better accuracy if a more sophisticated indoor propagation model is employed for calculating location. We also believe that this accuracy can be greatly enhanced by taking into account, the building structure, floor plans along with the AP locations and using this information to pinpoint the exact location of the client. Other factors that will come into play as part of this analysis is the number of APs not being heard and the substance through which signals pass though this may make the system less generic. This additional data should be analyzed in such a way that the system still retains its generality and flexibility.

2. The SenseMe system (Chapter 5) recognized five dimensions of a user's situation at any time instant - environmental context, physical activity, location, device activity and social context, to paint a context-rich picture of the user. It also displayed the recognized dimensions to the users via a proof of concept visualization called SenseMeVis. It demonstrated improved or comparable accuracy with respect to existing systems without the need for any calibration or input.

   To further improve the accuracy of SenseMe, more sophisticated methods for recognition and smoothing can be implemented. Also, the system should be enhanced in order to detect other context and activities and with a finer granularity. In addi-

tion, work should be done on displaying periodic context and activity summaries to users.

3. Our results in Chapter 7 demonstrated that the algorithms implemented in the TellMe system showed strong positive correlation with human supplied relevance rankings. Hence, they can be used as an effective means to discover and rank relevant services and information in context-aware systems. We also demonstrated that our approach is general, flexible and can determine relevant information for new users and unanticipated situations without requiring any user history, thus, avoiding cold start.

   This is an initial step and lays the groundwork for several new directions of research. The Semantic Relatedness scores can be improved by mining terms that co-occur in web search queries. Another refinement to the algorithm can be achieved by combining and boosting the results generated by an ensemble of algorithms that performed the best (say Wiki-HSR and STS-HSR). The lch computation can be enhanced by boosting parts of the category graph by combining it with the user's context. In addition, usage history for different users of the TellMe system over a longer period of time should be mined in order to incorporate their individual preferences and personal biases towards services.

4. The algorithm for unsupervised modeling of users' interests from their Facebook profiles, presented in Chapter 8, can be enhanced to further improve results. For instance, topic modeling could be used to generate topic words from the items to augment the list of features. Different mechanisms can be used to analyze the temporal decay of a user's interests. Finally, a hybrid method which combines

this content based approach with collaborative filtering could be used to improve performance. This approach would also address cold start for such a system.

5. The most significant limitation of our approach for multi-dimensional collaborative recommendations using tensor factorization, presented in Chapter 9, is its speed. This can be addressed by creating a Hadoop-based pipeline and parallelizing the algorithm to run on higher-dimensional data. This will also ensure that the approach scales to larger and more complex real-time datasets.

6. In Chapter 10, we explored learning diverse patterns from large-scale data collected from users' smartphones in order to help identify a variety of their behaviors, habits, and daily life places and activities. This, by no means, is exhaustive and several directions of research still need to be explored. Several other models can be developed using this framework. These include predicting the next called contact for users and battery lifetime for their devices, predicting their sleep cycles, classifying their personalities in terms of the Big Five personality model, and inferring their moods based on their smart phone usage. Moreover, it is important to investigate community models which utilize the power of crowd sourcing and group behavior. In addition, work can be done on analyzing behavioral sequences i.e. behaviors that follow each other (for instance, charging phone after reaching home, sending an SMS after missing a call etc.) and modeling them using HMMs.

7. Our planning framework, presented in Chapter 11, addressed a key capability required by a context-aware system to act proactively i.e. act autonomously without an explicit user request. However, in order for it to be really useful, a pipeline from

our context-aware middleware to client agent applications (running on devices such as smartphones or desktops) should be developed. This will enable the plan to be executed on the user's device itself rather than remotely.

Bibliography

[1] A.K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[2] A.K. Dey and G.D. Abowd. Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, volume 4, pages 1–6. Citeseer, 2000.

[3] J. Hong, E.H. Suh, J. Kim, and S.Y. Kim. Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications*, 36(4):7448–7457, 2009.

[4] Gregory D Abowd, Christopher G Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless networks*, 3(5):421–433, 1997.

[5] Daqing Zhang, Zhiwen Yu, and C Chin. Context-aware infrastructure for personalized healthcare. *Studies in health technology and informatics*, 117:154–163, 2005.

[6] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

[7] Preeti Bhargava, Shivsubramani Krishnamoorthy, Aditya Karkada Nakshathri, Matthew Mah, and Ashok Agrawala. Locus: An indoor localization, tracking and navigation system for multi-story buildings using heuristics derived from wi-fi signal strength. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 212–223. Springer, 2013.

[8] Preeti Bhargava, Shivsubramani Krishnamoorthy, Anilesh Shrivastava, Aditya Nakshathri, Matthew Mah, and Ashok Agrawala. Locus: Robust and calibration-free indoor localization, tracking and navigation for multi-story buildings. *Journal of Location Based Services*, 9(03):187–208, 2015.

[9] Preeti Bhargava, Nick Gramsky, and Ashok Agrawala. Senseme: a system for continuous, on-device, and multi-dimensional context and activity recognition. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2014.

[10] Preeti Bhargava, James Lampton, and Ashok Agrawala. Bootstrapped discovery and ranking of relevant services and information in context-aware systems. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2015.

[11] Preeti Bhargava, Oliver Brdiczka, and Michael Roberts. Unsupervised modeling of users' interests from their facebook profiles and activities. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 191–201. ACM, 2015.

[12] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *Proceedings of the 24th International Conference on World Wide Web*, pages 130–140. International World Wide Web Conferences Steering Committee, 2015.

[13] Preeti Bhargava and Ashok Agrawala. Modeling users' behavior from large scale smartphone data collection. In *in submission*.

[14] Preeti Bhargava and Ashok Agrawala. Enabling proactivity in context-aware middleware systems by means of a planning framework based on htn planning. In *Adjunct Proceedings of the 12th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2015.

[15] David S Linthicum. *Enterprise application integration*. Addison-Wesley Professional, 2000.

[16] Preeti Bhargava, Shivsubramani Krishnamoorthy, and Ashok Agrawala. An ontological context model for representing a situation and the design of an intelligent context-aware middleware. In *Proceedings of the ACM Conference on Ubiquitous Computing*, 2012.

[17] Shivsubramani Krishnamoorthy, Preeti Bhargava, Matthew Mah, and Ashok Agrawala. Representing and managing the context of a situation. *The Computer Journal*, 55:1005–1019, 2012.

[18] Shivsubramani Krishnamoorthy and Ashok Agrawala. M-urgency: a next generation, context-aware public safety application. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 647–652, New York, NY, USA, 2011. ACM.

[19] Preeti Bhargava, Nick Gramsky, and Ashok Agrawala. To sense or not to sense: An exploratory study of privacy, trust and other related concerns in personal sensing applications. In *in submission*.

[20] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[21] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000.

[22] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47, 1997.

[23] P. Bahl and V.N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 2000.

[24] M.A. Youssef, A. Agrawala, and A. Udaya Shankar. Wlan location determination via clustering and probability distributions. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003*, pages 143–150.

[25] Jonathan Ledlie, Jun-geun Park, Dorothy Curtis, André Cavalcante, Leonardo Camara, Afonso Costa, and Robson Vieira. Molé: a scalable, user-generated wifi positioning engine. *Journal of Location Based Services*, 6(2):55–80, 2012.

[26] Laura Radaelli and Christian S Jensen. Towards fully organic indoor positioning. In *Proceedings of the Fifth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pages 16–20. ACM, 2013.

[27] P. Bahl, V.N. Padmanabhan, and A. Balachandran. Enhancements to the radar user location and tracking system. Technical report, Microsoft Research, 2000.

[28] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *Pervasive computing*, pages 116–133. Springer, 2005.

[29] W.G. Griswold, R. Boyer, S.W. Brown, T.M. Truong, E. Bhasker, G.R. Jay, and R.B. Shapiro. Activecampus-sustaining educational communities through mobile technology. *University of California, San Diego, Department of Computer Science and Engineering, Technical Report*, 2002.

[30] X. Li. Ratio-based zero-profiling indoor localization. In *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, pages 40–49. IEEE, 2009.

[31] H. Lim, L.C. Kung, J.C. Hou, and H. Luo. Zero-configuration indoor localization over ieee 802.11 wireless infrastructure. *Wireless Networks*, 16(2):405–420, 2010.

[32] D. Madigan, E. Einahrawy, R.P. Martin, W.H. Ju, P. Krishnan, and AS Krishnakumar. Bayesian indoor positioning systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1217–1227. IEEE, 2005.

[33] H. Ye, T. Gu, X. Zhu, J. Xu, X. Tao, J. Lu, and N. Jin. Ftrack: Infrastructure-free floor localization via mobile phone sensing. In *Pervasive Computing and Communications, 2012 Tenth Annual IEEE International Conference on*. IEEE, 2012.

[34] A. Varshavsky, A. LaMarca, J. Hightower, and E. de Lara. The skyloc floor localization system. In *Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on*, pages 125–134. IEEE, 2007.

[35] Alex Varshavsky, Eyal de Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. Gsm indoor localization. *Pervasive and Mobile Computing*, 3(6):698–720, 2007.

[36] Firas Alsehly, Tughrul Arslan, and Zankar Sevak. Indoor positioning with floor determination in multi story buildings. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–7. IEEE, 2011.

[37] Nelson Marques, Filipe Meneses, and Adriano Moreira. Combining similarity functions and majority rules for multi-building, multi-floor, wifi positioning. 2012.

[38] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[39] Gabriella Castelli, Alberto Rosi, Marco Mamei, and Franco Zambonelli. A simple model and infrastructure for context-aware browsing of the world. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications, 2007*, pages 229–238. IEEE.

[40] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, 1994.

[41] Pengfei Zhou, Yuanqing Zheng, Zhenjiang Li, Mo Li, and Guobin Shen. Iodetector: A generic service for indoor outdoor detection. *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems*, 2012.

[42] Lenin Ravindranath, Calvin Newport, Hari Balakrishnan, and Samuel Madden. Improving wireless network performance using sensor hints. In *Proceedings of USENIX conference on Networked systems design and implementation*, 2011.

[43] John Krumm and Ramaswamy Hariharan. Tempio: inside/outside classification with temperature. In *Second International Workshop on Man-Machine Symbiotic Systems*, 2004.

[44] Nmea. http://www.gpsinformation.org/dale/nmea.htm.

[45] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350, 2008.

[46] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, 2010.

[47] Predrag Klasnja, Sunny Consolvo, Tanzeem Choudhury, Richard Beckwith, and Jeffrey Hightower. Exploring privacy concerns about personal sensing. In *Pervasive Computing*, pages 176–183. Springer, 2009.

[48] Mikkel Baun Kjærgaard, Jakob Langdal, Torben Godsk, and Thomas Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 221–234. ACM, 2009.

[49] Ganesh Ananthanarayanan, Maya Haridasan, Iqbal Mohomed, Doug Terry, and Chandramohan A Thekkath. Startrack: a framework for enabling track-based applications. In *Proceedings of the 7th ACM international conference on Mobile systems, applications, and services*, 2009.

[50] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.

[51] Daniel Wagner, Andrew Rice, and Alastair Beresford. Device analyzer: Understanding smartphone usage. In *10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Tokyo, Japan*, pages 212–223. 2013.

[52] Dilution of precision. http://en.wikipedia.org/wiki/Dilution_of_precision_%28GPS%29.

[53] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[54] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 315–330. ACM, 2010.

[55] Fehmi Ben Abdesslem, Andrew Phillips, and Tristan Henderson. Less is more: energy-efficient mobile sensing with senseless. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 61–62. ACM, 2009.

[56] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336, 2008.

[57] Android sensors overview. `http://developer.android.com/guide/topics/sensors/sensors_overview.html`.

[58] Android location strategies. `http://developer.android.com/guide/topics/location/strategies.html`.

[59] Bluetooth baseband. `https://www.bluetooth.org/en-us/specification/assigned-numbers/baseband`.

[60] Jehn-Ruey Jiang, Bing-Rong Lin, and Yu-Chee Tseng. Analysis of bluetooth device discovery and some speedup mechanisms. *Journal of the Institute of Electrical Engineering*, 11(4):301–310, 2004.

[61] Jacopo Staiano, Nuria Oliver, Bruno Lepri, Rodrigo de Oliveira, Michele Caraviello, and Nicu Sebe. Money walks: a human-centric study on the economics of personal mobile data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014.

[62] Nathaniel S Good, Jens Grossklags, Deirdre K Mulligan, and Joseph A Konstan. Noticing notice: a large-scale experiment on the timing of software license agreements. In *Proceedings of the ACM SIGCHI conference on Human factors in computing systems*, 2007.

[63] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2011.

[64] Michael Borenstein, Larry V Hedges, Julian PT Higgins, and Hannah R Rothstein. *Introduction to meta-analysis*. John Wiley & Sons, 2011.

[65] Alex Braunstein, Laura Granka, and Jessica Staddon. Indirect content privacy surveys: measuring privacy without asking about it. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 15. ACM, 2011.

[66] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[67] Jialiu Lin, Shahriyar Amini, Jason I Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 501–510. ACM, 2012.

[68] Aleecia M Mcdonald, Robert W Reeder, Patrick Gage Kelley, and Lorrie Faith Cranor. A comparative study of online privacy policies and formats. In *Privacy enhancing technologies*, pages 37–55. Springer, 2009.

[69] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. Standardizing privacy notices: an online study of the nutrition label approach. In *Proceedings of the ACM SIGCHI Conference on Human factors in Computing Systems*, 2010.

[70] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[71] Rebecca Balebako, Jaeyeon Jung, Wei Lu, Lorrie Faith Cranor, and Carolyn Nguyen. Little brothers watching you: Raising awareness of data leaks on smartphones. In *Proceedings of the Ninth ACM Symposium on Usable Privacy and Security*, 2013.

[72] Sunny Consolvo, Predrag Klasnja, David W McDonald, Daniel Avrahami, Jon Froehlich, Louis LeGrand, Ryan Libby, Keith Mosher, and James A Landay. Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays. In *Proceedings of the 10th ACM international conference on Ubiquitous computing*, 2008.

[73] Dan Cvrcek, Marek Kumpost, Vashek Matyas, and George Danezis. A study on the value of location privacy. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, 2006.

[74] George Danezis, Stephen Lewis, and Ross J Anderson. How much is location privacy worth? In *WEIS*, volume 5. Citeseer, 2005.

[75] Giovanni Iachello and Gregory D Abowd. From privacy methods to a privacy toolbox: Evaluation shows that heuristics are complementary. *ACM Transactions on Computer-Human Interaction*, 15(2):8, 2008.

[76] Sunny Consolvo, Ian E Smith, Tara Matthews, Anthony LaMarca, Jason Tabert, and Pauline Powledge. Location disclosure to social relations: why, when, & what people want to share. In *Proceedings of the ACM SIGCHI conference on Human factors in computing systems*, 2005.

[77] Nan Li and Guanling Chen. Sharing location in online social networks. *Network, IEEE*, 24(5):20–25, 2010.

[78] Daniel Wagner, Mariana Lopez, Andre Doria, Iryna Pavlyshak, Vassilis Kostakos, Ian Oakley, and Tasos Spiliotopoulos. Hide and seek: location sharing practices with social media. In *Proceedings of the 12th ACM international conference on Human computer interaction with mobile devices and services*, 2010.

[79] Scott Lederer, Jennifer Mankoff, and Anind K Dey. Who wants to know what when? privacy preference determinants in ubiquitous computing. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 724–725. ACM, 2003.

[80] Denise Anthony, Tristan Henderson, and David Kotz. Privacy in location-aware computing environments. *IEEE Pervasive Computing*, 6(4):64–72, 2007.

[81] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[82] Ildar Muslukhov, Yazan Boshmaf, Cynthia Kuo, Jonathan Lester, and Konstantin Beznosov. Understanding users' requirements for data protection in smartphones. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 228–235. IEEE, 2012.

[83] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 33–44. ACM, 2012.

[84] Ghita Kouadri Mostefaoui, Jacques Pasquier-Rocha, and Patrick Brezillon. Context-aware computing: a guide for the pervasive computing community. In *IEEE/ACS International Conference on Pervasive Services*, pages 39–48. IEEE, 2004.

[85] Kaijian Xu, Manli Zhu, Daqing Zhang, and Tao Gu. Context-aware content filtering & presentation for pervasive & mobile information systems. In *Proceedings of the 1st ICST international conference on Ambient media and systems*, 2008.

[86] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. Semantic measures for the comparison of units of language, concepts or instances from text and knowledge representation analysis. *arXiv preprint arXiv:1310.1285*, 2013.

[87] Andrei Broder, Marcus Fontoura, Vanja Josifovski, and Lance Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[88] Ramanathan Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003.

[89] David Garlan, Daniel P Siewiorek, Asim Smailagic, and Peter Steenkiste. Project aura: Toward distraction-free pervasive computing. *Pervasive Computing, IEEE*, 1(2):22–31, 2002.

[90] Sheila A McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE intelligent systems*, 16(2):46–53, 2001.

[91] Friederike Klan. Context-aware service discovery, selection and usage. In *Grundlagen von datenbanken*, pages 95–99, 2006.

[92] Vincenzo Suraci, Silvano Mignanti, and Anna Aiuto. Context-aware semantic service discovery. In *16th IST Mobile and Wireless Communications Summit*, pages 1–5. IEEE, 2007.

[93] Luke Steller, Shonali Krishnaswamy, and Jan Newmarch. Discovering relevant services in pervasive environments using semantics and context. *IWUC*, 6:3–12, 2006.

[94] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. Predicting mobile application usage using contextual information. In *Proceedings of the ACM Conference on Ubiquitous Computing*, pages 1059–1065. ACM, 2012.

[95] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the ACM Conference on Ubiquitous Computing*, pages 173–182. ACM, 2012.

[96] D. Salber, A.K. Dey, and G.D. Abowd. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441. ACM, 1999.

[97] Harry Chen, Tim Finin, and Anupam Joshi. An intelligent broker for context-aware systems. In *Adjunct proc. of Ubicomp*, 2003.

[98] A. Ranganathan and R.H. Campbell. A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, pages 143–161. Springer-Verlag New York, Inc., 2003.

[99] Till C Lech and Leendert WM Wienhofen. Ambieagents: a scalable infrastructure for mobile and context-aware information services. In *Proceedings of the fourth ACM international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

[100] Shih-Chun Chou, Wen-Tai Hsieh, Fabien L Gandon, and Norman M Sadeh. Semantic web technologies for context-aware museum tour guide applications. In *Proceedings of the19th IEEE International Conference on Advanced Information Networking and Applications*.

[101] Francesco Bellotti, C Berta, Alessandro De Gloria, and Massimiliano Margarone. User testing a hypermedia tour guide. *IEEE Pervasive Computing*, 1(2):33–41, 2002.

[102] Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of developing and deploying a context-aware tourist guide: the guide project. In *Proceedings of the 2000 ACM international conference on Mobile computing and networking*.

[103] Ronny Kramer, Marko Modsching, Joerg Schulze, and Klaus ten Hagen. Context-aware adaptation in a mobile tour guide. In *Modeling and using context*, pages 210–224. Springer, 2005.

[104] Google calendar event gadgets reference guide. `https://developers.google.com/google-apps/calendar/gadgets/event/`.

[105] Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 25, pages 16–33. Springer-Verlag, Berlin, Heidelberg, 2012.

[106] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

[107] Valentin I Spitkovsky and Angel X Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the Eight International Conference on Language Resources and Evaluation, Istanbul, Turkey*, 2012.

[108] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(1):17–30, 1989.

[109] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, volume 6, pages 1419–1424, 2006.

[110] William Edward Webber. *Measurement in information retrieval evaluation*. University of Melbourne, Department of Computer Science and Software Engineering, 2011.

[111] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[112] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[113] Prem Melville and Vikas Sindhwani. Recommender systems. In *Encyclopedia of machine learning*, pages 829–838. Springer, 2010.

[114] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[115] Matthias Eichstaedt, Qi Lu, and Shang-Hua Teng. Automatic user interest profile generation from structured document access information, May 7 2002. US Patent 6,385,619.

[116] Feng Qiu and Junghoo Cho. Automatic identification of user interest for personalized search. In *Proceedings of the 15th international conference on World Wide Web*, pages 727–736. ACM, 2006.

[117] Ryen W White, Peter Bailey, and Liwei Chen. Predicting user interests from contextual information. In *Proceedings of the 2009 ACM SIGIR conference on Research and development in information retrieval*, pages 363–370.

[118] Hyoung R Kim and Philip K Chan. Learning implicit user interest hierarchy for context in personalization. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 101–108. ACM, 2003.

[119] Krishnan Ramanathan, Julien Giraudi, and Ajay Gupta. Creating hierarchical user profiles using wikipedia. *HP Labs*, 2008.

[120] Jinming Min and Gareth JF Jones. Building user interest profiles from wikipedia clusters. 2011.

[121] Jennifer Golbeck, Cristina Robles, and Karen Turner. Predicting personality with social media. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pages 253–262. ACM, 2011.

[122] Yoram Bachrach, Michal Kosinski, Thore Graepel, Pushmeet Kohli, and David Stillwell. Personality and patterns of facebook usage. In *Proceedings of the 3rd Annual ACM Web Science Conference*, 2012.

[123] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42, 2009.

[124] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[125] Alan Mislove, Bimal Viswanath, Krishna P Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM, 2010.

[126] Qirong Ho, Rong Yan, Rajat Raina, and Eric P Xing. Understanding the interaction between interests, conversations and friendships in facebook. *arXiv preprint arXiv:1211.0028*, 2012.

[127] Fabian Abel, Eelco Herder, and Daniel Krause. Extraction of professional interests from social web profiles. *Proc. UMAP*, 2011.

[128] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on twitter for personalized news recommendations. In *User Modeling, Adaption and Personalization*, pages 1–12. Springer, 2011.

[129] Sandra Garcia Esparza, Michael P O'Mahony, and Barry Smyth. On the real-time web as a source of recommendation knowledge. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 305–308. ACM, 2010.

[130] Sandra Garcia Esparza, Michael P O'Mahony, and Barry Smyth. Catstream: categorising tweets for user profiling and stream filtering. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 25–36. ACM, 2013.

[131] Marco Pennacchiotti and Ana-Maria Popescu. Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[132] Yunfei Ma, Yi Zeng, Xu Ren, and Ning Zhong. User interests modeling based on multi-source personal information fusion and semantic reasoning. In *Active Media Technology*, pages 195–205. Springer, 2011.

[133] Facebooks growth since ipo in 12 big numbers. `http://techcrunch.com/2013/05/17/facebook-growth/`.

[134] How big is facebooks data? `http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500/-terabytes-ingested-every-day/`.

[135] Facebook features. `http://en.wikipedia.org/wiki/Facebook_features`.

[136] Semantic similarity. `http://en.wikipedia.org/wiki/Semantic_similarity`.

[137] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[138] Max Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*, 2010.

[139] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[140] Jeffrey David Ullman, Jure Leskovec, and Anand Rajaraman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[141] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[142] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[143] Yiyang Yang, Zhiguo Gong, and Leong Hou U. Identifying points of interest by self-tuning clustering. In *Proceedings of SIGIR*, pages 883–892. ACM, 2011.

[144] David Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web*, pages 761–770. ACM, 2009.

[145] Fabien Girardin, Josep Blat, Francesco Calabrese, Filippo Dal Fiore, and Carlo Ratti. Digital footprinting: Uncovering tourists with user-generated content. *Pervasive Computing*, 7(4):36–43, Oct.-Nov. 2008.

[146] Daniele Quercia, Rossano Schifanellai, and Luca Maria Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of Hypertext*, pages 116–125. ACM, 2014.

[147] Liangliang Cao, Jiebo Luo, and Thomas S Huang. Annotating photo collections by label propagation according to multiple similarity cues. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 121–130. ACM, 2008.

[148] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of KDD*, pages 1043–1051. ACM, 2013.

[149] Yue Shi, Pavel Serdyukov, Alan Hanjalic, and Martha Larson. Nontrivial landmark recommendation using geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 4(3):17–37, June 2013.

[150] Lin Liao. *Location-based activity recognition*. PhD thesis, University of Washington, 2006.

[151] Victoria Bellotti, Bo Begole, Ed H Chi, Nicolas Ducheneaut, Ji Fang, Ellen Isaacs, Tracy King, Mark W Newman, Kurt Partridge, Bob Price, et al. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1157–1166. ACM, 2008.

[152] Nicolas Ducheneaut, Kurt Partridge, Qingfeng Huang, Bob Price, Mike Roberts, Ed H Chi, Victoria Bellotti, and Bo Begole. Collaborative filtering is not enough? experiments with a mixed-model recommender for leisure activities. In *User Modeling, Adaptation, and Personalization*, pages 295–306. Springer, 2009.

[153] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038. ACM, 2010.

[154] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.

[155] Masoud Sattari, Murat Manguoglu, Ismail H Toroslu, Panagiotis Symeonidis, Pinar Senkul, and Yannis Manolopoulos. Geo-activity recommendations by using improved feature combination. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 996–1003. ACM, 2012.

[156] Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, pages 236–241, 2010.

[157] Richard A Harshman. Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(1), 1970.

[158] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[159] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[160] Evrim Acar, Tamara G. Kolda, and Daniel M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. In *MLG'11: Proceedings of Mining and Learning with Graphs*, August 2011.

[161] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.

[162] Alexandros Nanopoulos. Item recommendation in collaborative tagging systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4):760–771, 2011.

[163] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[164] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[165] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[166] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.

[167] Tamara Gibson Kolda. *Multilinear operators for higher-order decompositions*. United States. Department of Energy, 2006.

[168] Aeron M Buchanan and Andrew W Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 316–322. IEEE, 2005.

[169] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. Scalable tensor factorizations with missing data. In *SDM10: Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 701–712, 2010.

[170] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83. Wollongong, 1998.

[171] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.5. Available online, January 2012.

[172] Jorge J Moré and David J Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

[173] John Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods-support vector learning*, 3, 1999.

[174] Elke Achtert, Hans-Peter Kriegel, and Arthur Zimek. Elki: a software system for evaluation of subspace clustering algorithms. In *Scientific and Statistical Database Management*, pages 580–585. Springer, 2008.

[175] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin*, 2010.

[176] Juha K Laurila, Daniel Gatica-Perez, Imad Aad, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, Markus Miettinen, et al. The mobile data challenge: Big data for mobile computing research. In *Pervasive Computing*, number EPFL-CONF-192489, 2012.

[177] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[178] Denzil Ferreira, Anind K Dey, and Vassilis Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Pervasive computing*, pages 19–33. Springer, 2011.

[179] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[180] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, 12(1):307–328.

[181] Tobias Scheffer. Finding association rules that trade support optimally against confidence. In *Principles of Data Mining and Knowledge Discovery*, pages 424–435. Springer, 2001.

[182] Chi-Min Huang, Josh Jia-Ching Ying, and Vincent S Tseng. Mining users' behaviors and environments for semantic place prediction. In *Nokia Mobile Data Challenge 2012 Workshop. p. Dedicated task*, volume 1, 2012.

[183] Raul Montoliu, Adolfo Martínez-Uso, Jose Martínez-Sotoca, and J McInerney. Semantic place prediction by combining smart binary classifiers. In *Nokia Mobile Data Challenge 2012 Workshop. p. Dedicated task*, volume 1, 2012.

[184] Yin Zhu, Erheng Zhong, Zhongqi Lu, and Qiang Yang. Feature engineering for place category classification. *Mobile Data Challenge 2012 Workshop*.

[185] Elisabeth Lex, Oliver Pimas, Jörg Simon, and Viktoria Pammer-Schindler. Where am i? using mobile sensor data to predict a users semantic place with a random forest algorithm. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 64–75. Springer, 2013.

[186] Xuan Bao, Yilin Shen, Neil Zhenqiang Gong, Hongxia Jin, and Bing Hu. Connect the dots by understanding user status and transitions. In *Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*.

[187] Husain Husna, Santi Phithakkitnukoon, E-A Baatarjav, and Ram Dantu. Quantifying presence using calling patterns. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 184–187. IEEE, 2008.

[188] Huiqi Zhang and Ram Dantu. Quantifying the presence of phone users. In *5th IEEE Consumer Communications and Networking Conference*, 2008.

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

[190] Yifei Jiang, Abhishek Jaiantilal, Xin Pan, Mohammad AAH Al-Mutawa, Shivakant Mishra, and Larry Shi. Personalized energy consumption modeling on smartphones. In *Mobile Computing, Applications, and Services*, pages 343–354. Springer, 2013.

[191] Earl Oliver and Srinivasan Keshav. Data driven smartphone energy level prediction. *University of Waterloo Technical Report*, 2010.

[192] Nishkam Ravi, James Scott, Lu Han, and Liviu Iftode. Context-aware battery management for mobile phones. In *Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, 2008.

[193] Nathan Eagle and Alex Sandy Pentland. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[194] Yaniv Altshuler, Nadav Aharony, Michael Fire, Yuval Elovici, and Alex Pentland. Incremental learning with accuracy prediction of social and individual properties from mobile-phone data. In *2012 International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 2012.

[195] Katayoun Farrahi and Daniel Gatica-Perez. What did you do today?: discovering daily routines from large-scale mobile data. In *Proceedings of the 16th ACM international conference on Multimedia*, 2008.

[196] Vijay Srinivasan, Saeed Moghaddam, Abhishek Mukherji, Kiran K Rachuri, Chenren Xu, and Emmanuel Munguia Tapia. Mobileminer: Mining your frequent patterns on your phone. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*.

[197] David Tennenhouse. Proactive computing. *Communications of the ACM*, 43(5):43–50, 2000.

[198] Roy Want, Trevor Pering, and David Tennenhouse. Comparing autonomic and proactive computing. *IBM Systems Journal*, 42(1):129–135, 2003.

[199] Antti Salovaara and Antti Oulasvirta. Six modes of proactive resource management: a user-centric typology for proactive behaviors. In *Proceedings of the third Nordic conference on Human-computer interaction*, pages 57–60. ACM, 2004.

[200] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[201] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning: theory & practice*. Elsevier, 2004.

[202] Earl D Sacerdoti. The nonlinear nature of plans. Technical report, DTIC Document, 1975.

[203] David E Wilkins. Can ai planners solve practical problems? *Computational intelligence*, 6(4):232–246, 1990.

[204] Kutluhan Erol. Hierarchical task network planning: formalization, analysis, and implementation. 1996.

[205] Dana Nau, Yue Cao, Amnon Lotem, and Hector Muñoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 968–973. Morgan Kaufmann Publishers Inc., 1999.

[206] Julien Bidot, Christos Goumopoulos, and Ioannis Calemis. Using ai planning and late binding for managing service workflows in intelligent environments. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 156–163. IEEE, 2011.

[207] Maja Vukovic and Peter Robinson. Adaptive, planning based, web service composition for context awareness. In *In Proceedings of the Second International Conference on Pervasive Computing*, 2004.

[208] Maja Vukovic and Peter Robinson. Shop2 and tlplan for proactive service composition. In *UK-Russia Workshop on Proactive Computing*. Citeseer, 2005.

[209] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004.

[210] Abir Qasem, Jeff Heflin, and Héctor Muñoz-Avila. Efficient source discovery and service composition for ubiquitous computing environments. In *Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications (ISWC)*, 2004.

[211] Anand Ranganathan and Roy H Campbell. Autonomic pervasive computing based on planning. In *Autonomic Computing, 2004. Proceedings. International Conference on*, pages 80–87. IEEE, 2004.

[212] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *Knowledge and Data Engineering, IEEE Transactions on*, 21(1):6–20, 2009.

[213] Barbara E Ainsworth, William L Haskell, Stephen D Herrmann, Nathanael Meckes, David R Bassett, Catrine Tudor-Locke, Jennifer L Greer, Jesse Vezina, Melicia C Whitt-Glover, and Arthur S Leon. 2011 compendium of physical activities: a second update of codes and met values. *Medicine and science in sports and exercise*, 43(8):1575–1581, 2011.

[214] Christian B. Almazan. *ROVER: Architectural Support for Exposing and Using Context*. PhD thesis, University of Maryland, College Park, 2010.

[215] Omar Alonso, Daniel E Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15, 2008.

[216] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[217] Jie Bao, Yu Zheng, and Mohamed F Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 199–208. ACM, 2012.

[218] Michael Beigl, Albert Krohn, Tobias Zimmer, Christian Decker, and Philip Robinson. Awarecon: Situation aware context communication. In Anind Dey, Albrecht Schmidt, and Joseph McCarthy, editors, *UbiComp 2003: Ubiquitous Computing*, volume 2864 of *Lecture Notes in Computer Science*, pages 132–139. Springer Berlin / Heidelberg, 2003.

[219] Abraham Bernstein and Mark Klein. Towards high-precision service retrieval. In *The Semantic WebISWC 2002*, pages 84–101. Springer, 2002.

[220] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, 2010.

[221] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. Context-addict. Technical report, Dip. Elettronica e Informazione, Politecnico di Milano, 2006.

[222] C. Bolchini, C.A. Curino, E. Quintarelli, F.A. Schreiber, and L. Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, 2007.

[223] Peter Brusilovsky and Eva Millán. User models for adaptive hypermedia and adaptive educational systems. In *The adaptive web*, pages 3–53. Springer-Verlag, 2007.

[224] H.E. Byun and K. Cheverst. Harnessing context to support proactive behaviours. In *ECAI Workshop on AI in Mobile Systems, Lyon*. Citeseer, 2002.

[225] A. Chen. Context-aware collaborative filtering system: Predicting the users preference in the ubiquitous computing environment. *Location-and Context-Awareness*, pages 75–81, 2005.

[226] H. Chen. An intelligent broker architecture for context-aware systems. *PhD proposal in Computer Science, University of Maryland, Baltimore, USA*, 2003.

[227] Harry Chen, Tim Finin, and Anupam Joshi. The soupa ontology for pervasive computing. In Valentina Tamma, Stephen Cranefield, Timothy Finin, Steven Willmott, Marius Walliser, Stefan Brantschen, Monique Calisti, and Thomas Hempfling, editors, *Ontologies for Agents: Theory and Experiences*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 233–258. 2005.

[228] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(03):197–207, 2003.

[229] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 258–267. Ieee, 2004.

[230] Anne E. Cook, John E. Limber, and Edward J. O'Brien. Situation-based context and the availability of predictive inferences. *Journal of Memory and Language*, 44(2):220 – 234, 2001.

[231] J. Coutaz, J.L. Crowley, S. Dobson, and D. Garlan. Context is key. *Communications of the ACM*, 48(3):49–53, 2005.

[232] Ekahau. http://www.ekahau.com/.

[233] P. Fahy and S. Clarke. Cass–a middleware for mobile context-aware applications. In *Workshop on Context Awareness, MobiSys*. Citeseer, 2004.

[234] Gerhard Fischer. Context-aware systems: the'right'information, at the'right'time, in the'right'place, in the'right'way, to the'right'person. In *Proceedings of the ACM International Working Conference on Advanced Visual Interfaces*, 2012.

[235] FScore. `http://en.wikipedia.org/wiki/F1_score`.

[236] Sandra Garcia Esparza, Michael P O'Mahony, and Barry Smyth. Catstream: categorising tweets for user profiling and stream filtering. In *Proceedings of the ACM International conference on Intelligent user interfaces*, 2013.

[237] Simon Gerber, Michael Fry, Judy Kay, Bob Kummerfeld, Glen Pink, and Rainer Wasinger. *PersonisJ: mobile, client-side user modelling*. Springer, 2010.

[238] H. Haidarian, W. Dinalankara, S. Fults, S. Wilson, D. Perlis, M. Schmill, T. Oates, DP Josyula, and ML Anderson. The metacognitive loop: An architecture for building robust intelligent systems. In *PAAAI Fall Symposium on Commonsense Knowledge (AAAI/CSK10)*, 2010.

[239] K. Henricksen and J. Indulska. Personalising context-aware applications. In *On the Move to Meaningful Internet Systems Workshops*, pages 122–131. Springer, 2005.

[240] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. *Pervasive Computing*, pages 79–117, 2002.

[241] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9*, HICSS '03, pages 292.1–, Washington, DC, USA, 2003. IEEE Computer Society.

[242] M. Kaenampornpan and E. ONeill. An integrated context model: Bringing activity to context. In *Proc. Workshop on Advanced Context Modelling, Reasoning and Management*. Citeseer, 2004.

[243] SeokHoon Kang, Hangil Won, Gwanggil Jeon, and Young-Sup Lee. Call prediction model based on smartphone users behavior. In *Convergence and Hybrid Information Technology*, pages 180–185. Springer, 2012.

[244] Juuso Karikoski and Tapio Soikkeli. Contextual usage patterns in smartphone communication services. *Personal and ubiquitous computing*, 17(3):491–502, 2013.

[245] Anders Kofod-Petersen and Agnar Aamodt. Case-based situation assessment in a mobile context-aware system. In *Proceedings of Artificial Intelligence in Mobile Systems 2003 AIMS (2003)*, pages 41–49. University des Saarlandes, Saarbrucken, Germany, 2003.

[246] P. Korpipää and J. Mäntyjärvi. An ontology for mobile device sensor-based context awareness. *Modeling and Using Context*, pages 451–458, 2003.

[247] Shivsubramani Krishnamoorthy and Ashok Agrawala. A context-aware framework for mobile applications. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 403–404, New York, NY, USA, 2011. ACM.

[248] Shivsubramani Krishnamoorthy and Ashok Agrawala. Contextual information integration platform for humanitarian relief. In *Proceedings of the International Conference on Wireless Technologies for Humanitarian Relief (ACWR 2011), Amritapuri, Kerala, India*. ACM, 2011.

[249] O. Kwon. The potential roles of context-aware computing technology in optimization-based intelligent decision-making. *Expert Systems with Applications*, 31(3):629–642, 2006.

[250] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[251] T. Ma, Y.D. Kim, Q. Ma, M. Tang, and W. Zhou. Context-aware implementation based on cbr for smart home. In *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, volume 4, pages 112–115. IEEE, 2005.

[252] Ulrich Meissen, Stefan Pfennigschmidt, Agns Voisard, and Tjark Wahnfried. Context- and situation-awareness in information logistics. In Wolfgang Lindner, Marco Mesiti, Can Trker, Yannis Tzitzikas, and Athena Vakali, editors, *Current Trends in Database Technology - EDBT 2004 Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 448–451. Springer Berlin / Heidelberg, 2005.

[253] Elke Michlmayr, Steve Cayzer, and Paul Shabajee. Add-a-tag: Learning adaptive user profiles from bookmark collections. In *ICWSM*, 2007.

[254] M-Urgency. http://m-urgency.umd.edu/.

[255] V Naresh, Prasad Pingali, Vasudeva Varma, M Krishna, and P Venkata. Location based web search on gsm/gprs mobile phones. In *Proceedings of WWW*, 2006.

[256] Dana S Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J William Murdock, Dan Wu, and Fusun Yaman. Shop2: An htn planning system. *J. Artif. Intell. Res.(JAIR)*, 20:379–404, 2003.

[257] OpenCyc. http://opencyc.org/.

[258] Kamran Taj Pathan and Stephan Reiff-Marganiec. Towards activity context using software sensors. *arXiv preprint arXiv:0906.3925*, 2009.

[259] Santi Phithakkitnukoon, Husain Husna, and Ram Dantu. Behavioral entropy of a cellular phone user. In *Social Computing, Behavioral Modeling, and Prediction*, pages 160–167. Springer, 2008.

[260] Paul Prekop and Mark Burnett. Activities, context and ubiquitous computing. *Computer Communications*, 26(11):1168–1176, 2003.

[261] Protege-OWL. http://protege.stanford.edu/overview/protege-owl.html.

[262] Anne Rubruck, Azad Aminian, Jyothi Yalpi, Paul Hanzal, Sascha Winde, Sathya Kuppusami, Sohaib Younis, and Stefan Thomas. Domestic robot for collecting used cups in a known environment.

[263] A.H Sayed, A Tarighat, and N Khajehnouri. Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information. *IEEE Signal Processing Magazine*, 22:24 40, July 2005.

[264] A. Schmidt, M. Beigl, and H.W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[265] Robert Scoble and Shel Israel. *Age of Context: Mobile, Sensors, Data and the Future of Privacy*. CreateSpace Independent Publishing Platform, 2013.

[266] H. Si, Y. Kawahara, H. Morikawa, and T. Aoyama. A stochastic approach for creating context-aware services based on context histories in smart home. *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP*, 577:37, 2005.

[267] John Soldatos, Ippokratis Pandis, Kostas Stamatis, Lazaros Polymenakos, and James L Crowley. Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services. *Computer Communications*, 30(3):577–591, 2007.

[268] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Workshop on advanced context modelling, reasoning and management as part of UbiComp*, pages 1–8. Citeseer, 2004.

[269] Martin Szomszor, Harith Alani, Ivan Cantador, Kieron OHara, and Nigel Shadbolt. Semantic modelling of user interests based on cross-folksonomy analysis. In *The Semantic Web-ISWC 2008*, pages 632–648. Springer, 2008.

[270] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[271] Steven Van Canneyt, Steven Schockaert, Olivier Van Laere, and Bart Dhoedt. Time-dependent recommendation of tourist attractions using flickr. In *23rd Benelux conference on Artificial Intelligence (BNAIC 2011)*, 2011.

[272] Daniel T Wagner, Andrew Rice, and Alastair R Beresford. Device analyzer: Large-scale mobile data collection. In *Workshop on Big Data Analytics*, 2013.

[273] Kai Wang, Richong Zhang, Xudong Liu, Xiaohui Guo, Hailong Sun, and Jinpeng Huai. Time-aware travel attraction recommendation. In *Web Information Systems Engineering–WISE 2013*, pages 175–188. Springer, 2013.

[274] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. Ontology based context modeling and reasoning using owl. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 18–22. IEEE, 2004.

[275] Rainer Wasinger, Michael Fry, Judy Kay, and Bob Kummerfeld. User modelling ecosystems: a user-centred approach. In *User Modeling, Adaptation, and Personalization*. Springer, 2012.

[276] Molly McLure Wasko and Samer Faraj. Why should i share? examining social capital and knowledge contribution in electronic networks of practice. *MIS quarterly*, pages 35–57, 2005.

[277] Jason Watson, Andrew Besmer, and Heather Richter Lipford. + your circles: sharing behavior on google+. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 12. ACM, 2012.

[278] Zhiqiang Wei, Nana Wang, Mijun Kang, and Wei Zhou. An agent-based context-aware middleware for pervasive computing. In *International Symposium on Information Science and Engineering*, volume 2, pages 116–119. IEEE, 2008.

[279] Weka Machine Learning Project. `http://www.cs.waikato.ac.nz/ml/weka/`.

[280] S.J.H. Yang, APM Huang, R. Chen, S.S. Tseng, and Y.S. Shen. Context model and context acquisition for ubiquitous content access in ulearning environments. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 2, pages 78–83. IEEE, 2006.

[281] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 22(4):315–347, 2007.

[282] J.M. Zagami, S.A. Parl, J.J. Bussgang, and K.D. Melillo. Providing universal location services using a wireless e911 location network. *Communications Magazine, IEEE*, 36(4):66 –71, apr 1998.

[283] Xia Zhao, Yao Guo, Qing Feng, and Xiangqun Chen. A system context-aware approach for battery lifetime prediction in smart phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*.

[284] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, 184:17–37, 2012.

[285] Alchemy api. `http://www.alchemyapi.com/`.

[286] Android content providers. `http://developer.android.com/guide/topics/providers/content-providers.html`.

[287] Android wi-fi scanning. `http://developer.android.com/reference/android/net/wifi/WifiManager.html`.

[288] Angie's list. `http://www.angieslist.com/`.

[289] Angie's list category list. `http://www.angieslist.com/services.htm`.

[290] Borda count. `http://en.wikipedia.org/wiki/Borda_count`.

[291] Facebook graph api. `https://developers.facebook.com/docs/reference/api/`.

[292] Facebook page categories. `https://www.facebook.com/pages/create/`.

[293] Facebook statistics. `http://blog.kissmetrics.com/facebook-statistics/`.

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

[295] Foursquare categories. `http://aboutfoursquare.com/foursquare-categories/`.

[296] Foursquare venues api. https://developer.foursquare.com/overview/venues.html.

[297] The friedman test for 3 or more correlated samples. http://vassarstats.net/textbook/ch15a.html.

[298] Google feed api. https://developers.google.com/feed/.

[299] Google maps. https://maps.google.com/.

[300] Google news. https://news.google.com/.

[301] Google news category list. https://support.google.com/webmasters/answer/42993?hl=en.

[302] Google now cards. https://support.google.com/websearch/answer/2839499?hl=en.

[303] Google place types. https://developers.google.com/places/documentation/supported_types.

[304] The google places api. https://developers.google.com/places/.

[305] Groupon. www.groupon.com/.

[306] ibeacon experiments. http://blog.nerdery.com/2013/11/nerdery-labs-ibeacon-experiments/.

[307] Last.fm. http://www.last.fm/.

[308] Livingsocial. https://www.livingsocial.com/.

[309] Mapquest traffic web service. http://www.mapquestapi.com/traffic/.

[310] Meetup. http://www.meetup.com/.

[311] Amazon mechanical turk. https://www.mturk.com/mturk/.

[312] Natural language tool kit. http://nltk.org/.

[313] Nmea sentences. http://aprs.gids.nl/nmea/.

[314] Open directory project. http://www.dmoz.org/.

[315] Opencalais. http://www.opencalais.com/.

[316] Opencalais document categorization. http://www.opencalais.com/documentation/calais-web-service-api/api-metadata/document-categorization.

[317] Opencalais entity/fact/event definitions and descriptions. http://www.opencalais.com/documentation/calais-web-service-api/api-metadata/entity-index-and-definitions.

[318] Opencalais semantic metadata. http://www.opencalais.com/documentation/calais-web-service-api/api-metadata.

[319] Opencalais social tags. http://www.opencalais.com/documentation/opencalais-web-service-api/api-metadata-english/api-metadata-english-social-tags.

[320] Owl 2 web ontology language document overview http://www.w3.org/tr/owl2-overview/.

[321] Pearson's correlation coefficient. http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.

[322] Quantified self. http://quantifiedself.com/.

[323] Random forest. *http://www.stat.berkeley.edu/~breiman/RandomForests/*.

[324] *World Wide Web Consortium, "RDF Primer," REC-rdf-primer-20040210, 2004.*

[325] Residential delivery indicator. http://www.cdyne.com/terms/RDI.aspx.

[326] Model evaluation - regression. http://www.saedsayad.com/model_evaluation_r.htm.

[327] Restfb. http://restfb.com/.

[328] Android reverse geocoding. http://developer.android.com/training/basics/location/geocoding.html.

[329] Smartystreets liveaddress api. http://smartystreets.com/products/liveaddress-api.

[330] Spearman rank correlation coefficient. http://mathworld.wolfram.com/SpearmanRankCorrelationCoefficient.html.

[331] Umbc semantic similarity service. http://swoogle.umbc.edu/SimService/index.html.

[332] Tempo ai - tempo smart calendar. http://tempo.ai/.

[333] Twitter4j. http://twitter4j.org/en/.

[334] Weather underground api. http://api.wunderground.com/.

[335] Wifislam. https://angel.co/wifislam.

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

[337] Yelp categories. http://www.yelp.com/developers/documentation/category_list.

[338] Yelp api. http://www.yelp.com/developers/documentation/v2/search_api.