Title
Small Data Systems

Permalink
https://escholarship.org/uc/item/2rd9m1r2

Author
Hsieh, Cheng-kang

Publication Date
2016

Peer reviewed|Thesis/dissertation
Small Data Systems

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Cheng-Kang Hsieh

2016
A myriad of digital services mediate our daily lives. As a result, we are continuously generating digital traces, referred to as small data, that can be used to chronicle, characterize, and influence our behaviors and preferences. This thesis is focused on the notion of small data systems — the services and tools designed for individuals to more directly and personally leverage their collective small data. We develop novel algorithms, toolsets and the system infrastructure to address cross-cutting challenges in small data systems. We evaluate the efficacy and feasibility of our contributions with real-world datasets, system deployments, and user studies in two application contexts: (1) Lifestreams, a toolset to facilitate the exploration of small data for personal behavioral analysis and chronic disease prevention, and (2) Immersive Recommendations, a new recommendation paradigm using small data to enable effective personalization for online services across the web.
The dissertation of Cheng-Kang Hsieh is approved.

Deborah Lynn Estrin

Warren S. Comulada

Tyson Condie

Junghoo Cho

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2016
To my parents
1 Introduction ......................................................... 1
   1.1 Small Data .................................................. 1
   1.2 Small Data Systems ......................................... 2
   1.3 Our Contributions .......................................... 3
   1.4 Overview .................................................. 5

2 Small Data for Personal Behavioral Analysis ...................... 8
   2.1 Introduction ................................................ 8
   2.2 Personal Behavioral Analysis with Smartphone Data .......... 8
   2.3 Our Contributions .......................................... 9
   2.4 Related Work ............................................. 10
       2.4.1 Smartphone Data for Chronic Diseases Management ... 10
       2.4.2 Smartphone Data for Behavioral Studies ............... 12
       2.4.3 Smartphone Data for Self-Management ................ 15
   2.5 Conclusions .............................................. 16

3 Lifestreams: A Toolset for Small Data Behavioral Analysis ..... 18
   3.1 Introduction .............................................. 18
   3.2 Challenges for Small Data Behavioral Analysis .............. 18
   3.3 Lifestreams Architecture and Design Process ............... 20
       3.3.1 Architecture ......................................... 20
       3.3.2 Interactive Design Process ......................... 22
   3.4 Feature Extraction and Aggregation ......................... 23
5 Small Data System Architecture ........................................ 55
  5.1 Introduction .......................................................... 55
  5.2 System Challenges for Small Data ................................ 55
  5.3 Our Contributions .................................................... 57
  5.4 Proposed Architecture ............................................. 58
  5.5 DPU Containers: Lifestreams DB ................................. 60
    5.5.1 Data Modeling .................................................. 63
    5.5.2 Chunk-Based Data Management ............................. 66
    5.5.3 Two-Level GDS Chunk Replacement Policy ............... 70
    5.5.4 Chunk-Assisted RDF Query Evaluation .................... 71
  5.6 Performance Evaluation ........................................... 72
    5.6.1 Dataset ......................................................... 72
    5.6.2 Chunking Overhead ............................................ 73
    5.6.3 Query Performance ........................................... 74
    5.6.4 Performance with Simulated Workload ..................... 76
  5.7 Conclusions .......................................................... 78

6 Small Data for Personalizing Online Experiences ................. 81
  6.1 Introduction .......................................................... 81
  6.2 Immersive Recommendations ..................................... 81
  6.3 Challenges for Immersive Recommendations ................... 83
  6.4 Our Solutions and Contributions ................................ 84
  6.5 Related Work .......................................................... 86
    6.5.1 Recommendation Systems .................................. 86
    6.5.2 Collaborative Filtering ...................................... 88
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Lifestreams Data Analysis Stack</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison of execution time between the traditional DBSCAN and our Two-Phase approaches for different amount of samples (sample rate=1/60Hz). Each point is an average of 10 runs. The error bars are the corresponding 95% confidence interval.</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>A walkthrough of the change detection algorithm</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Lifestreams visualization allows users to interactively select different feature groups, switch between different visualization views, adjust parameters, and export the visualization results. Users can use Lifestreams visualization to quickly and flexibly navigate through vast and high-dimensional personal data streams and create visual aids to guide the discussion with patients or participants. (Note that the geo-information in the Place View has been obfuscated to protect the participant’s privacy.)</td>
<td>36</td>
</tr>
<tr>
<td>4.1</td>
<td>Individual 2D correlation matrix of a participant. This figure shows correlations between the participants working schedule, stress levels, exercise routines, and daily activity patterns.</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>A 3D correlation matrix of all 44 participants. This figure shows an overview of significant patterns among the participants. The number on the left (and right) of each cell indicates the number of participants with negative (and positive) correlation coefficients higher than 0.3.</td>
<td>42</td>
</tr>
</tbody>
</table>
4.3 Sample single feature change detection results for a participant (Startup=30 days, ARL0=500). This figure shows changes detected across daily stress levels, diet, exercise routines, and driving distances.

4.4 Tradeoff between the responsiveness of change detection and the time the detected changes are expected to last. With a shorter startup time or a smaller $ARL_0$, the algorithm detects changes earlier, but also tends to include temporary changes.

4.5 A sample correlation change detection result for a participant. This figure shows how the correlation between two features may vary over time.

4.6 A sample correlation change detection result for the other participant.

5.1 Small Data Architecture: the diagram represents the flow of data between data storage units (DSUs), processing units (DPUs), and visualizations/apps (DVUs). Lifestreams DB is a DPU container that manages common concerns among DPUs such as caching, security, and providing a unified query interface. Its outputs can be directly consumed by DVUs, or by other DPUs that provide additional data processing capability.

5.2 Lifestreams DB Architecture: Lifestreams Pipeline consists of a set of DPU modules that acquire and process data from various small data DSUs. Lifestreams Triplestore manages the generated data and make them available to DVUs through a unified SPARQL endpoint.
5.3 Query Performance of Different Scenarios: our approach outperforms the Jena TDB by up to 14x when the chunks are readily available. Decompressing is much faster than reproducing a chunk, while decryption adds only negligible overhead. Gmail data requires more time to reproduce since more HTTP requests need to be made. The varying performance in different scenarios evidence the need of an cost- and size-aware chunk replacement policy.

5.4 Real-World Query Performance: the performance gain of our chunk-skipping technique becomes more evident (up to 14x) for a complex real-world query where more scans need to be made over the search space.

5.5 Query Performance with Simulated Workload: our Two-Level GDS approach shows superior performance over LRU, and outperforms Jena TDB that retains all 50.44MB of data, by up to 4.7x using only about 1/10 of storage.

6.1 An overview of immersive recommendation. In the profiling phase, Context-Aware-LDA (CA-LDA) is proposed to systematically profile users’ digital traces from different contexts and create user profiles. In the recommendation phase, a hybrid collaborative filtering algorithm is proposed to fuse the user profiles, the item profiles, and the existing ratings to predict the ratings that are still unknown.

7.1 Context-Aware LDA’s co-training structure: All the corpora shares the same set of salient topics that reflect the main topics of the documents while each of the corpora has its own unique background topic that is associated with the noise specific to that context.
7.2 Precision-recall curves for different profiling algorithms. CA-LDA outperformed the prior algorithms by 18.7% and 77.4% in mean average precision for Medium.com and Meetup.com respectively.

8.1 Average Recall@50 and Mean Reciprocal Rank when a user had made 0 to 5 or 10 feedback signals. Immersive recommendation (ImmRec) significantly outperformed the second best approach in every case by up to 57.9% in Recall@50 and 42.6% in MRR, and was able to smoothly improve the performance when more feedback was available (post cold start).

8.2 Expected improvement of ImmRec in utility over other algorithms in the user experiment (N=33). ImmRec outperformed other algorithms by up to 20.7%, but had less significant improvement during the cold-start phase for news and the steady-state phase for meetups. (Asterisks represent the significance level of the paired t-test. *: p < 0.05, **: p < 0.01, ***: p < 0.001)
### List of Tables

5.1 Data Modeling: the ontology defined in `schema.org` is used to allow data interoperability. Some extensions are made to fill the gap in the existing ontology (see the asterisked entries). Of note is that the purchase record is derived from email receipts on an opt-in basis. The phone-based data are uploaded to ohmage [THL15], a participatory sensing platform, that serves as a DSU in the small data ecosystem. .................................................. 63

5.2 Gmail and Moves data show varying characteristics in sizes and reproduction time. These differences must be taken into account to achieve efficient resource utilization. .......................... 73

5.3 Chunking incurs negligible overhead in storage, but with sizable overhead in write throughput due to the time spent in extracting chunk features. .......................................................... 74

7.1 Frequent background terms learned by Context-Aware LDA. The words were stemmed before included in the model. .................... 102

7.2 The predictive power of different types of tweets using CA-LDA. .... 108
Acknowledgments

I would like to express my sincere gratitude to my advisor, Deborah Estrin, for her guidance in my journey through graduate school. I learned from Deborah how to choose research topics that may make impact, and to present the ideas and work clearly. I also learned from her how to always be open-minded, patient, and supportive to people around me. She inspired my passion for good works and collaboration. I deeply appreciated the support she gave me, and the endless encouragement she provided during my graduate study.

Second, I would like to thank Nithya Ramanathan, Hongsuda Tangmunarunkit, and Hossein Flaki, with whom I worked closely on the research problems that center this thesis. Working with them on a day-to-day basis, especially in the earlier years of my graduate study, was a very valuable experience and made me grow as a capable researcher.

Furthermore, I would like to thank my committee members Mani Srivastava, Junghoo Cho, Scott Comulada, and Tyson Condie for their encouragement and invaluable comments and feedback. I would also like to thank Dallas Swendeman, Mor Naaman, JP Pollak, and Arnaud Sahuguet for their generous help and comments on the central research problems considered in this thesis. Furthermore, I would like to thank Tei-Wei Kuo, Pi-Cheng Hsiu, and Yuan-Hao Chang for advising me during my undergraduate years, and Jeremy Elson for mentoring me during my summer internship in Microsoft Research.

I would also like to thank the members of the Small Data Lab: Jinha Kang, Brent Longstaff, Faisal Alquaddoomi, Dony George, Longqi Yang, Fabian Okeke, Lucky Gunasekara, Michael Carroll, Ivy Wu, Judy Wu, Narumi Toida, Steve Nolen, Cameron Ketcham, Joshua Selsky, John Jenkins, Hongyi Wen, Honghao Wei, and Yordanos Goshu; and my friends in Los Angeles and New York: Yuh-Jie Chen, Mohan Yang, Zhiyang Wang, Tsung-Yi Lin, Cing-Yu Chu, and Ke Xie.
I have been very fortunate to know them, to work with them, to hangout with them, and to learn from them. Without their support, this thesis would not have been possible.

Finally, I would like to thank my family for their unconditional love and support. This thesis is dedicated to them.
VITA

2009 B.S. (Computer Science), National Taiwan University.


2012 Teaching Assistant (CS 130 Software Engineering), UCLA.

2013 Summer Research Intern, Microsoft Research, Redmond.

2011–present Research Assistant, Computer Science Department, UCLA.

2014–present Visiting PhD Student, Cornell Tech, NYC.

PUBLICATIONS


Min S. Hane Aung, Faisal Alquaddoomi, Cheng-Kang Hsieh, Mashfiqui Rabbi,
CHAPTER 1

Introduction

In this chapter, we introduce the notion of small data and small data systems, and provide an outline of the remainder of the thesis.

1.1 Small Data

With the rise of the web, social media, e-commerce, and mobile communications, digital applications and tools that capture and analyze consumer behaviors are proliferating at an explosive rate. Big data, referring to the potential of extracting insight and value from behaviors of a large number of people, has been the main theme of this trend. In this thesis, however, we consider a complementary concept — small data.

Small data are the data individuals generate as a by-product in their daily life, such as sending e-mail, texting, or tweeting, buying groceries or take-out, going to work on foot or by car, watching TV shows or movies at home, playing games on mobile devices or game consoles, or talking to their Amazon Echo. These traces reflect who we are, what we do and with whom, and what we are interested in, and are positioned for an explosion in richness and variety [Est14].
1.2 Small Data Systems

Small Data Systems refer to the services and tools designed for individuals to more directly and personally leverage their collective small data. The premise of small data systems is that by reassembling diverse, cross-channel information, we can create a more coherent and actionable view of ourselves to improve well-being, productivity, and the services we receive across the web. This is different from the traditional provider-centric services where the services are only concerned with and have access to a partial view of our behaviors that take place within their platform.

In this thesis, we develop novel algorithms, toolsets and the system infrastructure to address cross-cutting challenges in small data systems, and carefully evaluate their efficacy and feasibility. To drive this research, we consider the following two small data application contexts:

- **Using small data for personal behavioral analysis.** Small data enables a unprecedentedly detailed view to our daily behaviors. These data can be used to systematically monitor chronic conditions or health behavior outside the clinical setting and have great potential for applications in behavioral research, intervention, and self-management. We develop Lifestreams, a personal behavioral analysis toolset that analyzes diverse small data streams about users’ diet, stress and exercise, which are three major factors to many chronic diseases. Lifestreams is able to identify key behavioral patterns and trends in these behaviors that behavioral researchers are interested in and are potentially not otherwise identified by individuals themselves.

- **Using small data for personalizing online experiences.** Our collective small data captures our diverse interests, preferences, and aspirations. These data can be used to enable a new kind of recommendation experiences, called Immersive Recommendations, where users use their small data
to personalize their experiences across the web and more effectively retrieve the contents, entertainments, or social connections that suit their interests. We explore the feasibility of immersive recommendations for news and local meetup event recommendations, and develop a real-world news recommendation system, called Newsfie.org, that uses diverse small data, including users’ Twitter streams, email, Youtube watch history, etc., to make immediate personalized news recommendations.

1.3 Our Contributions

The above application contexts highlight many challenges shared by a broad range of small data systems. In particular, we make technical contributions to the following areas:

Data Transformation and Information Extraction

Most small data streams are collected in their raw form. These raw small data are highly variable and not particularly relevant to the target application contexts. For example, consider the readings from sensors on-board the smartphones or textual data from users’ social or personal communications, such as tweets and email. Specific data transformation and information extraction techniques are often required to make the raw small data useful. Moreover, since small data are generated as byproducts during users daily activities, they are inherently riddled with noise and contextual biases introduced by the context in which they were generated. Consequently, how to denoise the small data and extract information truly relevant to the user and the specific task at hand are major challenges we address in this thesis.

To this end, we develop various data transformation techniques in Lifestreams to extract important behavioral indicators from raw smartphone data (Chapter
In the context of immersive recommendations, we develop **Context-Aware LDA**, a novel textual data profiling algorithm to extract users’ interests from a variety of small data channels while suppressing the noise and contextual biases introduced by different contexts (Chapter 7).

**User Modeling with Diverse Small Data**

The power of small data comes from combining signals from multiple data streams. How to aggregate and model with such continuous data streams and enable specific applications, such as detecting behavioral patterns or making personalized recommendations, is another thread of major technical challenges we address in this thesis.

We propose a series of automatic inference algorithms in Lifestreams to systematically identify key behavioral patterns, trends, and potential behavior change points from a large number of concurrent small data streams, and evaluate their efficacy with the data collected in the real-world studies (Chapter 3 and Chapter 4). For immersive recommendations, we propose a hybrid collaborative filtering algorithm, called **collaborative user-item regression**, that incorporates multi-channel small data to make significantly more accurate recommendations than the state-of-the-art approaches that do not use small data (Chapter 8).

**Data Interoperability, Management, and Security**

A small data system often needs to simultaneously interface with a large number of third-party services and process a variety of data streams with different data schemas dictated by various service providers. Developing and maintaining such systems can be rather costly and the system itself tends to be fragile given a large number of external dependencies and varying data schemas.

We strive to create a **small data ecosystem** in which small data apps can
be readily developed and deployed. We propose a modular small data system architecture to achieve this goal. The central component of this architecture is Lifestreams DB, a RDF-based database system that allows diverse small data streams to be merged against each other or against external data sources while developers only need to write simple and maintainable database queries. Lifestreams DB adopts a soft-state design and a chunk-based data management scheme to improve some aspect of security, storage efficiency, and query performance (Chapter 5).

The majority part of this thesis will be devoted to the discussion about the technical challenges in the above three areas, the related work, our proposed solutions, and the results of detailed evaluations conducted with the real-world datasets, live experiments, or real-world system deployments. Next, we provide an overview of the remaining chapters of this thesis.

1.4 Overview

Chapter 2: Small Data for Personal Behavioral Analysis

In this chapter, we discuss the use of small data for behavioral analysis with a particular focus on smartphone data. We discuss the potential of using smartphone data for chronic disease management, to enable free-living behavioral studies, and to improve self-management and personalized interventions.

Chapter 3: Lifestreams: A Toolset for Small Data Behavioral Analysis

In this chapter, we introduce Lifestreams, its design principles, and its 4-tired software stack from feature extraction, feature selection, to inferences and visualization.
Chapter 4: Using Lifestreams for Real-World Behavioral Studies

In this chapter, we present the results and the lessons learned from applying Lifestreams to three real-world behavioral studies regarding 1) cardiovascular risk factors in young mothers, 2) familial wellness among family members, and 3) educational program that brings computational thinking to high school students.

Chapter 5: Small Data System Architecture

In this chapter, we introduce the notion of small data ecosystem and propose a modular system architecture for small data systems. We describe Lifestreams DB, the most central component in the proposed architecture, its design principles, and important implementation considerations. We then present the results of evaluating Lifestreams DB’s query performances and storage efficiency against real-world small data and a simulated workload.

Chapter 6: Small Data for Personalizing Online Experiences

In this chapter, we turn our attention to the second application context, Immersive Recommendations. We introduce the notion and challenges of Immersive Recommendations and review the prior work in the recommendation systems, hybrid recommendation algorithms, and cross-domain recommendations.

Chapter 7: Profiling Multi-Context Small Data

In this chapter, we introduce Context-Aware LDA, a novel topic-model-based user profiling model that simultaneously learns users’ interests from multiple small data streams generated in different contexts while suppressing the unique contextual noise and biases introduced by different contexts. We present the results of evaluating Context-Aware LDA’s profiling performance using real-world news and local
meetup event datasets.

Chapter 8: Making Recommendations With Immersive Profiles

In this chapter, we introduce collaborative user-item regression, a hybrid collaborative filtering model that combine multi-channel small data profiles, item contents, and the rating information to enable highly-personalized recommendations. We present the results of a large-scale offline evaluation and a small 33-person user study that evaluate the accuracy and utility of immersive recommendations.
CHAPTER 2

Small Data for Personal Behavioral Analysis

2.1 Introduction

In this chapter, we describe the application context of using small data for personal behavioral analysis and review the prior work in the related domains. We particularly focus on the applications of smartphone data due to their wide applicability in this context.

2.2 Personal Behavioral Analysis with Smartphone Data

Our collective small data provides a unprecedentedly detailed view to our daily life. Among these data, the digital traces collected by smartphones are particularly useful for personal behavioral analysis due to their ubiquitousness, unobtrusiveness, and rich sensing capabilities [LML11, ES10, API11]. Since smartphones’ debut in 2007, there has been a dramatic increase in their adoption. As in 2016, 72% of population in US own a smartphone [Pou16]. With the increasingly richer set of sensors on-board the phones, the smartphone in our pocket, which we carry with us throughout everyday life, has become a powerful tool to chronicle our activities and record the environments in which those activities take place.

For example, the accelerometer and gyroscope sensor data can be used to determine our mobility states, such as walking, running, riding a bicycle, or being in a vehicle; the GPS data and the ambient WiFi signals can be used to record
our location; and the acoustic data collected by microphone on-board the phone can be used to infer our social interactions and sleep qualities. These continuous background sensing capabilities make smartphones a valuable tool for monitoring and understanding an individual’s behaviors while requiring little effort from the individual themselves [LML10].

Another useful data stream enabled by smartphones is *in-situ self-reports* [THL15]. Traditional methods of self-monitoring through retrospective self-report are prone to errors and biases due to people’s limitations in accurately recalling prior events. For this reason, smartphones offer the advantage of supporting active data capture from participants in the context of their everyday lives and closer to real time [SSH08]. A structured, in context, form of real time self-report, referred to in the literature as Ecological Momentary Assessment (EMA), are developed to monitor affect, cognition, and behaviors in real time in a person’s natural environment. Error and bias are reduced by collecting and recording data in real time. On the other hand, such self-monitoring has been shown to encourage behavior changes and increases adherence by supporting self-awareness and self-efficacy.

### 2.3 Our Contributions

The key contributions we made to this application context are:

- We proposed **Lifestreams**, a 4-tiered software stack and extensible set of analytical building blocks that facilitates the exploration of diverse small data streams, in particular, the data collected through smartphones. (See Chapter 3)

- We demonstrated Lifestreams’ ability to highlight significant correlations and changes over time for single or across multiple simultaneous behaviors during a real-world 6-month study with 44 young mothers; and includes a
qualitative study based on interviews with 8 out of 44 moms who were able to make use of the Lifestreams visualization component guided by a research coordinator trained in the system (See Section 4.3).

- We additionally applied Lifestreams to two additional studies, one with continuous audio and another of larger scale, to further explore the generality and extensibility of Lifestreams (See Section 4.4).

In the following, we review the prior work related to this application context. Then, in the next chapter, we will describe our main contribution, Lifestreams.

2.4 Related Work

Personal behavioral analysis is a rather broad context. In this section, we review the prior works that focus on using smartphone data to enable chronic disease prevention and management, free-living behavioral studies, and self-wellbeing-management.

2.4.1 Smartphone Data for Chronic Diseases Management

Chronic diseases, such as diabetes, asthma, and obesity, account for 46% of global disease burden [ES10]. The traditional clinic-based treatment model is insufficient for improving chronic disease whose development and treatment mostly take place in the patients’ daily life rather than in the clinical settings. Over the past few years, many smartphone applications, known as Mobile Health (mHealth) applications, have emerged to make it possible for patients to collect and share relevant data with the care givers to allow more rapid convergence to optimal treatment [ES10]. Such applications are being developed in a variety of domains, including chronic pain management [SSA15], diabetes [Klo13, GLY16], asthma [HW09], depression [VMA13], etc.
Taking Rheumatologic (RA) disease management for example, an effective RA treatment requires physicians to make appropriate adjustments to the type and frequency of the dosing prescribed to the patients through closely observing the patients’ RA disease activity. However, RA disease activity is unpredictable with flares that last a couple of days to several weeks. In fact, managing an RA patient is similar to operating a complex feedback control system where the input variable is the RA medication and the output variable is the RA disease activities [SSA15].

Say et al. proposed to create a smartphone-enabled clinically informative measure, called Mobility Index, to assist physicians to better monitor patients’ condition [SSA15]. Specifically, they extract multiple dependent variables from the accelerometer and GPS/Wi-Fi location data. For each day, they compute (1) time spent walking/running, (2) gait speed, (3) distance travelled, (4) number of hours away from the house, etc. The decile value of each of these variables were computed, and then the Mobility Index was defined as the weighted average of the decile values of multiple variables in each day. They designed a calendar-based visualization in which Mobility Index was color-coded and superimposed onto a calendar to summarize the passive mobility measures that the rheumatologists confirmed would be clinically relevant.

For another example, Cardiovascular disease (CVD) is the leading cause of death among women. One of the major risk factors for CVD is weight. More than 60% of women in the United States are overweight [OCK14]. This situation is worsened when women become mothers. The Moms study aimed to assess the validity and reliability of using smartphones to monitor CVD risk factors of diet, stress, and exercise in young mothers [HTA13]. The study collected participants’ diet, stress, mood, exercise, and activity traces throughout the day for 6 months, in which the first 3 months of data were used to establish an individual’s baseline, and during the subsequent 3 months, moms received feedback that compared their current reports with their established baselines [HRK10].
We designed Lifestreams to support mobile health studies like the ones mentioned above. Such studies require systematical processing of a large amount of high-dimensional and multi-channel mobile health data. In their raw form, these data are highly variable and difficult to interpret (e.g. consider the data collected by accelerometer and GPS). Lifestreams’ data analysis building blocks transform these multi-dimensional, heterogeneous data streams into actionable and robust **behavioral indicators**. The hope is that such behavioral indicators can be used to characterize a user’s baseline, and to identify significant variations, trends and shifts in specific behaviors or symptoms that are relevant to an individual’s health conditions and disease activities [HTA13, ES10].

In addition to the requirements for data processing and analysis, a successful mobile health study also requires the support for robust data collection, secure data storage, and study managements. **Ohmage** ([http://ohmage.org](http://ohmage.org)) is a modular and extensible open-source mobile sensing platform that has been developed to fulfill these requirements [THL15]. The ohmage platform consists of three major blocks: (1) Ohmage backend, an online service that serves as a central datastore; (2) Mobile data collection apps, a collection of smartphone apps that run on the participants’ smartphones and collect various mobile health data. (3) Web-based tools for study management and administration; and (4) Web-based data analysis and visualization tools for exploring, analyzing, and visualizing the collected data. In fact, the proposed toolset, Lifestreams, is part of the ohmage’s data analysis blocks. The data collected by ohmage platform are used as input to the Lifestreams’ data analysis pipeline, and the analysis results are presented to the users in a web-based portal built into ohmage.

### 2.4.2 Smartphone Data for Behavioral Studies

Smartphones’ background sensing and in-situ self-report capabilities have enabled behavioral researchers to study people’s behavior under a *free-living* setting (i.e. in
people’s everyday life) as opposed to traditional survey based or lab based experiments. Many systems have been developed to support such behavioral studies. In the following we describe several representative systems and studies, and discuss their relation to Lifestreams.

**Reality Mining** was one of the early study conducted by Eagle and Pentland [EP06]. It gathered call logs, Bluetooth signals, cell tower IDs, and application usage data from 100 mobile phone users. They showed that such data can be used to understand users’ information use in different contexts, and recognize social patterns in users’ daily activities, such as relationships, socially significant locations, organizational rhythms. The studies like Reality Mining are the kind of studies Lifestreams was designed to support. However, rather than developing specific system for each particular study, we developed Lifestreams in a way that it can be used in many different behavioral studies with varying focuses.

**funf** was a mobile sensing framework developed by Aharony et al. that supported a large number of hardware and software sensors for Android devices [API11]. They used funf to deploy a 15-month study to investigate the relations between users’ financial status and other behavioral factors. De Montjoye et al. also deployed a study based on funf, in which they evaluated whether standard mobile phone logs can be used to predict the users’ Big Five Personality traits [MQR13]. They extracted a set of psychology-informed indicators from users’ location, phone calls, and texting records and trained an SVM classifier that achieves a prediction accuracy 42%-better than the random guess across the traits. Such predictions, if made more reliable, could provide an unobtrusive and cost-effective alternative to survey-based measures of personality.

**SystemSens** was a passive logging tool developed by Hossein et al. [FME11]. It was designed to help research studies capture users’ usage context. SystemSens collects phone usage data, including battery, CPU/memory utilization, network traffic, screen on and off, etc. These data provided insight into the usage context
of the phone even when the research studies are not directly related to these data. For example, it helped the researcher optimize deployed applications to consume less energy, and pointed them to external factors that were affecting the studies, such as issues with the phones or the bugs in the deployed applications.

Another important data stream for behavioral studies is affect and mood measurement, in particular, for the patients with depression, bi-polar, or other mental diseases. Affect state is frequently changing and difficult to recall and express. Pollack et al. developed Photographic Affect Meter (PAM), a novel smartphone app for frequent, unobtrusive measurement of affect \cite{PAG11}. The app prompts users with a grid containing photos, each of which depicting an emotional state. The app then asks users to select a photo that best describes how they feel right now. Based on users’ selection, PAM app generates the standard PANAS measurement for users’ arousal and valence, or positive affect and negative affect \cite{WCTSS}.

MoodSense developed by LiKamWa et al. was a smartphone app that passively collects phone usage data such as emails, web history, and calls to estimate the user’s mood \cite{LLL13}. They validated the inference accuracy with a two-month dataset collected from 32 users and found that a general mood model achieved 66% accuracy while a personalized mood model achieves 93% accuracy after over two months of training with individual users’ data.

We designed Lifestreams to be compatible with data collection tools like funf, SystemSens, MoodSense, and PAM. Specifically, the data collected by these tools were uploaded or exported to ohmage discussed earlier. Then, the corresponding feature extraction modules in Lifestreams were used to extract relevant features from these data. After that, the features from different sources can be compared or combined against each other to draw insights that individual data streams may not enable alone. (See Section 4.4.2 for example.)
2.4.3 Smartphone Data for Self-Management

A more consumer-facing application context of smartphone data is to help users manage and improve their wellbeing through personalized feedback and interventions enabled by smartphones. These applications use smartphone to track users’ daily activities and provide feedback to encourage users to adopt a healthier lifestyle.

**Ubifit Garden** was an early system in this domain [CMT08]. It used on-body sensors to measure users’ physical activity throughout their everyday life. The sensor readings were transmitted to the user’s phone through Bluetooth. Then, the phone inferred the users’ activity types and provided users with feedback by changing the wallpaper of the phone’s home screen. Specifically, the phone displayed an aesthetic image, presenting key information about users’ physical activities and goal attainment. A metaphor of “garden” was adopted in the presentation, where the flowers represented different activities performed and butterfly represented the goals attained. The participants in a 12-person-3-week field trial gave positive feedback to the usability of the program. In particular, participants mentioned UbiFit Garden helped them focus on planning or simply finding time for physical activities.

**BeWell** was a smartphone app that shared a similar idea with Ubitfit Garden, but, instead of using external sensors, it directly utilized the sensors on-board the Android smartphones to enable activities tracking [LML11]. It monitored users’ sleep, physical activities, and social interactions, and provided feedback to the user to promote better daily routines. BeWell’s feedback system was similar to that of Ubifit Garden, in which they adopted the metaphor of “aquarium” and displayed an animated aquarium in the phone’s home screen to encourage healthier daily routines.

**MyBehavior** was another smartphone app along this line of work, but took
the idea one step further [RAZ15]. In addition to providing key information about the users’ activities, MyBehavior actively generated personalized recommendations to encourage behavior changes. Specifically, MyBehavior automatically learned a user’s physical activity and dietary behaviors, and strategically suggested achievable changes to those behaviors. For example, it used Multi-Armed Bandit algorithm to generate diet plans that maximize calorie loss and are easy for the user to follow.

In addition to general well-being management, many consumer applications have been developed to support specific disease management and provides additional support mechanisms, such as goal setting and patient communities, to enable effective self-management. For example, the bant2 app was designed to facilitate the self-monitoring of lifestyle behaviors for diabetic patients [GLY16]. The app assesses data points in different contexts, identify positive and negative behaviours based on the analysis, and facilitates remedial decision making. The patients can use the app to set goals and receive reminders, participate in a social community, and accumulate reward points for positive behaviors.

Differing from the above-mentioned consumer-facing applications, we designed Lifestreams to be more focused on providing researchers or designers of interventions with insights to the study participants’ behaviors and facilitate the development and validation of intervention mechanisms. The researchers can conduct pilot studies with Lifestreams to validate their research hypothesis or intervention efficacy, and then repackage the Lifestreams’ data analysis capabilities into the consumer-facing applications.

2.5 Conclusions

In this chapter, we introduced the application context of using small data for personal behavioral analysis. We mainly focused on the use of smartphone data
to help individuals to improve health and well-being. This incorporates a variety of methods, including passive collection of activity, location, and communication, and prompted self-report. Such detailed data collection can be used to systematically monitor chronic conditions outside the clinical setting, enable behavioral researchers to conduct studies in free-living environments, and empower individuals to more effectively manage their wellbeing. In the next chapter, we introduce Lifestreams, our main contribution to this application context, that aims to facilitate the exploration and development of the above-mentioned applications.
CHAPTER 3

Lifestreams: A Toolset for Small Data Behavioral Analysis

3.1 Introduction

In this chapter, we introduce Lifestreams, a modular data analysis toolset for small data behavioral analysis. We describe the technical challenges Lifestreams designed to address, its design principles, and its 4-tiered software stack from feature extraction, feature selection and aggregation, to inference and visualization. While Lifestreams is also applicable to other kind of small data, in this chapter, we focus on its usage in smartphone data due to their wide applicability for personal behavioral analysis.

3.2 Challenges for Small Data Behavioral Analysis

As discussed in the previous chapter, small data, in particular smartphone data, have many advantages over the traditional data collection approaches for chronic disease management, behavioral studies, and self-wellbeing-management. However the characteristics of smartphone data and their application contexts also pose unique challenges. In particular, we focus on the following technical challenges in this thesis:
**Smartphone Data Transformation**

The sensor data collected by smartphones are highly variable and sometimes of very high frequency (e.g. accelerometer data can be collected at a rate of more than 100 Hz). These data, in the raw, are not immediately relevant or useful to the intended analysis. Consequently, we need specific data transform techniques to extract features that are relevant to the application at hand from these data. For example, the raw location data, consisting of a series of GPS coordinates and WiFi signals, is hardly relevant to the behavioral analysis. However, with place detection algorithms (described in Section 3.4.2), we can recover a sequence of semantically meaningful places users have been to, such as home, school, or work. Based on these features, we can further extract information, such as time leave or return home, time spent at work, etc., that are important to many behavioral analysis tasks.

**Drawing Actionable Insights and Reducing Information Overload**

The premise of using smartphone data for behavioral analysis is that we can extract valuable insights from such data. For example, smartphone data gathered from patients with certain medical conditions, if processed and presented to the care givers effectively, could improve care givers’ decision-making. Unfortunately, care givers are already on the brink of an information overload. For example, on average, a primary care physician might receive 1000 test results per week [BWT11]. Given the already excessive work of primary care physicians, data from patients’ smartphone applications is unlikely to be adopted by physicians if the clinical relevance is not immediately apparent [SSA15]. Much research needs to be done to identify measurements that are truly useful and determine the best way to present the results in a clinically relevant way.
3.3 Lifestreams Architecture and Design Process

The challenges of transformation of and extracting actionable insights from small data call for a system that can automatically and systematically turn raw data into useful features, identify key patterns, and present the end users with actionable information in an easy to understand way. In this thesis, we develop Lifestreams to achieve these goals.

Lifestreams is a modular and extensible open-source data analysis stack designed to facilitate the exploration and evaluation of small data analysis. The development of Lifestreams is driven by the Moms’ Study, an NIH-funded 6-month-long study focusing on cardiovascular disease risk factors among young mothers. 56 young mothers were recruited, and 44 of them carried the smartphones that ran the ohmage app (discussed in Section 2.4.1). Ohmage collects self-reports and passive accelerometry and location traces throughout the 6-month study period. The details of the study is described in Section 4.2. The design goal of Lifestreams is to allow behavioral and health science researchers (i.e. our end users in this case) to analyze high-dimensional multi-person smartphone datasets, gain insight into particular conditions, and ultimately design and evaluate the efficacy of intervention mechanisms. In this work, most of our analysis focuses on diet, stress, and exercise data collected during the study as they are the three major factors for cardiovascular diseases.

3.3.1 Architecture

The Lifestreams stack consists of four layers–1) feature extraction and aggregation, 2) feature selection, 3) inference and 4) visualization. Each layer consists of modular plugins or building blocks that can process the data provided by the lower layer, and then send the result to the next layer up in the stack. Figure 3.1 shows a diagram of the Lifestreams analysis stack.
Figure 3.1: Lifestreams Data Analysis Stack

- At the bottom of the stack are different personal data streams. For our deployment and analysis, we used data collected by ohmage, an open-source personal data collection platform described earlier in Section 2.4.1. These streams include intermittent self-report data (both prompted and user-initiated) and passive data streams collected from sensors and applications on-board the mobile device (e.g. accelerometer data, location traces, communication app usage, etc.).

- These data streams are then sent to the feature extraction layer, which is the first step in transforming them from raw datasets into actionable behavioral indicators. For example, the raw accelerometer and location streams are passed to an Activity features module to be transformed into meaningful classifications of activities (i.e. still, walk, run, drive). After features are extracted, they may be further aggregated based on spatial and temporal attributes.

- Features produced by the feature extraction layer are multi-dimensional with many of them irrelevant or redundant to the analysis goal, and therefore need to be further processed by the feature selection layer for dimensionality reduction.
reduction.

- The inference layer uses the selected features to detect patterns and trends, and infer behavioral states.

- Visualizations, tightly coupled with the analytical building blocks, make the analyses available and actionable for the end user.

### 3.3.2 Interactive Design Process

Using Lifestreams’ data analysis pipeline, we present three sets of analytical results based on inference methods developed to explore participant’s behaviors. These inference methods were developed in collaboration with subject matter experts, using the Lifestreams stack to identify significant behavioral patterns in several specific participants. The accuracy and utility of the analytical building blocks were qualitatively verified through the interviews with the participants, in which Lifestreams visualization was used to guide some of the discussions (see Chapter 4).

Lifestreams’ design process has heavily depended on the feedback from study participants and the study coordinator who attempted to use Lifestreams to make sense of the data collected during the study. In order to explore the generality and extensibility of Lifestreams pipeline, we further applied Lifestreams to two additional studies, both of which introduces at least one new type of data streams, such as audio data and mobile phone usage data streams (see Section 4.4). In all three studies, Lifestreams’ integrated analysis pipeline was able to identify key behaviors and trends in the data that were not otherwise identified by participants.

In the remaining part of this chapter, we describe each layer in the Lifestreams’ data analysis stack and highlight the important modules.
3.4 Feature Extraction and Aggregation

We use the term feature extraction to refer to a process of transforming input data into a representative set of features that describe the data and could be used to calculate a behavioral indicator. For example, the activity feature module described below produces a stream of important features related to a user’s activities throughout the day, using the accelerometer, Wi-Fi signature, and location data streams. Through collaborations with clinicians and patients, we have identified several disease domains in which activity and location patterns are relevant to determining the patients’ status. These diseases include depression, chronic pain, gastro-intestinal, inflammatory, and auto-immune disorders. For each disease and demographic in which such measures might be used, domain-specific exploration and knowledge is essential. Here we focus on the system support needed to define and refine the use of this type of data for a specific study or intervention. In the following sections, we introduce each Lifestreams’ feature extraction module in more detail.

3.4.1 Activity Features

Lifestreams currently implements the following activity features:

- **Mobility state per minute** Mobility states are still, walk, run, and drive; each minute of a stream is annotated with the associated mobility state produced by the classifier. While it is susceptible to noise from individual classifier errors, it allows short events on the order of minutes (not seconds) to be identified robustly, e.g., a walk to the water cooler or waiting for a long stop light.

- **Mobility Events** Mobility events are a higher-level set of longer periods of activities, providing an overview of the general activity level for the day.
This gives a better indication of how sedentary and active users were, and when they had to travel. This feature includes start and stop time for the event and the associated mobility state.

- **Total Distance Travelled** Each mobility event is also annotated with the distance travelled during that event. Total distance traveled is used to determine how strenuous an ambulatory activity is, or how long a drive is. This can help determine activity levels or how much the user had to travel.

- **Geo-Diameter** The longest distance between two locations included in a mobility event. Geo-Diameter is useful for determining how far from home a user traveled during the day, and offers additional information to the health researcher beyond total distance traveled.

Activity events are calculated using accelerometer, Wi-Fi fingerprints, and location traces collected on the phone by ohmage. The classification is based on supervised machine learning techniques [LRE10]. The activity classification is performed in two stages. First, the accelerometer records one-second worth of triaxial samples. To be orientation-independent, it first calculates the magnitude of acceleration and then calculates the variance and FFT coefficients on the set of values. These are used to classify ambulation, i.e., sedentary, walking, or running. If the user is classified as sedentary, then the current Wi-Fi fingerprint is compared to the access points encountered in the past 10 minutes. If the fingerprint has not changed, the user is classified as still. If it has changed, the user is presumed to be driving. If there are not enough Wi-Fi access points for the comparison, then GPS speed is used to determine whether the user is driving or stationary. This specific activity classifier is modular and can be replaced by other classification techniques, such as in [RMB10] and [SLF08].

Our activity classifier produces a series of activity estimates on the order of one per minute. Since user routines consist of blocks of activities, we calculate
activity events, that is, periods of time during which the user was in a single activity. To prevent noise resulting from occasional miscalculations, we smooth the data, ignoring brief changes that revert to the previous activity, such as one instance of walking during a period when the user is still. There is an obvious tradeoff between responsiveness to short events and noise reduction; features can be calculated with various degrees of smoothing depending on the objective of the study and the noisiness of the data source.

Once periods of activity are identified, features of the user’s routine are calculated from event properties, such as the start and end times and duration of the event, and the number of events occurring within a time window. Features are also calculated based on alternate activity labels. Rather than handling each activity class separately, we aggregate results of the same type. For example, total cumulative time spent sedentary or in physical activity might be used as a behavioral indicator in some contexts. The features largely depend on the domain and how they are used. For example, if it does not matter whether the user is sedentary in a moving vehicle or in a chair at home, the Lifestreams user can aggregate drive and still into a single activity class, sedentary.

3.4.2 Semantic Location Features

Lifestreams also produces a stream of important features related to semantic locations throughout the study [KHG09]. Each location data point collected by ohmage contains a Wi-Fi fingerprint and a GPS coordinate. From this raw data stream, we first use a place detection algorithm to identify potentially meaningful places, such as home and office, etc. Then, we compute the following features:

(a) **Duration at a Place**: The amount of time in a day that a participant stays at a place. (b) **Arrival and Leaving Time To/From Places** (c) **Daily Work Schedule Deviation**: The deviation of a participant’s daily work schedule from her normal working routines, which is defined as the time difference between
the arrival time to work and the median of the arrival time to work among all 
the working days during the study period. Such deviations have been shown by 
behavioral research to correlate with patients’ physical, psychological, and social 
outcomes.

The prerequisite of the above-mentioned features is an effective place detection 
algorithm. In the following, we describe such an algorithm that is designed to 
efficiently identify places from a large amount of location traces.

3.4.2.1 Two-Phase DBSCAN for Place Detection

A common approach to discover meaningful places from location traces is to as-
sume that a place is a location where the user stays for more than a certain period 
of time [AS03]; such locations are revealed with clustering algorithms, where each 
resulting cluster is regarded as a place. A density-based clustering algorithm, 
called DBSCAN, is commonly used for this purpose [ZFL04]. DBSCAN defines
two points as *neighbors* if their distance is below a threshold denoted by $Eps$. If a point has more than $MinPts$ of neighbors (i.e. its local density is high), this point and its neighbors are declared as a cluster. Two overlapping clusters will be merged into one larger cluster recursively until no cluster is overlapped. The advantages of DBSCAN are three-fold: (1) it can reveal places of arbitrary shapes; (2) it can work on location traces with arbitrary number of places; and (3) its final clustering result does not depend on the initial random assignment of the clusters (i.e. its result is deterministic) [ZFL04].

However, DBSCAN’s complexity is $O(n^2)$, where $n$ is the number of data points, and therefore does not scale well to location traces collected in real-world studies where $n$ could be quite large. To address this problem, instead of extracting places from all data points at once, we adopted a two-phase process to first extract daily places based on 24 hours of location traces and then merged similar places extracted throughout the study into a final set of places. DBSCAN is used to identify places in both phases as follow:

- **Phase 1 DBSCAN** extracts the ambient signal signatures that characterize the Wi-Fi and GPS signals of each place

- **Phase 2 DBSCAN** merges the places based on the similarity of their ambient signal signatures and generates a final set of places.

The time complexity of our two-phase algorithm is $O(K^2D + M^2)$; where $K$ denotes the number of data points collected daily, $D$ denotes the number of days, and $M$ denotes the total number of places generated in the first phase. As shown in Figure 3.2, when the study duration increases from one day to fifteen days (i.e. increases by 15x), the execution time of our Two-Phase DBSCAN approach increases 24.6x as compared to 92.6x of increment with the traditional DBSCAN. More importantly, our approach can run incrementally (referred to as Two-Phase DBSCAN (incremental) in Figure 3.2). If all places from previous days are cached,
we only have to run the first phase for those days that have not been processed, and re-run the second phase to update the final place assignments. These advantages make our approach more appropriate for processing evolving location traces for longer term studies.

### 3.4.3 Features Extracted from Self-Report Data

Unlike the feature extraction modules that extract features from passive data streams, the module that extracts self-report data features results in a dimensionality expansion from the original dataset.

Self-report data, captured in the form of surveys, are classified into one of the following data types: (1) quantitative (e.g. number), (2) text, and (3) categorical (e.g. single choice/multiple choices). The quantitative data type is used as is. We have not yet implemented text feature extraction but anticipate future NLP plugins to extract features such as vocabulary difficulty and sentiment analysis. In this section, we describe the transformation of the categorical data type which is commonly used in survey questions since it is easy for participants to respond.

There are two types of categorical features: ordinal and nominal. For an ordinal feature (e.g. food quality rating as high, medium, low options), a rank or number is assigned to each element to indicate the order among categorical elements. These assignments are usually done by the study researcher and the ordinal values are derived from the study configuration. For a nominal feature (e.g. cause of stress as relationship, financial, school/work, etc. options), all options are independent and have no ranking among them. In our analysis, each option is transformed into a binary feature with the value equal to one if the option is selected and zero otherwise.

For a set of features that are possibly correlated, Principle Component Analysis (PCA) can reduce them to a smaller set of features. In our analysis, we found
high correlation among the ten options of the mood questions including happy, calm, stress, etc. These options are highly correlated as both positive and negative mood are not expected to be present simultaneously. Therefore, PCA is applied to transform these features into a single feature that will have a higher value when more positive mood options are selected, and vice versa. Data aggregation

All features derived from different modules can be further aggregated, when appropriate, based on statistical functions (e.g., min, max, or average), time (e.g. daily or weekly) and/or location. For example, all extracted features can be aggregated over an arbitrary time period and starting timestamp, both of which are adjustable parameters. For our analysis, we aggregate these features over 24 hour periods starting at midnight on the first day of the study to represent daily data points.

3.5 Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. Redundant or irrelevant features are removed during the feature selection process. Feature extraction creates new features from the raw datasets, while feature selection returns a subset of these features. We describe a simple feature selection module based on pairwise correlation analysis. Features with high correlation, using pairwise correlation analysis, are then highlighted. Other feature selection modules can be substituted, such as principal component analysis (PCA), and minimum Redundancy Maximum Relevance (mRMR) [PLD05].

3.5.1 Pairwise Correlation Analysis

A correlation matrix is one of the most common methods used to summarize relationships between any two features. However, when dealing with a multi-
A dimensional heterogeneous dataset where different features are of different types, one correlation measure cannot simply be applied to all features. Five different correlation measures are used to compute the correlation coefficients between quantitative, ordinal, and binary (nominal) features (see Table 1).

The proper hypothesis test of each correlation measure is performed with no correlation as the null hypothesis. A desired significance level (with 0.05 as a default) can be specified. Any correlation that does not have sufficient significance level will be excluded. The pairwise correlation coefficients by default are calculated across all data. However, the user can choose to calculate correlation coefficients on subsets of the data through an interactive interface (described in Section 3.7); for example, based only on weekday or weekend data to investigate weekday/weekend effect.

3.6 Lifestreams Inference

Inference is a process of arriving at some logical conclusions from premises known or assumed to be true. The Lifestreams inference layer focuses on techniques that enable researchers to understand user contexts, behaviors, and changes, based on selected features. We provide three different inference methods to investigate long-term relationships between participants’ behaviors and to detect behavior changes from the participants’ behavioral data streams.

3.6.1 Correlation Summary

The pairwise correlation coefficient is useful for exploring relationships or dependencies between any two sets of time-series data. A strong and significant correlation indicates consistent relationship throughout the entire time-series. The pairwise correlation analysis module calculates a three-dimensional (3D) correlation matrix across all pairwise features and participants. However, the 3D matrix
is large and hard to visualize. The correlation summary module is a thin layer on top of the pairwise correlation analysis that provides different capabilities to quickly shift through the large 3D data and draw conclusions about long-term relationships, such as pairwise correlation for an individual and similarity across individuals.

The 3D matrix can be filtered by individual participant and by a threshold value to produce a 2D correlation matrix for that individual. The 2D matrix shows only features with correlation coefficient values stronger than the specified threshold (e.g. 0.3). This capability allows researchers to quickly explore pairwise features with different coefficient values. A statistical summary matrix capturing similarity across all participants based on different coefficient thresholds can be calculated. This matrix shows, for each pair of features, the number of participants with the correlation coefficient greater than or equal to a specified threshold. We can further filter this matrix by only showing pairwise features with the number of participants greater than or equal to a specified value (e.g. 25%). This capability will allow researchers to identify common pairwise relationships among participants. We show examples of these plots based on our pilot data in Section 4.3.

3.6.2 Change Detection (Single Feature)

The ability to detect changes in an individual’s behaviors will enable behavioral and health sciences researchers, as well as context-aware applications, to react accordingly and to provide active-interventions to participants. For example, a mobile app could display stress reduction techniques to a participant when changes that have previously-resulted in higher-level stress are detected. In this section, we introduce the change detection module that detects behavior changes on each individual feature. We will introduce another change detection module that detects the correlation changes between pairwise features in the next subsection.
Figure 3.3: A walkthrough of the change detection algorithm

We model the behavior change detection problem as a statistical change detection problem. We detect behavior change by comparing the distributions of recent observations of the participant’s behavior with previous observations, and identifies a potential behavior change when these two distributions are significantly different. Such statistical change detection approaches have been applied in many fields, such as finance, robotics, and quality control [Lai95, FCN01]. Our iterative procedure for behavior change detection algorithm works as follows given a sequence of observations \( x_1, \ldots, x_t, \ldots \), where \( x_t \) is the observation at time \( t \). Suppose we want to know if any behavior changes have occurred before time \( t \). We apply a two-sample test to every possible splitting point \( t' \in (1, t) \) and compute the test statistic \( D(t', t) \) that estimates the degree of difference between the data distributions of \( x_1, x_2, \ldots, x_{t'} \) and \( x_{t'+1}, x_{t'+2}, \ldots, x_t \). Letting \( t^* \) denote the time \( t' \) that maximizes the statistic \( D(t', t) \) among all the possible splitting point \( t' \), we compare this maximum statistic (i.e. \( D(t^*, t) \)) with a predefined threshold. If \( D(t^*, t) \) exceeds the threshold, we declare that a change has occurred at time \( t^* \). After a change point has been identified, the observations before the change point
that are, \(x_1, \ldots, x_t\), will be discarded, and the procedure restarts.

The core design decision of such a statistical change detection algorithm is to define a test statistic that best suits the characteristics of the observations. Behavioral data is known to be non-normally distributed and might not satisfy the assumptions of many parametric statistics \cite{Coh92}. Therefore, we adopt the Mann-Whitney (MW) test, which is a non-parametric test, to estimate test statistic \(D(t', t)\) for quantitative and ordinal behavioral data streams. The MW test performs a two-sample test based on the ranks of observations instead of the raw values. Although it is less sensitive than Student-t test, we found the MW test to be a more appropriate choice for behavioral data. For categorical binary data streams, we adopt Fisher’s exact test, which is a common test used to detect distribution changes in a Bernoulli sequence \cite{Ros13}.

Two parameters have to be determined for this algorithm. \textit{Startup time} is the number of observations after which detection begins; it is necessary since the statistical test has low power for a small sample size. \textit{ARL0} is the expected time difference between a false positive detection and the real change point; the size of ARL0 determines the threshold levels at different times \(t\). There is a tradeoff between the responsiveness of the detection algorithm and the expected duration of the detected changes. We will investigate this tradeoff in Section 4.3.2 in more detail based on participants’ data.

Figure 3.3 provides a walkthrough of our change detection algorithm. Each green circle on the bottom plot indicates an actual observation on a different Day \(t\), and each blue circle on the top plot indicates the \(D(t^*, t)\) for that day. The algorithm will not start to detect any changes until after startup time (20 days). Initially, there are a number of anomalies observed between Day 20 and Day 30. However, these anomalies do not last long and are not consistent enough to trigger change detection. From Day 98, the algorithm starts to observe more consistent higher-level outcomes, and on Day 111, when the computed \(D(t^*, t)\) exceeds a
specified threshold, the algorithm determines that a change was observed on Day 98.

An R package, called cpm, is used to implement the above-mentioned change detection algorithm and determine the appropriate threshold [Ros13]. Since many individuals have different patterns during the week and weekend, we introduced a weekday/weekend/all parameter (with weekday as a default) to identify different subsets of data for analysis.

### 3.6.3 Correlation Change Detection (Pairwise)

The pairwise correlation described in Section 3.5.1 is useful for exploring long-term relationships between any two behavioral features. However, the overall pairwise correlation cannot capture the changes of relationships between two features and may overlook important information. The change detection on pairwise correlation addresses this issue. The detection can be used to identify changes in behavior or to discover the shorter-term relationships that could be overlooked in the overall correlation analysis.

We introduce two techniques — pairwise correlation change detection and moving average to further investigate changes in behavior based on pairwise features. The correlation change detection is modeled in a similar way as the single feature change detection. The main difference is that the test statistic for deriving $D(t^*, t)$ is calculated based on two sets of observations from the both features. We implemented a detection algorithm developed by Wied et al [WG13] to detect the pairwise correlation changes. There are two parameters required for this algorithm: (1) start time $m$ assumes that the correlation will remain constant among the first $m$ observations. (2) $r$ is a parameter that determines the threshold function. A higher value of $r$ results in a higher false alarm rate, but with shorter detection delay. We select the parameter values $m = 20$ and $r = 0$ as a default.
for a longer detection time, but lower false positive rate.

Moving average is a technique commonly used in time-series data analysis to provide an estimate of the trend of the observations. Moving average smooths out short-term fluctuations and reveals the longer-term trends of the data. Each data point is an average of a subset of data extracted around the original raw data point e.g. a subset of 7-day data starting from the current day.

In our analysis, we transform the original dataset using the seven-day moving average before applying the pairwise correlation change detection algorithm. Similar to single feature change detection, a parameter weekday/weekend/all can be specified.

3.7 Lifestreams Visualization

The purpose of Lifestreams visualization is two-fold: (1) to allow researchers to quickly and flexibly navigate through high-dimensional personal data streams by focusing on important trends/patterns, and (2) to explore visual aids for an intervention, such as to guide discussions with patients in clinical or coaching sessions. It allows users to interactively select different feature groups (e.g. diet, stress), switch between different visualization views, adjust visualization parameters, and export the visualization results. We describe a few examples of visualization views below.

- **Behavior Change View** displays time-series plots of individual features and allows users to select only features that contain potential change points during the study (Figure 3.4a). This function helps researchers select features with potentially-noteworthy trends from a large feature pool.

- **Place View** marks the locations that are potentially meaningful to a participant on an interactive map interface (see Figure 3.4b). These locations
Moving average is a technique commonly used in time-series data analysis to provide an estimate of the trend of the observations [8].

In our analysis, we transform the original dataset using the seven-subset of 7-day data starting from the current day. A subset of data extracted around the original raw data point, such as to guide discussions with patients in clinical or coaching sessions. It allows users to interactively select different feature groups, switch between different visualization views, adjust parameters, and export the visualization results. Users can use Lifestreams visualization to quickly and flexibly navigate through vast and high-dimensional personal data streams and create visual aids to guide the discussion with patients or participants. (Note that the geo-information in the Place View has been obfuscated to protect the participant’s privacy.)

Figure 3.4: Lifestreams visualization allows users to interactively select different feature groups, switch between different visualization views, adjust parameters, and export the visualization results. Users can use Lifestreams visualization to quickly and flexibly navigate through vast and high-dimensional personal data streams and create visual aids to guide the discussion with patients or participants. (Note that the geo-information in the Place View has been obfuscated to protect the participant’s privacy.)

are identified by the Location Analysis module described in Section 3.4.2. In interviews with the participants, we found this interactive map, along with the time-related statistics, such as the amount of time spent in each place, and the arrival and departure time, are particularly useful in helping participants to recall the semantic meanings of different places (e.g. home, work place, gym, children’s schools, and etc.). This semantic information is in turn used in transforming the participants’ raw location traces to more meaningful behavioral features.

- **Day View** uses ANalysis Of VAriance techniques (ANOVA) to evaluate how a patient’s behaviors vary throughout the day. For example, Figure 3.4 shows that this participant’s stress level and mood tend to have statistically significant variation at different times of day (i.e. the p-value ≤ 0.05). Such
analysis is particularly useful when self-report data exhibit high variation and recall biases depending on the hour, such as how much food, how hungry, current stress levels, etc.

- **Time View** plots the change points of different features on a single timeline (see Figure [3.4d]). It provides an overview of a participant’s behavioral trends and suggests potential interactions among different behaviors (e.g. an increase in stress level shortly after an increase in working hours).

### 3.8 Conclusions

In this chapter, we described Lifestreams, a 4-tiered data analysis stack with an extensible set of analytical building blocks to facilitate the exploration of diverse personal data streams and identify important behavioral patterns. Lifestreams’ existing analysis modules are not meant to be comprehensive. They provide an initial set of techniques that allow us, and the broader community, to start our data exploration and to validate the techniques and findings. We focus on modularity so that Lifestreams can be extended and generalized to a wide range of research studies with varying types of data streams and research goals. In the next chapter, we will present the analysis results of applying Lifestreams to three different real-world personal behavioral studies.
CHAPTER 4

Using Lifestreams for Real-World Behavioral Studies

4.1 Introduction

In this chapter, we explore the accuracy and utility of Lifestreams by applying it to the data collected from 44 Moms who carried smartphones in a 6-month, NIH-funded study of cardiovascular disease risk factors for young mothers. We compare the output from the analyses with qualitative information obtained directly from the participating Moms about their diet, stress, and exercise. We find that in many instances, Lifestreams is able to automatically identify trends and changes that were confirmed by participants.

4.2 Dataset: Diet, Stress, and Exercise in Young Moms

Behavioral researchers used Lifestreams to study cardiovascular risk factors in young mothers. The purpose of the study was two-fold. First, to evaluate the validity and reliability of the phone in capturing diet, stress, and exercise for women; and a secondary aim to evaluate the efficacy of using the phone for behavior change. Basic measurement parameters included a participants daily exercise routines, their diet, and their stress and mood levels throughout the day measured by survey; and a participants accelerometry, location traces, and mobility states recorded continuously using our automated mobility classifier.
56 young moms based in the Los Angeles area participated in the study between January 2012 and March 2013. Among the 54 moms who completed the study, the experimental group consisted of 44 moms who used our smartphone application, ohmage, to collect data. Among these 56 participants, the 44 moms' ethnicity and races were diverse and their ages range from 18 to 40 with an average of 30 years old. 15 moms worked full time, 24 moms worked part time including studying, and 17 moms worked in the home. All moms had at least one child living at home. The study start date of each mom varied from January 2012 to September 2012. 52 moms had completed their study with an average duration of 7 months. 4 moms dropped out in the middle; two moms became pregnant and the other two did not specify. Lifestreams visualization was ready for use in January 2013 for a qualitative study involving 8 out of 44 moms during their in-person interview discussions.

15,599 survey responses were collected across 44 moms in the experimental group. The responses are distributed uniformly across the four surveys, 4,248 (27%) morning surveys, 3,968 (25%) midday surveys, 3,722 (24%) late afternoon surveys, and 3,661 (24%) bedtime surveys. The average of survey numbers answered per user is 354, and participants answered 2 surveys on average per day during periods where participants answered at least once.

A total of 115,228 questions were answered, with an average of 2,619 questions per participant and 16 questions per participant per day. The most popular questions participants persistently answered were "Have you eaten since you completed your last diet survey?" at midday, "Have you felt stressed in the last two hours?" in the morning, and "How many hours in total did you sleep last night?"

The participants could choose to turn on or turn off the mobility data collection. 3,834 days of mobility data were collected from 44 moms with a minimum of 5, a maximum of 202, and an average of 87 days. Since the number of the entire study days across all moms is 7,272 days, participants contributed mobility data
during one-half of their study period on average.

4.3 Lifestreams Data Analysis Results

The analysis presented here uses self-report survey data and passive accelerometry and location data streams from the Moms. The survey data consists of 47 questions grouped into 4 different surveys for participants to complete in the morning, midday, late afternoon and evening. The self-report data are transformed into extracted features using techniques described in section 2.1.3 including nominal data transformation and PCA. A set of activity and location features was extracted from the passive activity and location data streams. All data are aggregated into a daily summary. At the end of the feature extraction and aggregation process, there are 142 features extracted for each participant. In the remainder of this section we discuss results derived using several inference modules used in this study. In each section we describe findings for a single exemplar participant in order to highlight the utility of the modules. Agreement between events and relationships identified by the modules and qualitative data from participants indicates the utility of the modules in automatically identifying important features that could not otherwise have been found through manual parsing.

4.3.1 Correlation Summary

Correlation analysis allows researchers to infer high-level, long-term, behavioral patterns. We use Cohen’s conventions to interpret correlation coefficient of behavioral data \[\text{Coh13}\]. A correlation coefficient of .10 is thought to represent a weak or small association; correlation coefficients of 0.30 and 0.50 are considered as moderate and strong correlation, respectively. Only significance levels greater than 0.05 (i.e. \(p\)-value \(<\ 0.05\)) are considered as statistically significant.
In this section, we explore the accuracy and utility of Lifestreams in a real dataset. All participants were moms who had at least one child living at home. The dataset covered a total of 115,228 questions answered by 44 moms, with an average of 2,619 questions per participant and 16 questions per participant per day. The study ran for an average duration of 7 months, with 4 moms dropping out in the middle. 52 moms completed the study, with a range of study start dates from January in 2012 to September in 2012. The participants' ethnicity and races were diverse, with ages ranging from 18 to 40 with an average of 30 years old. 15 moms worked full time, and 24 of the moms became pregnant during the study.

The analysis used self-report survey data and passive recorded data, including measures of mobility and activity patterns. The analysis presented here uses self-report survey data and passive recorded data to identify important features that were not otherwise identifiable. These analyses are intended as illustrative examples that could be used to infer her long-term behavioral patterns.

### 4.3.1.1 Individual 2D Correlation Matrix

A participants 2D correlation matrix among pairwise features can be used to infer her long-term behavioral patterns. Figure 4.1 shows a specific participants 2D correlation matrix where only the pairs of features that exhibit correlation greater than or equal to 0.30 are shown.

This matrix shows many interesting relationships between different aspects of the participants behavior. For example, her daily work schedule deviation (1st row) shows moderate correlations with both her daily overall stress level (5th row) and the late afternoon stress level (11th row). Note that the work schedule deviation is defined in Section 2.1.2 and the daily overall stress level is the stress level that a participant rated before bedtime. These correlations between work schedule and the mentioned stress levels were confirmed by the participants she mentioned that her occasionally-varying work schedule was one of her causes of stress since she could not maintain her regular routines on those days when her work schedule changed. This result also aligns with behavioral studies of working

---

Figure 4.1: Individual 2D correlation matrix of a participant. This figure shows correlations between the participants working schedule, stress levels, exercise routines, and daily activity patterns.

| 1 | Work Schedule Variance | 0.45 |
| 2 | Time Arrive Work | 0.45 |
| 3 | Time in Old House (Place 2) | 0.35 |
| 4 | Time in New House (Place 1) | -0.38 |
| 5 | Overall Stress Level | -0.28 |
| 6 | Stress Caused by Traffic | -0.83 |
| 7 | Stress Caused by Finances | -0.28 |
| 8 | Time for Self | -0.51 |
| 9 | Stress Level in Morning | 0.55 |
| 10 | Stress Level in Mid Day | 0.56 |
| 11 | Stress Level in Late Afternoon | 0.23 |
| 12 | Stress Level in Morning | -0.34 |
| 13 | Stress Level in Mid Day | 0.32 |
| 14 | Stress Level in Late Afternoon | 0.31 |
| 15 | Time for Self | -0.38 |
| 16 | Stress Level in Morning | -0.28 |
| 17 | Stress Level in Mid Day | -0.36 |

---

**Correlation Coefficient**

-1.0 ≤ Correlation Coefficient ≤ 1.0

**Legend**

-0.5 < Correlation Coefficient < 0.5

-1.0 ≤ Correlation Coefficient ≤ -0.5

0.5 < Correlation Coefficient < 1.0

---

| Time Arrive Work | 0.45 |
| Time in Old House (Place 2) | 0.35 |
| Time in New House (Place 1) | -0.38 |
| Overall Stress Level | -0.28 |
| Stress Caused by Traffic | -0.83 |
| Stress Caused by Finances | -0.28 |
| Time for Self | -0.51 |
| Stress Level in Morning | 0.55 |
| Stress Level in Mid Day | 0.56 |
| Stress Level in Late Afternoon | 0.23 |
| Stress Level in Morning | -0.34 |
| Stress Level in Mid Day | 0.32 |
| Stress Level in Late Afternoon | 0.31 |
| Time for Self | -0.38 |
| Stress Level in Morning | -0.28 |
| Stress Level in Mid Day | -0.36 |

---

**Correlation Coefficient**

-1.0 ≤ Correlation Coefficient ≤ 1.0

-0.5 < Correlation Coefficient < 0.5

-1.0 ≤ Correlation Coefficient ≤ -0.5

0.5 < Correlation Coefficient < 1.0

---

**Legend**

-0.5 < Correlation Coefficient < 0.5

-1.0 ≤ Correlation Coefficient ≤ -0.5

0.5 < Correlation Coefficient < 1.0

---

**Correlation Coefficient**

-1.0 ≤ Correlation Coefficient ≤ 1.0

-0.5 < Correlation Coefficient < 0.5

-1.0 ≤ Correlation Coefficient ≤ -0.5

0.5 < Correlation Coefficient < 1.0

---

**Legend**

-0.5 < Correlation Coefficient < 0.5

-1.0 ≤ Correlation Coefficient ≤ -0.5

0.5 < Correlation Coefficient < 1.0
The matrix also reveals a strong negative correlation between the daily amount of time she spent in Place 1 (new house; 4th) and Place 2 (old house; 3rd row); which is consistent with the participant moving into a new house during the study period. Moreover, the correlation matrix highlights moderate to strong correlation between her new house and lower stress levels, less concern about finances, and increased time for herself; all of which were confirmed by the participant. In addition, when she had a larger daily geo-diameter (15th row) or a longer driving distance (16th row), her cause of stress was more likely due to the traffic (6th row). Finally, there is a strong correlation between her reported exercise time and the daily walking time detected by our activity classifier. This was also verified by the participant, who reported that her primary exercise activity was walking around the neighborhood.
4.3.1.2 Correlation Statistics Across All Participants

In addition to the 2D correlation matrix for an individual, a population-wise 3D correlation matrix can provide researchers with an overview of significant patterns among the population, and can also help an individual understand how they fit in with the broader population. Figure 4.2 shows a selective set of pairwise features that have at least 25% of participants with correlation coefficients higher than 0.3. Note that the number of participants and/or coefficient levels can be changed to different thresholds. For some pairwise features the data show consistent correlation across multiple participants. For example, there are 11 participants that have a negative correlation between whether they exercised or not (2nd row) and daily overall stress level (12th row), and between the time they had to themselves (8th row) and daily overall stress level. Moreover, interestingly, comparing the stress levels at different times of day and the overall stress level, only 8 participants show positive correlation between the reported morning stress level (11th row) and the overall stress level (12th row), while 19 and 20 participants show positive correlation between the reported mid-day stress level (10th row) and the overall stress level, and the late-afternoon stress level (9th row) and the overall stress level, respectively.

In addition, for some pairwise features (e.g. whether a participant exercised vs. planning to exercise), the data show both positive and negative correlation among different participants. For example, for one participant, having a plan to exercise (3rd row) has a negative correlation with whether the actual exercise took place (2nd row).

4.3.2 Single Feature Change Detection

We found the single feature change detection algorithm (described in Section 3.6.2) to be one of the most useful building blocks in helping researchers make
Figure 4.3: Sample single feature change detection results for a participant (Startup=30 days, ARL0=500). This figure shows changes detected across daily stress levels, diet, exercise routines, and driving distances.

sense of behavioral data. Feedback from participants and the study coordinator was consistent with this finding.

Figure 4.3 shows sample change detection results for participant 016. The solid vertical lines in the plots indicate the change points estimated by our change detection algorithm, and the dashed lines following the solid vertical lines indicate the time when the corresponding change is detected by the algorithm. Based on the information we learnt in a follow-up interview with the participant, we found that the change detection results were consistent with the participants real life events. This participant is over-weight and reported that she became more concerned about her health around the beginning of June. This can be seen by the increasing number of times she reported her health as the cause of the
Figure 4.4: Tradeoff between the responsiveness of change detection and the time the detected changes are expected to last. With a shorter startup time or a smaller ARL₀, the algorithm detects changes earlier, but also tends to include temporary changes.

stress (3 is the maximum number per day), and the increasing daily overall stress levels, both of which are detected by our algorithm. In addition, she also reported that she stopped eating at restaurants and enrolled in a weight-watcher program, which corresponds to the improvement in food quality detected in June and July. Moreover, the participant reported taking a two-week family trip at the beginning of July, which can be seen from the increase in her driving distance as well as the decrease in the walking time around the same period.

Even though Lifestreams cannot know the specific events that caused the detected changes, the visualization and analytics may help a study coach narrow in on important timeframes and behaviors where significant changes occur. Automated analyses for example, could help health care providers give feedback when stress levels begin increasing, and identify the triggers of changes to deploy proper interventions.

As shown in Figure 4.3, the change detection algorithm usually detects changes within a month after the changes occurred, and some of the changes were tem-
porary (e.g. the increase in driving distance) while some of them lasted longer (e.g. the increase in stress level). There is a tradeoff between the responsiveness of change detection and the expected duration of the detected changes; and this tradeoff can be made by adjusting the two parameters of the change detection described in Section 3.6.2. Figure 4.4 shows the effects of different values of these two parameters. Each point in the figure is an average of all the changes detected in 142 features of 44 participants, and the error bars are the corresponding 95% confidence intervals. With a shorter startup time or a lower threshold (i.e. lower ARL0), the algorithm is able to detect changes earlier, but it also results in a shorter average change duration the average time interval between two consecutive change points. A shorter average change duration indicates that the algorithm detected more short-term changes. The change detection should be adjusted to fit the purposes of different applications. For example, an intervention program might want to detect changes as early as possible to provide participants with timely feedback, but a study on the effect of a new depression treatment might be willing to trade the responsiveness of change detection for a higher probability that the detected change was persistent. The results presented here can be used as a guideline for Lifestreams users to choose the parameters that best fit their purposes.

4.3.3 Pairwise correlation change detection

Another type of change detection, as described in Section 2.3.3, is to detect changes in the correlation of pairwise features. We applied this technique to all pairwise features of each participant. Figure 4.5 shows a sample result of correlation change detection for participant 016. As with the change detection, the solid vertical line indicates the estimated change point, and is followed by a dashed line indicating the time when the algorithm can detect the change. In the beginning of the study, her daily stress level was strongly correlated with whether she exercised
Even though Lifestreams cannot know the specific events that caused the detected changes, the visualization and automated analyses for change detection can help a study coach narrow in on important timeframes and behaviors where significant changes occur. Automated analyses for change detection can be used as a guideline for Lifestreams users to choose the parameters that best fit their purposes. For example, the responsiveness of change detection and the expected duration of the detected changes; shorter average change duration indicates that the algorithm is adjusted to fit the purposes of different applications. For example, an intervention program might want to detect changes as early as possible to adjust the condition promptly. On the other hand, a shorter startup time or a lower ARL0 (Average Run Length) indicates the estimated change point, and is followed by a dashed line that shows where the change was persistent. The results presented here can be used to help coaches be prepared for important changes and adjust their intervention programs accordingly.

As shown in Figure 7, the change detection algorithm usually detects changes earlier, but it also tends to include temporary changes. With a shorter startup time or a lower ARL0, the algorithm is able to detect changes earlier, but also tends to include temporary changes. The responsiveness of change detection for a higher probability that the detected changes will be useful can be improved by adjusting the two parameters of change detection and the expected duration of the detected changes; the tradeoff can be made by adjusting the two parameters of the detection algorithm.

Interestingly, looking at the overall correlation number of the whole study period would not have been useful in detecting the strong relationship between these two features at the beginning of the study. It is only when the time-series is segmented into two components that the pattern during the first three-fourths of the study emerges. Interestingly, looking at the overall correlation number of the whole study period would not have been useful in detecting the strong relationship between these two features at the beginning of the study. It is only when the time-series is segmented into two components that the pattern during the first three-fourths of the study emerges.

Figure 4.5 shows another example of correlation changes. It shows that, for participant 029, while the overall correlation between daily exercise time and stress level is weak (0.1), the correlations of different subsets of time are much stronger: The correlation changes from -0.58 before the first change point, to 0.71, and then back to -0.45 after the second change point. The first change timeframe is consistent with the beginning of the summer class session 1 of her school when the participant started to spend more time exercising due to weight concern/stress which then tapered off during summer class session 2 when she started to fall back to the minimum routine exercise required by her position.

Figure 4.5: A sample correlation change detection result for a participant. This figure shows how the correlation between two features may vary over time.

with her child (with a correlation of 0.68), but after mid-August, the time when her school starts, the correlation between these two features became 0.07, which is considered as weak or no correlation.

Interestingly, looking at the overall correlation number of the whole study period would not have been useful in detecting the strong relationship between these two features at the beginning of the study. It is only when the time-series is segmented into two components that the pattern during the first three-fourths of the study emerges.

Figure 4.6 shows another example of correlation changes. It shows that, for participant 029, while the overall correlation between daily exercise time and stress level is weak (0.1), the correlations of different subsets of time are much stronger: The correlation changes from -0.58 before the first change point, to 0.71, and then back to -0.45 after the second change point. The first change timeframe is consistent with the beginning of the summer class session 1 of her school when the participant started to spend more time exercising due to weight concern/stress which then tapered off during summer class session 2 when she started to fall back to the minimum routine exercise required by her position.
Figure 4.6: A sample correlation change detection result for the other participant.

After mid-summer session 2, her high stress was reported to be due to class work required for advancing in her position and she spent less time exercising due to class load, hence the negative correlation.

### 4.4 Extensibility of Lifestreams

This section evaluates the extensibility of Lifestreams by demonstrating how Lifestreams can be useful for both smaller preliminary explorations (Family Wellness), and larger more mature studies (Mobilize); they also demonstrate how readily new analyses can be added to Lifestreams and applied to additional data streams such as audio data and phone usage data that were not originally contained in Lifestreams. In this section, we briefly describe the two studies, their unique requirements, how Lifestreams modules were reused or newly developed to fulfill these requirements, and preliminary findings.

#### 4.4.1 Family Wellness Study

Family Wellness is a pilot study aimed to evaluate the feasibility of using mobile technologies for familial behavior studies. To date, 15 families have been recruited
and completed the study. All families included mothers and one child aged 10-14, and about half also had fathers participating. The participants used ohmage to answer four surveys a day for two weeks about their interactions with each other, and individual-level responses on stress and affect. In addition, the participants ran a smartphone-based audio sensing application that collected and uploaded in situ audio features used to classify speech versus non-speech in the environment.

For this study, we reused the existing self-report module and the correlation analysis module, and developed two additional modules including: 1) an audio data extraction module to extract audio features; and 2) an inter-user correlation module to investigate interactions among family members. 82 lines of code were added to implement these two additional modules while 468 lines of code in pre-existing modules were reused for this study. We report our preliminary analysis results below:

**Correlation between the audio data and family interactions**

Due to the exploratory nature of the audio data collection module, only one participant complete audio data was captured and analyzed. For this particular participant, there was significant correlation between speech/non-speech audio features and self-reported family interaction. For example, daily speech time detected by the application is highly correlated with the levels of argument reported with the study child over a day (coefficient = 0.85, p-value < 0.001) and with the levels of argument reported specifically in the evening (coefficient = 0.75, p-value < 0.01). While more data are needed, these preliminary results suggest that there is a potential value in applying audio sensing in such a context, and the researchers are planning a larger scale study.
Discordance among family members subjective feelings

The inter-user correlation module enables researchers to compare the responses from multiple individuals. For some families, the data show that all three family members have similar responses on objective questions, such as how much time they spent together; as indicated by the high correlation between their responses. In these same families, however, the data exhibit much more disagreement, on subjective questions, such as how is the overall condition of the family, as indicated by low or none correlation.

4.4.2 Mobilize: Mobilizing for Innovative Computer Science Teaching and Learning

Mobilize (mobilizingcs.org) is an NSF-funded educational program that brings computational thinking, and data collection and analysis skills into the LAUSD STEM classrooms through the use of participatory sensing technology. During the deployment in 2013, 446 high school students from 21 classes in 8 schools used ohmage to submit surveys and interpret data on three different subjects—Explored Computer Science (ECS), Mathematics, and Science. In addition to using ohmage, some students also ran SystemSens, which is an Android-based system logging application that captures and analyzes participants’ phone usage [FME11]. Lifestreams is used to answer the following two questions: 1) whether the students’ engagement with the study varied across different classes or different subjects, and 2) whether the students attachment to the phone impacts their data collection engagement. The first question is useful in understanding the effectiveness of different pedagogy methods, and the second question will help program managers decide whether to invest in phone access to encourage engagement.

The survey submission counts are used as a metric to measure the students engagement in a class. The one-way ANOVA analysis module (see Section 3.7)
revels that the students’ survey submission counts differed significantly across subjects, classes and teachers. For some teachers with multiple same-subject classes, the results show consistent pattern among their classes. These results indicate that classroom management including individual pedagogy method is a significant factor in student participation.

To answer the second question, we developed a data extraction module to extract phone usage features of the 90 students with the Systemsens data. These features include the interaction time with the phone, the amount of time the phone is power-on, the amount of network usage, the number of apps installed, camera usage events, and etc. A regression module is developed based on lme4 package to estimate the effect of phone usage on the student engagement [BMB14]. To adjust for the effects from different classes and subjects, we introduce two random-effect variables, one for class effects and the other for subject effects. The Class effect variable varies across different classes, but remains constant within the same class, and a similar property applies to the Subject effect variable. Using this model, we found that logarithm of the mean phone interaction time and the amount of network usage both have small but statistically significant positive effects on the survey submission counts (coefficients = 0.082 and 0.045, both p-values < 0.0001). This indicates that the attachment to the phone may have effects on the students engagement with the data collection. In total, 203 lines of code were added to implement the additional modules for this study, and 393 lines of code were reused.

4.5 Limitations and Challenges

Our analyses suffer from incomplete or missing data due to misunderstanding, participants constraints, and technical problems such as phone malfunction and battery consumption issue. Currently, Lifestreams treat missing data in survey responses by excluding them from the analyses (e.g. pairwise deletion in the
correlation analysis) and treat missing sensory data, such as activity, audio, and location data, by setting a cutoff threshold (e.g. any day that has less than 70% of coverage or less than 16.8 hours of data will be ignored). The missing data would cause problems in certain analyses such as the change detection. In the future, statistical methods of data augmentation will be used to infer and impute the missing data.

Furthermore, the phone-based sensing applications may sometimes fail to capture users behaviors. For example, some participants did not carry their phones during exercise, and some left their phones stationary all day. While some of these specific usability problems can be mitigated by using wearable sensors such as Fitbit, and others were an artifact of the participant having a second primary phone, a broader challenge is to create a more engaging mobile user experience and more sophisticated data models.

Some statistical results (e.g. the number of changes detected per dataset) shown in the analysis are sensitive to the parameters. As with most statistical methods, many of the analysis techniques in Lifestreams require iteratively tuning and validation based on the ground truth. In this work, we use the qualitative interview data as the ground truth. However, participants descriptions of their condition might be inaccurate, or it might be hard or inappropriate to elicit certain information from the participants. Expert domain knowledge from behavioral and health science researchers is needed to provide further insight into and help validating the results.

This version of Lifestreams was designed for researchers or highly trained individuals who have familiarity with statistics or at least extensive experience in their domain. The analytic modules are intentionally highly configurable and provide detailed information that allows users to interpret the results. Such amounts of flexibility and information, however, would be overwhelming to users in a real-world clinical setting where time per patient is limited and regulated. Much work
will be needed, both in analysis methods and interface designs, to summarize information and produce results that are more accessible to busy users. More generally, there is significant work to do related to the data collection, validation and generalization of these techniques.

### 4.6 Conclusions

In Chapter 2 we describe one of key contributions of this thesis — Lifestreams, a data analysis software stack for small data behavioral analysis, consisting of four layers: feature extraction and aggregation, feature selection, inference and interactive visualization as well as an initial set of analytical building blocks. Lifestreams facilitates the exploration and evaluation of multi-dimensional personal data streams for potential behavioral indicators.

Using Lifestreams analytical pipeline, we present three sets of analysis results based on our inference methods to help make sense of a real-world behavioral study dataset. These inference methods identified significant behavioral trends and patterns, and their accuracy were qualitatively verified in follow-up interviews conducted with participants. In addition, Lifestreams was found to be useful as a tool for the research coordinator to quickly navigate through the data and provide visual aids to guide the discussion with participants during interview sessions. Lifestreams is designed to be an extensible system that can be generalized and applied to different studies containing varying data streams. We evaluated Lifestreams extensibility by applying it to two additional studies with unique requirements and new types of data streams.

Ultimately, researchers will develop interventions that can greatly improve the personalization and precision of patient-centric care. For example, consider a rehabilitation program for patients with hip surgery. A healthcare provider could deploy a mHealth study in which 50 patients are asked to use phones to passively...
monitor their mobility. During the study, the provider would use a tailored instance of Lifestreams to check the data weekly and identify if patients are having any trouble complying with the study requirements. After a few months, when a large amount of data has been collected and is ready for the provider to analyze, she could use Lifestreams to study the interaction between patients recovery progress and their daily life behaviors; such as working schedules, sleep patterns, stress and mood, each automatically extracted by the Lifestreams feature extraction modules.

Moreover, the provider could use Lifestreams to monitor if significant change occurs in the patients recovery progress and other related behaviors in response to a new treatment plan. Assisted with the visualizations generated by Lifestreams, the provider could use this information to guide the discussion with patients and families, to provide feedback, to identify triggers of the changes, and to help patients with problem solving, goal setting, and monitoring progress towards goals.

Small data provide a wealth of high-resolution data about an individual and their context. However, the promise of using our small data to truly revolutionize patient care and actualize the information collected lays in the ability to translate the information collected into simple and meaningful insights about an individual. Lifestreams attempts to take the first step to achieve that goal.
CHAPTER 5

Small Data System Architecture

5.1 Introduction

So far, we have discussed the application context of using small data for personal behavioral analysis (Chapter 2-4), and have mainly focused on the data analysis challenges for small data systems. However, developing and maintaining a real-world small data system also faces many system challenges that need to be addressed. In this chapter, we discuss some of these challenges, and propose a software architecture to address these shared challenges. The work presented in this chapter was done in collaboration with Faisal Alquaddoomi. We contributed equally to this work.

5.2 System Challenges for Small Data

Small data are “digital traces” of our activities that are stored as we interact with the world around us. These traces are passively produced when we use tools and services that maintain logs: credit cards, grocery receipts, websites and other streaming content services, browsers themselves, etc. They can also be intentionally produced and tracked by sensors and smartphone applications as discussed in the previous chapters. Small data systems usually run as a third-party service that aggregates these diverse data from multiple different platforms on the behalf of the users.
Our vision is to create an ecosystem in which small data systems can be readily developed and deployed atop an infrastructure that standardizes their inter-operation and addresses concerns that are common across applications. Specifically, we focus on the following requirements for small data systems:

**Data Interoperability**

The power of small data systems often comes from their capability of combining individual users’ diverse information that was originally locked in different data silos. These data are from different service providers and usually of different schemas and formats. For example, users’ location data may be extracted from users’ check-in records on Facebook or Foursquare, or their location trajectory collected by apps, such as Moves and Google Map. Each of these services exposes the location data in a different format and through their own proprietary APIs.

A great amount of the development effort is spent on interfacing with a large number of external services, transforming different data formats and maintaining the consistency across different data schemas, and often results in excessively and unnecessarily complex application logic.

**Data Management and Security**

Small data systems differ from many online systems in that most of the data they consume are not generated locally, but from various external sources. Most of the original sources (e.g. Google, Facebook, etc.) persist users’ data in their own databases and individually provide security and access control. It may be wasteful, or even harmful to the users’ security and privacy for a small data system to permanently replicate these data in one place. On the other hand, however, a small data system also wants to be responsive to the users’ requests. Making requests to the data sources on demand will significantly degrade the
responsiveness of the service and hinder the user experiences. How to address this tension between storage efficiency, data security, and usability is another major challenge for the small data application developers.

5.3 Our Contributions

In this thesis, we propose an architecture for small data systems to address the challenges mentioned above. In particular, our contributions are three-fold:

- We develop **Lifestreams DB**, a RDF-based database system that provides a unified interface for querying, combining, and fusing diverse small data streams. We show that Lifestreams DB can support the core data requirements of many small data systems with simple SPARQL queries (See Section 5.5)

- We propose a **Soft-State Design** and a chunk-based management scheme to manage the small data in Lifestreams DB. Our system provides client applications with virtual access to all the user’s small data, but only caches a part of the hot data locally, and reproduces the rest on demand from the data sources. Such a design simultaneously improves the storage efficiency, query performance, and data security. (See Section 5.5 and Section 5.5.2)

- We evaluate the proposed designs with three sample small data applications and real-world user data. We compare Lifestreams DB to Jena TDB, a popular RDF database, and observe a 4.7x improvement in query time while only using 1/10 of the storage space. (See Section 5.6)

In the following, we present the proposed architecture for small data systems. Then, we describe the design and implementation of the most central component in the architecture – Lifestreams DB.
Figure 5.1: Small Data Architecture: the diagram represents the flow of data between data storage units (DSUs), processing units (DPUs), and visualization/apps (DVUs). Lifestreams DB is a DPU container that manages common concerns among DPUs such as caching, security, and providing a unified query interface. Its outputs can be directly consumed by DVUs, or by other DPUs that provide additional data processing capability.

5.4 Proposed Architecture

Our architecture is motivated by three main features of small data:

1. Unlike traditional web/mobile applications, small-data sources are inherently decoupled from their processing and presentation,
2. The data are almost exclusively streaming temporal data, and
3. The data are often considered “sensitive” or private by the end user.

The architecture is also shaped by the observation that the data can generally be reacquired from the data provider at the cost of some increased latency and bandwidth. This has implications for both performance and security: since the data is streaming, the amount of storage space must be bounded or else the applications will eventually fail. Given that the data may be sensitive, storing
excessive amounts of data in the processing and visualization layers increases the
surface area of attack, thus it is advantageous to minimize the amount of stored
data while still providing the desired functionality.

With the above points in mind, the architecture is composed of three layers,
as depicted in Figure 5.1. In keeping with the terminology established by Open
mHealth, data sources are referred to as data storage units (DSUs), while mobile
phone apps, websites, and other consumers of processed data are collectively re-
ferred to as data visualization units (DVUs). In between the DSUs and DVUs
is the processing layer, in which data is acquired from the DSUs, processed to
extract features, compute aggregates, and perform other functions across poten-
tially many incoming data streams, and produce results for the DVUs. Each unit
in this processing layer is referred to as a data processing unit (DPU). A DPU
actively polls for data from its source, which can either be a DSU or another DPU,
and places its processed results in a cache while waiting for its own consumers to
request the data, which can either be other DPUs or DVUs in the visualization
layer.

The division that we describe between storage, processing, and use of the
data is similar to the typical three-tier architecture commonly used when creating
mobile or web applications. These have traditionally been built to support a
single, tightly-coupled application. However, we separate these functionalities into
independent and inter-operable modules in order to encourage the modularity
of small data systems, which is a particular important property for small data
systems due to their needs to interface and process a large variety of data streams.

We explore an architecture in which the DPU layer stores only ephemeral
(aka soft) state in the middle layer\textsuperscript{[Cla88]}, caching only what it needs to access to
ensure the minimum level of functionality, interactivity, and efficiency. If the cache
must be evicted (due to space requirements or a breach), the data is re-requested
from the source.

59
Figure 5.2: Lifestreams DB Architecture: Lifestreams Pipeline consists of a set of DPU modules that acquire and process data from various small data DSUs. Lifestreams Triplestore manages the generated data and makes them available to DVUs through a unified SPARQL endpoint.

5.5 DPU Containers: Lifestreams DB

Lifestreams DB is an important component in our architecture. Positioned between data sources and small data apps, Lifestreams DB is designed to be the narrow waist of the small data ecosystem that provides a unified interface for querying, combining, and fusing diverse small data streams.

Lifestreams DB extracts, transforms, and loads an individual’s digital traces from different sources into one place to enable diverse small data applications. Figure 5.2 illustrates the architecture of Lifestreams DB. On the left is Lifestreams Pipeline, a data processing pipeline that contains a set of reusable DPUs that extract raw data from different small data sources and transform raw data into structured, readily useful information. For example, raw accelerometer and geolocation sensor samples from a mobile app are transformed into structured data that describe the time, location, speed, and distance of each activity episode. These extracted data are loaded into Lifestreams Triplestore, a RDF datastore customized built on top of Jena TDB [ap14], that exposes an integrated view...
to all the diverse RDF data for apps to query. We made two important design
decisions when designing Lifestreams DB: 1) model data using RDF and 2) uti-
elize a soft-state system design. The rationales behind these design decisions are
described in the following.

Model Data Using RDF

Data interoperability is the key for the success of such a data warehouse system.
Raw data extracted from different data silos need to be transformed into a com-
patible form to allow one to learn knowledge from them. In Lifestreams DB, we
utilize RDF to enable data interoperability. Each DPU outputs data in JSON,
and the DPUs at the final stage generate RDF data in JSON-LD format, which
are transformed into RDF triples and made available via a SPARQL query inter-
face enabled by the Triplestore. Two main advantages of using RDF are as follow.
First, it eliminates the need to define data schema unlike, for example, in a SQL
datastore. Data generated by different DPUs are inherently interoperable if the
DPUs follow the same ontologies to model the data. This is an great gain for the
small data ecosystem since it allows DPUs developed by different people to be
plug-n-play without the need to modify the system’s database schema. Also, any
client application developers, given the ontologies, can compose queries to filter,
join, and aggregate various types of data generated by different DPUs without
knowing the implementation detail, such as table and column names, etc.

A Soft-State System Design

A unique characteristic of small data is that an individual’s small data are mostly
persisted in each external service provider’s databases. In many cases, there is no
need, and actually wasteful and harmful to security for Lifestreams DB to replicate
all these data into one place and become a honeypot for identity thieves and other
malicious entities. Thus, we utilize a soft-state design that allows Lifestreams DB to provide the applications with a virtual access to all the data while only caching a small portion of them locally in the system. More specifically, Lifestreams DB maintains an index to all the available data (including those that are not cached), and if a client application requests for the data that are not currently available in the system, Lifestreams DB will rerun the corresponding pipeline to reproduce and reload the data into the database on demand and on-the-fly.

The advantages of such a design are two-fold: first, our soft-state design requires much less storage to serve the requests, and thus allows the system to scale more effortlessly to serve a larger number of users and integrate more diverse information. Further, our design enables elastic storage provision, where a service provider is enabled to provide the service with smaller storage (and low cost), and increases the storage provision only when a better performance is demanded. Second, a soft-state design inherently has better security properties. Since only a small amount of information is cached in the system at any given time, the exposure of any single data breach is limited. Also, the fact that the data can be repopulated into the database on-the-fly enables a data encryption scheme that allows encryption of the sensitive data, and only decrypts and loads them into the database when they are demanded.

These advantages do not come without a price. A soft-state system tends to incur much overhead in indexing, reproducing, and reloading data. In Lifestreams DB, we reduce these overheads by utilizing a chunk-based data management strategy that generates and manages data in chunks and also makes some query optimizations possible. As a result of this design, we improved Lifestreams DB’s query performance by multiple factors using only 10% of the storage space than a hard-state system that stores all the data.

In the following, we first describe our RDF-based data modeling approach and demonstrate its advantages using the SPARQL queries of the small data applica-
Table 5.1: Data Modeling: the ontology defined in *schema.org* is used to allow data interoperability. Some extensions are made to fill the gap in the existing ontology (see the asterisked entries). Of note is that the purchase record is derived from email receipts on an opt-in basis. The phone-based data are uploaded to ohmage [THL15], a participatory sensing platform, that serves as a DSU in the small data ecosystem.

Then, we describes the architecture and techniques we used to realize the above-mentioned chunk-based soft-state system, followed by the performance evaluation section.

### 5.5.1 Data Modeling

When modeling data using RDF, one needs to follow certain *ontology* that formally describes the terminology and relations of concepts in the relevant domain. In order to facilitate the interoperability with external datasets, we tend not to reinvent the wheel, defining our own ontology, but seek for an existing ontology that will suit our requirements. In small data, the concepts we come across most are the various *actions* performed by the users, such as sending emails, making purchases, etc.

Three ontologies that cover this domain have been considered: 1) Hydra, a vocabulary to describe the capability of Web APIs; 2) Activity Streams (AS), an ontology for serializing social activities data; and 3) Schema.org, a collection of ontology for various common domains in the Web and contains a large number
of Action types. We soon found Hydra is ill-suited for us due to its focus on the actions performed by Web APIs rather than human.

The AS and Schema.org have much similarity in syntax but differ significantly in their type system, which leads us to choose schema.org at the end. Unlike AS’s flat action type system, schema.org defines a hierarchical type system that describe different categories and sub-categories of actions. At the root is Action, a generic type that describes the common properties across all the actions. It is then subclassed by more specific types, such as MoveAction, which, in turn, are subclassed by even more specific actions, such as ArriveAction, DepartAction, and TravelAction.

This hierarchical structure enables us to write semantic query to reason across different types of actions within specific categories. For example, an app that encourages better sleep hygiene may analyze users’ before-sleep routines by querying certain categories of actions (e.g. only the subclasses of ExerciseAction) that occurred before the sleep time. Such queries are made possible by the hierarchical type system and RDF’s recursive inference construct: rdfs:subClassOf*.

We extend Schema.org’s ontology when the existing models fall short. For example, StayAction, a direct subclass of Action, is defined to represent the user staying at a place for more than a certain period of time. CallAction, a subclass of InteractAction, is defined to model voice call records. Table 5.1 summarizes eight different kinds of data we have extracted and modeled from four different data sources. Each kind of data generates one or multiple types of actions along with their related objects. For example, a EmailMessage is generated from a email record and is a targeted object of a SendAction.

In the following, we demonstrate how our data modeling approaches, via the Lifestreams DB’s SPARQL query interface, satisfy a wide range of requirements of the small data applications using three sample applications — namely Ora, Pushcart, and Partner.
**Ora** extracts 20 different features about an individual’s daily behavior from location, mobility, and email data. Listing 5.1 shows a snippet of Ora query that computes the geodiameter and the number of emails sent in a day and total number of words written in those mails. The first part of the snippet computes the geodiameter by selecting the maximum distance between any pairs of places at which the user stayed (i.e. the location property of **StayAction**’s). The second part of the query counts the number of **SendAction**’s of which the targeted object is an email, and sums the number of words written in those emails. This example is intended to demonstrate two things. First, it shows that, with Lifestreams DB, how much an application developer can achieve using such a succinct and easy to understand query. Without Lifestreams DB, in our case, only the email data would require at least 789 lines of code to be written and actively maintained for acquiring, parsing, and cleaning the raw email data from the source.

Second, this example demonstrates how heterogeneous data streams (i.e. Location/Mobility and Email) are modeled and queried in a standardized way. Although not shown in this example, one can easily extend the query to join these two data streams to compute, says number of email sent in different places, such as office and home.

**Pushcart** learns the users nutrition consumption from their online grocery shopping records. Listing 5.2 shows a snippet for Pushcart query. This is an example demonstrates the data interoperability by joining an individual’s small data with an external food nutrition database. A RDF dump of the USDA nutrient database is pre-loaded into a separate triplestore [USD09]. The query joins the individuals’ food purchase records with the entries contained in the USDA database based on free-text matching of the product names, and select the amount of carbohydrates and protein contained in each of the purchased items. The database

---

1The line number is computed based on the code we wrote for the email DPUs (in Clojure and Ruby), excluding the comments and the code in an existing email parsing library.
 currently contains about 7500 food items; more comprehensive nutritional information is available at Nutritionix as a paid service [Nut16].

Partner investigates the potential relation between the time two people spend together and the extent of mimicry between their language patterns. It is an example that, in addition to Lifestreams DB, requires a more domain-specific DPU. It relies on Lifestreams DB to compute the amount of time two participants spent together based on the distance between the locations at which they stay (see Listing 5.3) and uses Email Analysis Framework (EAF), a DPU for email language analysis [AKE14], to evaluate language style similarity. It is also an example that shows how Lifestreams DB enables an application to express a query that gains insight from not only one but multiple users’ data with RDF named graphs that refer to different users’ data.

5.5.2 Chunk-Based Data Management

As mentioned, Lifestreams DB is an efficient soft-state system enabled by a chunk-based data management strategy. The basic strategy behind our chunk-based
Listing 5.2: Pushcart Query joins an individual’s food purchase records with the corresponding nutritional information contained in the USDA nutrient database.

approach is as follows: Each DPU generates data in chunks, including the ones at the final stage of the pipeline that output RDF data. Lifestreams Triplestore create an index to these data chunks based on certain features. When a client application submits a query, it will additionally submit a meta-query that selects the chunks that are desired by the application. If there is a chunk that is selected by the meta-query, but is not currently available in the system, Lifestreams DB will request Lifestreams Pipeline to re-run the corresponding DPUs and reproduce the chunk on the fly.

To realize such a design, three major technical issues need to be addressed:

- What features to use for chunk indexing?
- How to reproduce chunks with Lifestreams Pipeline?
- How to decide which chunks to evict when the available space is low?

In the following, we describe our solutions to each of these technical issues, as well as, two query optimization techniques that are made possible by our chunking design. Then, in the performance evaluation section, we put these all together and evaluate the overall system performance.
Listing 5.3: Partner Query computes the co-present time of two users based on the location trace extracted from Moves. Each user’s data is referred to by their named graph.

Listing 5.4: Sample chunk matching query that matches all the chunks containing CommunicationActions (including Send, Call, Receive, and ReplyActions) since 2014-10-01.

5.5.2.1 Chunk Index

- Distinct types of subjects.
- Start time and end time of the aggregate timespan.
- Geo-coordinates of a convex hull that covers all the spatial features in the chunk.

These features are modeled in RDF as well, and can be matched using a query with OWL reasoning capability. For example, Listing 5.4 shows a query that
matches chunks with any type of communication action, including Send, Call, Receive, and ReplyActions, since 2014-10-01.

5.5.2.2 Lifestreams Pipeline: a reproducible data pipeline

The Lifestreams Pipeline consists of two types of DPUs: **Acquirers** acquire raw data from the sources while **Transformers** transform data from one form into another (including generating RDF data at the end of the pipeline). These DPUs are treated as passive functions invoked by the system. Consider a simple pipeline where one Acquirer and one Transformer linked in sequence. In each iteration, the system invokes the Acquirer with a *state variable* that indicates the chunk we want the Acquirer to acquire. After acquiring the corresponding chunk, the Acquirer will return the raw data chunk along with a new state variable that indicates the subsequent chunk to be acquired in the next iteration. The system then invokes the Transformer to transform the raw data chunk, and stores the output along with the state variable that was used at the beginning.

For this to work, we make two assumptions. First, the raw data are permanently persisted in the data sources (i.e. DSUs), and can be re-acquired by the Acquirer as many times as we want. If this is not the case, a *shim* can be implemented to transfer the data to a DSU with these properties (such as Amazon S3 or Google Cloud Storage). Second, the Acquirers and Transformers will generate the same output if the same input is given. All the DPUs we have needed to implement so far meet this assumption.

Unlike some chunk-based systems where the size of chunks is pre-determined, Lifestreams DB allows each Acquirer to decide the chunk sizes according to the characteristics of the APIs it acquires data from. A typical chunk size is daily (e.g. the email data of 2015-05-05) as it is supported by the most data sources we have integrated. However, as the state variable is updated by the Acquirers
themselves, different Acquirer can have state variables in different format and with varying granularity they find suitable (e.g. hours, weeks, or page IDs for paged result sets). This feature is important for small data as a small data system usually need to work with a large variety of external data sources where it has no control to the specification of the APIs. One thing of note is that in some cases, the requested chunks would be unavailable or only partially available at the moment – for instance, the email data for today wouldn’t be fully available until the midnight. In such cases, an Acquirer will return empty or partial data and specify when it would like the system to invoke it again to retry.

5.5.3 Two-Level GDS Chunk Replacement Policy

Our soft-state design essentially make Lifestreams DB a local data cache and, similar to many cache systems, it requires a replacement policy to properly choose chunks to replace when the available space is low. There are two ways to make space in Lifestreams DB: (1) compress the chunk, or (2) evict the chunk entirely.

Compression on average results in 7.2x size reduction and with lower recovery latency. However, it may still be preferable to evict a compressed chunk if it is not expected to be used again in the near future. Given the different characteristics of these space reduction approaches, and the varying size and cost in reproducing different types of data (see Table 5.2), we develop a Two-level Greedy-Dual-Size (Two-Level GDS) replacement algorithm that is both cost- and size-aware and makes appropriate decisions between compression and eviction. The basic Greedy-Dual (GD) algorithm assigns each chunk a cost value $H$. When a replacement needs to be made, the chunk with the lowest $H$ value $H_{min}$ is replaced, and then every chunk reduces their $H$ values by $H_{min}$. If a chunk is accessed, its $H$ value is restored to its initial value. Greedy-Dual-Size (GDS) incorporates the different chunk sizes by assigning $H$ as cost/size of the chunk [C197].
On top of that, our Two-Level GDS algorithm additionally considers the different characteristics of compression and eviction. When a chunk is first inserted into the chunk, its $H$ value is set to $H_{\text{compress}}$, where the cost is the estimated decompression latency, and the size is the estimated size reduction brought by compression. When such a chunk is selected for replacement, it will be compressed and re-inserted into the cache with its $H$ value set to $H_{\text{evict}}$, where the cost is the estimated latency for the Pipeline to reproduce it (measured when it was first produced), and the size is its size in the compressed form. Only when this chunk is selected again for replacement will it be completely evicted from the system.

Similarly, when a chunk is reproduced due to a cache miss, it will be first stored in its compressed form. When it is accessed again, it will have a certain probability to be restored to its decompressed form. In this way, our algorithm uses compression as the default space-reduction method for efficiency, but still removes those compressed chunks that have been unused for long to avoid cache pollution. When there are two compressed chunks used about the same time ago, the one with the lower cost/size ratio will be chosen for replacement.

### 5.5.4 Chunk-Assisted RDF Query Evaluation

The flexibility of RDF is not without its drawback: compared to SQL-based datastores, a RDF datastore tends to have slower query evaluation due mainly to the difficulty of constructing an effective data index [MUA10]. Our chunk-based data management strategy has several desirable side benefits that mitigate this performance problem and make Lifestreams DB feasible for our targeted applications.

First, chunk indexes can be utilized as a multi-column RDF index that allows the query evaluation engine to take a short path by skipping those data that do not belong to the demanded chunks. This optimization shows promising perfor-

\footnote{The default probability for a compressed chunk to be restored is 0.2}
formance improvement in our experiments when the number of demanded chunks is relatively small (see Section 5.6.3) and can be a complement to the existing indexes.

Chunking also enables a more effective query result cache, which caches the result of a query and returns it when the same query is given. In traditional record-based system, any modification to the records can potentially render the cached results invalid [MUA10]. With our chunk-based approach, however, one only needs to track the modifications of those chunks that generate a cached result to ensure it is still valid. This technique is particularly effective in our system, as most chunks won’t change after they have been fully generated.

5.6 Performance Evaluation

In this section, we evaluate the feasibility and performance of our system using Gmail and Moves data. Using Jena TDB as a baseline, we first evaluate our chunk-based management strategy’s overhead in storage and write throughput. Next, we evaluate the system performance in different scenarios and for different data. Finally, we evaluate the overall system performance with a real-world query with a workload simulation based on an assumed application usage. The experiment was conducted on a AWS instance with 8 Intel Xeon E5-2680 processors and 15GB of memory.

5.6.1 Dataset

A dataset of 180 days worth of Gmail and Moves data from three regular users of these services is used to evaluate the system performance. There are in total 360 chunks in the dataset, where each chunk contains a single day’s Gmail or Moves data. Table 5.2 summarizes the characteristic of the two types of data. These varying characteristics must be taken into account to achieve efficient resource
Table 5.2: Gmail and Moves data show varying characteristics in sizes and reproduction time. These differences must be taken into account to achieve efficient resource utilization. For example, while smaller in size, a Gmail chunk requires many more HTTP requests to be issued to the Gmail API and thus has longer reproduction time. A Moves chunk, on the other hand, can be reproduced in a much shorter time, but usually is much larger in size due to the high-frequency tracking points collected when the user was on the move. These differences will result in different performance characteristics as shown in the following subsections.

### 5.6.2 Chunking Overhead

Table [5.3] summarizes the overhead of our chunking approach. In terms of storage, our approach incurs modest overhead for both Moves and Gmail data to store the extracted chunk features. The size of chunk features are comparable across different types of data. More significant overhead, though, is observed in write throughput. This overhead is mainly due to the time spent in extracting chunk features, and is particularly expensive for Moves chunks that have a greater number of triples to be processed. This overhead, however, has not significantly degraded the end-to-end system performance, as it is relatively short and occurs less frequently compared to other operations, such as data queries.

<table>
<thead>
<tr>
<th>Avg. Values of 180 Chunks</th>
<th>Gmail</th>
<th>Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chunk Size (KB)</td>
<td>20.32</td>
<td>392.44</td>
</tr>
<tr>
<td>Compressed Chunk Size (KB)</td>
<td>3.08</td>
<td>54.12</td>
</tr>
<tr>
<td>Required HTTP Requests</td>
<td>14.24</td>
<td>1</td>
</tr>
<tr>
<td>Reproduction Time (msec)</td>
<td>1423.63</td>
<td>182.17</td>
</tr>
</tbody>
</table>
Table 5.3: Chunking incurs negligible overhead in storage, but with sizable overhead in write throughput due to the time spent in extracting chunk features.

<table>
<thead>
<tr>
<th>% Overhead</th>
<th>Gmail</th>
<th>Moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overhead in Storage</td>
<td>4.90%</td>
<td>1.43%</td>
</tr>
<tr>
<td>Overhead in Throughput</td>
<td>15%</td>
<td>23%</td>
</tr>
</tbody>
</table>

Figure 5.3: Query Performance of Different Scenarios: our approach outperforms the Jena TDB by up to 14x when the chunks are readily available. Decompressing is much faster than reproducing a chunk, while decryption adds only negligible overhead. Gmail data requires more time to reproduce since more HTTP requests need to be made. The varying performance in different scenarios evidence the need of an cost- and size-aware chunk replacement policy.

5.6.3 Query Performance

We compare the query performance of our system with our baseline, Jena TDB, based on the following four scenarios:

1. The demanded chunks are readily available.
2. The chunks need to be decompressed before query.
3. The chunks need to be decompressed and decrypted (with 256-bit AES Encryption).
4. The chunks need to be reproduced from the data source.
The results suggest up to 14x performance improvement over Jena TDB for both a simple query and a complex real-world query. The experiment was conducted with all 360 chunks pre-loaded into the triple store. Each data point presented below is an average of 30 runs of the experiment. The error bars in the figures are the 95% confidence interval.

5.6.3.1 Simple Query Performance

We first evaluate the performance with a simple query that counts the number of distinct Action subjects. Figure 5.3a and Figure 5.3b show the experimental results for Gmail and Moves data respectively, where the x-axis is the number of chunks demanded in the query, and the y-axis is the mean query evaluation time. When the demanded chunks are readily available, our system outperforms Jena TDB by up to 14x and 10x for Gmail and Moves respectively. This performance gain is mainly attributed to the chunk-skipping optimization mentioned in Section 5.5.4. For Gmail data, decompressing shows up to 36x better performance than reproducing, and decryption adds only negligible overhead (less than 1.3%). This difference is not that significant for Moves, since Moves data can be reproduced in a relatively shorter time, but incurs larger overhead to be inserted into the triple store in either scenario.

5.6.3.2 Real-World Query Performance

Next, we use a real-world query to demonstrate the system performance in a more realistic setting. A query from one of our small data applications, Ora, is used (See Section 5.5.1 for the description of Ora). It consists of 211 lines of SPARQL script, extracting 20 daily features from Gmail and Moves data, including time leaving/returning home, total exercise time and distance, the number of emails sent, and the number of distinct people contacted, etc. Since this more complex
query requires a larger number of scans to be made over the search space, as shown in Figure 5.4, the performance gain of our chunk-skipping technique becomes more evident (up to 14x improvement over Jena TDB). In addition, due to the longer overall query time, the overhead in decompressing and decrypting them becomes less significant. Reproducing is still the slowest among the four scenarios, but it outperforms Jena TDB by up to 1.8x too.

### 5.6.4 Performance with Simulated Workload

The varying performance for different types of data and scenarios stresses the need for a chunk replacement policy that is able to incorporate these discrepancies. In this subsection, we evaluate the effectiveness of the proposed Two-Level GDS algorithm using a simulated workload. We compare our approach with Least-Recently-Used (LRU) replacement policy, as well as Jena TDB that retains all
the data. The results suggest that our Two-Level GDS algorithm is more effective than LRU with a performance improvement of up to 2.3x; overall, our system outperforms Jena TDB by up to 4.7x using only about 1/10 of storage.

We generate 120 days worth of data for the workload based on an assumed usage pattern of Ora. We only consider the performance of the last 60 days in which the cache space has been saturated. We ran the experiment with an empty triple store, and assume the data is acquired from the data sources on a daily basis. To allow a fair comparison, we modify the traditional LRU in a way that the chunk chosen for replacement will be first compressed and re-inserted into the LRU list. Only if it is chosen again will it be entirely evicted. We refer to this variant of LRU as Two-Level LRU. In addition, for the baseline, Jena TDB, we assume it retains all the 120-day worth of data in the system, which is 50.44MB in size.

Figure 5.5 shows the performance of different approaches with cache sizes varying from 5MB to 20MB. Our Two-Level GDS shows superior performance over Two-Level LRU especially with a smaller cache size. This advantage comes from the fact the our approach takes the cost of different space reduction methods, and the size of each individual chunk into account. For example, our approach tends to evict a Moves chunk for its shorter reproduction time and larger size. On top of that, if we use 0.5MB of the cache space to cache the query results, we see another 2x of performance improvement. Overall, our approach achieves up to 4.7x performance improvement over Jena TDB, using only about 1/10 of storage. Such a performance improvement is important for small data services to be provided effectively and affordably.
Figure 5.5: Query Performance with Simulated Workload: our Two-Level GDS approach shows superior performance over LRU, and outperforms Jena TDB that retains all 50.44MB of data, by up to 4.7x using only about 1/10 of storage.

### 5.7 Conclusions

In this chapter, we present an architecture to support small data applications that decouples the data sources from the processing and visualization layers, and accounts for the unique challenges presented by contending with sensitive streaming spatio-temporal data from multiple providers. We describe our implementation of this architecture, Lifestreams DB, and several candidate applications built on top of it.

Lifestreams DB includes several improvements over existing RDF datastores in terms of storage requirements and query latency, which are likely attributable to the constraints of our domain (i.e. streaming spatio-temporal data which can be reproduced at a cost in latency from an external source.) The application of chunking to the datastore, and a cache eviction policy that leverages both the cost of reproduction/compression and the size of the data, is demonstrated to improve query latency for both a few candidate queries and in a simulated experiment.
Integrating with RDF Projects

Going forward, we hope to extend both the small data architecture and Lifestreams DB specifically along several axes. First, the fact that Lifestreams DB is an RDF datastore provides an opportunity to integrate with emerging RDF projects, such as CrossCloud. We envision Lifestreams DB as fulfilling the role of a Cross-Cloud publisher, allowing processed data from external, likely non-RDF sources to be included as components in CrossCloud-enabled applications. More generally, the SPARQL query interface that Lifestreams DB exposes allows the processed data streams to be easily joined against other SPARQL query engines, and Lifestreams DB’s use of schema.org’s ontology eases its compatibility with datastores that use the same ontology. We also hope to integrate with more distributed/user-controlled forms of identity verification such as WebID [SIS11], which would allow us to decouple from the identities that data providers use, e.g. Google Identity or Facebook Login.

Improving Security and Privacy

While this work proposes a soft-state architecture to ameliorate the impact of a breach, there is still much work to be done in secure data storage and distribution so that breaches are diminished or, preferably, eliminated in the first place. On a related note, there are many improvements that can be made to ensure that the processed data does not compromise the raw data source, and to selectively control who can consume processed data in the case that it is sensitive. We intend to investigate an approach to sharing data that we refer to as Functional Information Sharing (FIS), in which security assurances are integrated deeply into compliant systems. We hope to achieve this through hardware tools such as trusted platform modules (TPMs), identity verification between systems through the use of public-key authentication, and fine-grained
security policies that detail how the data may be used with revocation mechanisms in place when these agreements are violated.

Supporting the Small Data Ecosystem

In general, we hope to **foster the growth of the small data ecosystem** by continuing to implement small data systems, continuing research into effective and secure means to process and share small data, and providing reference implementations/infrastructure components to support development. Lifestreams DB will hopefully be one of many DPU-supporting components that will be available to developers looking to implement their own small data systems.
CHAPTER 6

Small Data for Personalizing Online Experiences

6.1 Introduction

In this chapter, we turn our attention to the second application context — using small data for personalizing online experiences. We introduce the notion of immersive recommendations and discuss the technical challenges we need to resolve in order to realize this new personalization paradigm.

6.2 Immersive Recommendations

The rise of the web, social media, e-commerce, and mobile communications creates almost continuous digital traces about us. While the proliferation of these traces accelerates, most of our data are kept separately in each service provider’s data silos. Little exploration has been done to utilize individuals’ small data across services to improve users’ experience online and tailor it based on individuals’ broad preferences.

In this application context, we aim to use small data to create a future in which hyper-personalized content, entertainment, and tools will let individuals benefit from the data they generate more directly, selectively, and transparently. We propose a new user-centric recommendation model, called Immersive Recommendations, that utilizes individual users’ cross-platform small data to make recommendations, satisfying users’ diverse needs and interests.
Immersive recommendations is a user-centric recommendation model, where a recommendation system, through users themselves, accesses a wide range of users’ small data kept across different platforms and creates a comprehensive view of the users’ preferences that single service provider may not be able to capture. This is in contrast to the traditional provider-centric model, such as the recommendations we see on Amazon or Netflix, where only the in-channel traces on those platforms are used to make recommendations.

To illustrate this idea, we initially target the immersive recommendation applications for news and local meetup events due to their utility and societal importance [JK10, Put95, FK01]. We develop systems and novel algorithms to leverage users’ digital traces, including social media records and personal email communications, to make personalized news and local meetup event recommendations. We evaluate the feasibility and efficacy of the proposed approach through 1) a large-scale offline evaluation with real-world user data, and 2) a 33-person interactive user study with users’ direct feedback. In the both cases, immersive recommendations show significant improvements in recommendation accuracy compared to the state-of-the-art recommendation approaches that use only the in-channel data. The results suggest promising benefits of immersive recommendations in improving recommendation accuracy for new users (i.e. cold start) and existing users.

Furthermore, we develop a real-world web application, called Newsfie.org, to publicly demonstrate the practicability of the proposed recommendation model, and incorporate a wider range of personal data sources, including watch history on Youtube, and team communications on Slack. The application will support future user studies to understand more qualitative aspects of the recommendation performance and the extent to which these additional data streams further improve performance and user experience.
6.3 Challenges for Immersive Recommendations

Many technical challenges need to be resolved to unlock the potential of immersive recommendations. In particular, in this thesis, we focus on the following areas:

Profiling Unstructured Multi-Context Small Data

Most prior work in recommendation systems only considered simple demographic information about users. In contrast, immersive recommendations utilize unstructured, dynamic, and highly-dimensional digital traces, such as social media records and personal communication traces generated by users, to enable highly-personalized recommendations. The profiling of these multi-channel data pose significant challenges. In particular, these traces are inherently riddled with noise and contextual biases introduced by the context in which they were generated. How to denoise the small data and turn them into useful user profiles that can be fed into a recommendation system is a major challenge for immersive recommendations.

Making Recommendations with Immersive User Profiles

Constructing a practical immersive recommendation system requires novel recommendation models to combine a wide range of information, including 1) the immersive user profiles extracted from users’ small data, 2) users’ feedback to the items, such as ratings and click-through, and 3) item information, such as item descriptions or item contents. How to fuse these different threads of information and make personalized recommendations for users even when they just start using the system, and rapidly fine-tune the recommendations when more feedback from the users are available are important technical challenges we will address in immersive recommendations.
Figure 6.1: An overview of immersive recommendation. In the profiling phase, Context-Aware-LDA (CA-LDA) is proposed to systematically profile users’ digital traces from different contexts and create user profiles. In the recommendation phase, a hybrid collaborative filtering algorithm is proposed to fuse the user profiles, the item profiles, and the existing ratings to predict the ratings that are still unknown.

6.4 Our Solutions and Contributions

We divide the recommendation process into two phases and propose the technical solutions to each of the challenges mentioned above. As illustrated in Figure 6.1

In the User Profiling Phase, we develop Context-Aware LDA to infer users interests from digital traces of different contexts and create a user profile that has strong predictive power to the kinds of items the user will be interested in. In the Recommendation Phase, we propose a hybrid collaborative filtering
algorithm, called **Collaborative User-item Regression**, that carefully fuses the user profiles, user ratings, and item information to predict users’ preferences for each given item.

In sum, our contributions to this application context are as follow:

- **•** We proposed **Immersive Recommendations**, a new user-centric recommendation model that leverages users’ diverse personal digital traces to make recommendations on the user’s behalf. To our knowledge, this is the first work to study personal, cross-platform, news and local-event recommendations based on individual-user’s multi-channel digital traces.

- **•** We proposed a novel profiling algorithm — Context-Aware LDA, that simultaneously infers users’ interests from multi-context small data while suppressing the noise introduced by different contexts. (See Chapter 7)

- **•** We proposed a novel recommendation model — Collaborative User-Item Regression, that carefully fuses the user/item profiles and rating information to achieve beyond state-of-the-art recommendation accuracy, and can make recommendations for new users and rapidly fine-tune the recommendations according to users’ feedback. (See Chapter 8)

- **•** We conducted a large-scale offline evaluation, a small user study, and the real-world service deployment to explore the feasibility, efficacy, and practicability of this new recommendation model for two key application domains. The results suggest promising benefits of leveraging users’ digital traces to improve future recommender systems and, at the same time, suggests that further research is needed to refine the techniques through real world experiments in order to realize the full potential of immersive recommendations. (See Section 7.6 Section 8.2.3 and Section 8.3)
6.5 Related Work

Recommendation systems have been some of the most successful personalization technologies to date—as evidenced by recommendations for music in Pandora, Movies on Netflix, consumer goods in Amazon [LSY03], and articles in Wikipedia [CFT07]. Today, these systems rely heavily on users’ application-specific (in-channel) histories. Our premise is that tapping into more diverse small data sources (e.g. daily travel patterns, online purchases from gaming to dining, focus of interest expressed in personal communication, etc.), can bring about immersive recommender systems that address a richer and more nuanced model of the user’s interests and preferences, and make the models more intelligible by end-users.

In this section, we review the related work regarding general concepts in recommendations systems, collaborative filtering, hybrid recommendation systems, and cross-domain recommendations and topic modeling.

6.5.1 Recommendation Systems

There are two classes of entities in a recommendation system: users and items. Users have preferences for certain items, and the goal of a recommendation system is to infer these preferences from the data. A common representation of the data is user-item matrix $R$ (also known as utility matrix), in which the value of $r_{ij}$ denotes the preference of user $i$ to item $j$. The user preferences can be expressed either directly, such as by ratings, or indirectly using binary values indicating whether the user has clicked, viewed, or purchased the items. Note that the known preferences of users to items are usually very limited, which makes the matrix $R$ rather sparse, and we use $r_{ij} = ?$ to represent unknown preferences. Generally, the goal of a recommendation system is to predict these unknown preferences in the user-item matrix, and make recommendations accordingly.

Most recommendation algorithms need the user-item matrix to be populated
to some extent in order to achieve a reasonable accuracy. In online services, there are two general sources of users’ preferences:

Explicit Feedback

We can ask users to explicitly rate items. The rating scale can be a 1-5 likert scale [Lik32], or simple thumb-up and thumb-down. This approach is used by e-commerce and video recommendation systems (e.g. Amazon, Netflix, and Youtube), and some news sites. However, this approach is limited in its effectiveness as many users are unwilling to provide responses, and when users are willing to provide responses, they could be biased in certain ways. For example, Kostakos found that one-off voters tend to vote on popular items, while domain experts mostly vote for obscure, low-rated items [Kos09].

Implicit Feedback

Alternatively, we can implicitly infer the users’ preferences from their behavior. For example, if a user buys a product at Amazon, watches a video clip on YouTube, or ”Likes” a post on Facebook, we can assume the user is interested in this item and set $r_{ij} = 1$. In this thesis, we mainly consider implicit feedback for its wide availability and better immunity to bias.

Note that an implicit rating system really has only one class of values for $r_{ij}$ (i.e. $r_{ij}$ are all positive), and therefore making recommendations with implicit feedback is sometimes referred to as one-class recommendation problem [PZC08]. The unique characteristic of this kind of problems is that, for the items that the user did not interact with (i.e. $r_{ij} = ?$), we can not say for sure if it is a negative response or it is just because the user was not aware of the item [HKV08]. Several techniques have been developed to deal with this uncertainty, and we will discuss some of them later in this section.
6.5.2 Collaborative Filtering

Over the last two decades, the most successful recommendation technique is collaborative filtering (CF). CF is based on the assumption that the users who have similar interests in the past will share the same interests in the future. The process of identifying similar users and filtering the items based on those users’ ratings is called collaborative filtering. In the following we briefly introduce two flavors of CF approaches: memory-based CF and model-based CF.

6.5.2.1 Memory-Based Collaborative Filtering

Given a user-item pair \((i, j)\), a typical memory-based CF approach predicts user \(i\)’s rating to item \(j\) by aggregating the ratings the item \(j\) received from other users who have a history of agreeing with user \(i\) (i.e. they rated other items similarly) \cite{RIS94}. The affinity of two users’ ratings is characterized by a predefined similarity metric, such as Pearson correlation, Jaccard index, or the cosine similarity, based on the users’ rating vectors, each of which contains the previous ratings the user has made. Usually only the ratings of the top \(K\) nearest neighbors (i.e. the top \(K\) users whose ratings are the most similar to user \(i\)) are used to make prediction. Specifically, the prediction is made as follows:

\[
\hat{r}_{ij} = \frac{1}{C} \sum_{k \in Z_i} \text{sim}(i, k) R_{kj}
\]

where \(Z_i\) is the set of \(K\) nearest neighbors of user \(i\), \(C\) is a normalizing constant, and \(\text{sim}(i, k)\) denotes the similarity between user \(i\) and user \(k\), measured by the predefined similarity metric.
6.5.2.2 Model-Based Collaborative Filtering

Model-based CF is another flavor of CF, in which we train a predefined rating prediction model with the user-item matrix, and use the trained model to predict the unknown ratings. Matrix factorization has been a dominant technique for model-based CF. It turns the rating prediction problem into the matrix completion problem. With a trained matrix factorization model, the prediction is made by recovering the missing entries in the user-item matrix $R$. We introduce several matrix factorization based recommendation models in the following.

**Incomplete SVD**

Incomplete SVD is one of the most used model for matrix factorization, and was made popular by its success in Netflix Prize. Given $n$ users, $m$ items, and user-item matrix $R \in \mathbb{R}^{n \times m}$, Assuming $R$ is low-rank, we learn:

$$\hat{R} = \hat{U}^T \hat{V}$$

where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{m \times k}$ ($k$ is substantially smaller than $n$ or $m$), and $A$ is the set of ratings we have observed. The complete matrix is estimated by $\hat{R} = \hat{U}^T \hat{V}$, and can be used to estimate the rating $r_{ij}$ of the user-item pairs $(i, j)$ that are not in the training set, i.e. $(i, j) \notin A$.

Another way to interpret the matrix factorization model is to break down the matrix $U$ and matrix $V$ into vectors for users and items, where each user is represented by a latent user vector $\mathbf{u}_i \in \mathbb{R}^k$ (i.e. row $i$ in matrix $U$), and each item is represented by a latent item vector $\mathbf{v}_j \in \mathbb{R}^k$ (i.e. row $j$ in matrix $V$). The rating $r_{ij}$ of user-item pair $(i, j)$ is then characterized by the inner product between their latent vectors, i.e.
\[ r_{ij} = \mathbf{u}_i^T \mathbf{v}_j. \]

Note that we can additionally add bias and regularization terms to the model. For example, The L2-norm regularization is usually applied to the user vectors and items vectors. We can also impose non-negativity constraints to the user and item latent vectors to make a Non-Negative Matrix Factorization model, which tends to produce interpretable latent vectors and works better when ratings are much sparser [LSL12]. Please see [KBV09] for a comprehensive review of various matrix factorization techniques.

**Probabilistic Matrix Factorization**

The matrix factorization model can be generalized as a probabilistic model [MS07]. In probabilistic matrix factorization (PMF), we assume the following generative process for ratings,

1. For each user \( i \), draw user latent vector \( \mathbf{u}_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K) \)
2. For each item \( j \), draw item latent vector \( \mathbf{v}_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K) \)
3. For each user-item pair \((i, j)\), draw the rating

\[
    r_{ij} \sim \mathcal{N}(\mathbf{u}_i^T \mathbf{v}_j, c_{ij}^{-1}).
\]

The \( \lambda_u \) and \( \lambda_v \) serve as regularization terms that regularize the L2-norm for user vector \( \mathbf{u}_i \) and item vector \( \mathbf{v}_j \) respectively; the precision parameter \( c_{ij} \) serves as a confidence parameter for rating \( r_{ij} \). If \( c_{ij} \) is large, we trust \( r_{ij} \) more. The precision parameter are useful when dealing with the uncertainty issue in the implicit feedback, which we will discuss further in Section 8.2.
6.5.3 Hybrid Recommendation Approaches

Beyond CF models, a recent trend in recommendation systems is to incorporate information beyond the user-item matrix, such as item contents or user profiles, into the recommendation models \cite{SLHT14}. Such approaches are generally categorized as hybrid recommendation approaches for them combining both CF-based approaches and the content-based approaches. The model we propose for immersive recommendations also belongs to this category. A hybrid recommendation approach may produce better recommendation accuracy and have more interpretable recommendations results. In the following, we introduce one of such approaches, Collaborative Topic Regression, which our model is built on top of.

Collaborative Topic Regression

Collaborative Topic Regression (CTR) is an extension to the above-mentioned PMF model, specifically designed to improve the scientific article recommendations \cite{WB11}. It is based on a widely-used topic model, called Latent Dirichlet Allocation (LDA), that extracted the topical features from the articles. We defer the introduction to LDA to Section \ref{sec:lda} and focus on the recommendation part of the model here.

Denoting topic proportions of each article \( j \) learned by LDA as \( \mathbf{v}_j \in \mathbb{R}^k \), CTR additionally introduces a latent offset vector \( \epsilon_j \in \mathbb{R}^k \) that offsets the topic proportions \( \mathbf{v}_j \) and predicts the ratings by:

\[
r_{ij} = \mathbf{u}_i^T (\mathbf{v}_j + \epsilon_j).
\]

For each user \( i \), their user vector \( \mathbf{u}_i \) will be biased towards the topics they tend to be interested in (learned from the articles they gave high ratings to). For each item \( j \), the offset variable \( \epsilon_j \) captures a specific phenomenon that this article
tends to attract users who like certain topics while the article’s item profile $v_j$ does not say this article is about those topics. In other words, the offset variables are designed to capture the hidden characteristics of the items that we did not learn using LDA, but are manifested in the users’ ratings. How much of the prediction relies on the topics learned by LDA (i.e. $v_j$) and how much it relies on users’ ratings (i.e. $\epsilon_j$) will be determined by the number of users who have rated the article $j$ and a regularization parameter that regularizes the scale of $\epsilon_j$. Specifically, CTR assumes the following generative process:

1. For each user $i$, draw user latent vector $u_i \sim \mathcal{N}(0, \lambda_{u}^{-1}I_K)$
2. For each item $j$, draw item offset vector $\epsilon_j \sim \mathcal{N}(0, \lambda_{v}^{-1}I_K)$
3. For each user-item pair $(i, j)$, draw the rating

$$r_{ij} \sim \mathcal{N}(u_i^T(v_j + \epsilon_j), c_{ij}^{-1}).$$ (6.2)

The $\lambda_v$ parameter controls the L2-norm of $\epsilon_j$. A larger $\lambda_v$ results in $\epsilon_j$ with a smaller L2-norm, which, in turn, makes the prediction of $r_{ij}$ rely more on the item content (i.e. $v_j$) and less on the users’ ratings.

**Other Hybrid Recommendation Approaches**

In addition to CTR, a great amount of prior work has been devoted to incorporating other types of information into the recommendation models, including, social network graph [MYL08, YLS11], user contributed data, such as comments [BDB15] or images [ZKS15, LWY10, YHE15], and user attributes, such as gender and age [AC09, PC09] (see [SLH14] for a comprehensive survey).

However, little prior work considered the recommendation problems from the angle of the individual users, who have access to almost continuous digital traces,
spanning professional and personal communication and activities. Immersive recommendation approaches the recommendation problems from this user-centric perspective and empowers individuals to use their own diverse data to improve the quality of the recommendations they receive.

6.5.4 Cross-Platform Recommendations

Immersive recommendation is a generalization of the prior work on cross-platform recommendation, where user data in one platform is used to improve recommendations on another platform. For example, prior work used social media records to recommend Pinterest boards [YLL15], Youtube videos [YSX14], and ebooks [SSB14], or aggregated the user profile across different platforms to streamline the on-boarding process on a new platform [AHH10]. However, whereas most prior studies focused on using specific data sources to improve the cold-start recommendations for a specific application, in immersive recommendation, we developed techniques that are able to simultaneously profile multi-context data and improve recommendations in multiple applications beyond the cold-start phase and throughout a user’s lifetime as the interests change.

6.5.5 Other Related Models

Our user profiling algorithm is related to the previous models for comparative text mining [ZVY04, PG09], but we additionally introduce a unique background topic [CS07] for each corpus that captures the specific background noise of different contexts to improve the profile accuracy. Our recommendation model is an extension to Wang and Blei’s CTM [Ble12] mentioned earlier and Agarwal and Chen’s fLDA models [AC10] and belongs to the general framework of the regression-based latent factor [AC09]. However, most prior work in this line of research only considered categorical user attributes, such as gender or age [PC09], and thus cannot
be directly applied to immersive recommendations, where dense high-dimensional user-generated data is available. In contrast, the proposed collaborative user-item regression model is able to utilize rich user data and significantly improve the recommendation performance.

6.6 Conclusions

In this chapter, we describe the second application context we consider in this thesis — using small data for personalizing online experiences. We propose a new user-centric recommendation paradigm, called Immersive Recommendations, that aims to combine users’ diverse small data traces to enable a highly-personalization online experience. Unlike existing work in recommendation systems, to realize the potential of immersive recommendations, we need effective profiling algorithms to learn users’ interests from unstructured and noisy small data, and novel recommendation models to combine the immersive profile with other information. In the next two chapters, we will describe our solutions to these technical challenges and evaluate the solutions’ efficacy with real-world data.
CHAPTER 7
Profiling Multi-Context Small Data

7.1 Introduction

In this chapter, we describe our strategy to profile users’ interests from their small data of textual forms, such as social media records and personal communications, e.g. emails and text messages. We demonstrate contextual-noise issues of using traditional topic model to profile users’ multi-context digital traces, and propose a novel profiling algorithm, called Context-Aware LDA (CA-LDA), to address the problems. We evaluate the efficacy of CA-LDA with a large number of users’ Twitter data and compare its accuracy with other prior approaches in predicting the users’ preferences for news and local meetup events.

7.2 Problem Formulation

We formulate the multi-context user profiling problem as follows:

Given a user $i$, the input of the user profiling problem is a set of digital trace instances denoted as:

$$N_i = \{n_{i,m} = (c_{i,m}, E_{i,m}), m = 1, ..., M\},$$

where $n_{i,m}$ represents each instance in the user $i$’s digital traces. $n_{i,m}$ can be an email thread, a set of relevant tweets or a series of Facebook posts. $c_{i,m} \in C$ denotes the context in which the instance was generated, such as Email, Twitter,
or Facebook, and $E_{i,m}$ is the content of the instance. In this work, we focus on the textual content of the digital traces for its availability in both social and personal context. Therefore $E_{i,m}$ can represent the text in an email thread or in the social media records. The goal of the user profiling is to create a function that transforms $N_i$ into a feature vector $u_i$ that characterizes user $i$’s interests. Similarly, each item $j$ for recommendation is also associated with an item profile $v_j$ derived from its contents.

The prediction power of the proposed profiling approach is measured by how accurately we can predict users’ preferences for news and local meetup event based on users’ profile $u_i$ and item profile $v_j$. Please see Section 7.6 for evaluation details.

### 7.3 Profiling Small Data with Topic Models

Our user profiling strategy builds on the premise that a person tends to be interested in the topics she is engaged with in her daily life and would mention or consume such information regularly [HR06, Hof10]. We use topic modeling techniques to characterize this engagement from the user’s digital traces and discover the user’s interests in the domain we want to make recommendations for (e.g. news and meetup events).

Given a collection of documents, topic modeling reveals a human-interpretable relatively-low-dimensional representation for documents — namely the topic distribution of each document [Ble12]. Topic modeling was originally designed for corpus exploration, and has been extended to other applications, including profiling item contents in a recommender system [Ble12] and categorizing users’ social media feeds [BZG14].

In contrast to prior topic modeling work that focuses on text from a single corpus, immersive recommendations are based on textual data that users generate in multiple different contexts in different daily activities. Directly applying
traditional single-corpus techniques causes the model to incorporate much noise and bias from these contexts and produces rather poor recommendation performance (See Section 7.6). As such, we propose **Context-Aware LDA** (CA-LDA), a topic modeling algorithm that is able to perform cross-corpus topic modeling and simultaneously model text data from multiple small data contexts.

Such a topic-model-based approach has three major advantages for immersive recommendation: First, it is fully unsupervised and can be easily extended to analyze user-data from a new context, or new types of items, without costly human-labeling process or hand-tuning. Second, in contrast to other representation learning algorithms, such as *doc2vec* or matrix factorization [LM14, ZCH15], the user profile based on topic modeling is semantically-meaningful. This is an important feature to allow for recommendations that are more transparent and trustworthy to the user [SS02, Ble12]. Third, this approach does not use personal data to train the model. The trained model can then perform inference entirely on the client-side to mitigate privacy concerns of immersive recommendation (as demonstrated in [moz14]).

Next, we provide a short introduction to LDA, a widely-used topic modeling algorithm, and discuss its drawbacks when applied to immersive recommendations.

### 7.4 Latent Dirichlet Allocation (LDA)

A widely-used topic modeling algorithm is Latent Dirichlet Allocation (LDA). Given a corpus of $D$ documents and a vocabulary of size $V$, LDA assumes there are $K$ topics in the corpus, each of which is characterized by a word distribution $\phi_k \sim \text{Dirichlet}(\beta)$. LDA assumes the following generative process for each document $d$:

1. Draw a topic distribution $\theta_d \sim \text{Dirichlet}(\alpha_{1..K})$
2. For each word $n$ in document $d$,

   (a) Draw a topic assignment $z_{d,n} \sim \text{Mult} (\theta_d)$

   (b) Draw a word $w_{d,n} \sim \text{Mult} (\phi_{z_{d,n}})$

With a sparse topic distribution $\theta_d$ (controlled by $\alpha_{1...K}$), words that co-occur in many documents tend to be assigned the same topic. The word distribution of each topic reveals different themes underlying a corpus while the topic distribution $\theta_d$ of a document characterizes the themes the document is associated with. From an embedding point of view, $\theta_d$ is document $d$’s projection in a low-dimensional non-negative topical embedding [ABE14]. Two documents associated with the similar themes will be projected to points that are proximate to one another in this embedding.

7.4.1 Basic LDA Profiler

Denoting a user’s digital trace collection as $N_i$, where $i$ is user index, one straightforward way to model user $i$’s digital traces, referred to as Basic LDA, is as follows:

1. Train an LDA model with the item corpus (e.g. the news articles or meetup group descriptions)

2. Use the trained model to infer the topic distributions for each instance $n_{i,m} \in N_i$ denoted as $\theta_{i,m}$

3. Define the user profile $u_i$ as the weighted sum of the topic distributions of $\theta_{i,m}$ based on each instance’s potential relevance to the user’s interests. $^1$

This approach defines a profile $u_i$ to be the center of mass of $n_{i,m}$’s projections in the topical embedding created with the item corpus, and an item whose topic distribution $\theta_d$ is closer to $u_i$ is supposed to be associated with the themes that are

$^1$See Section [7.6.2] for more on the weighting scheme.
frequently mentioned in the user’s digital traces. While intuitive, this approach has two major problems:

**Insufficient coverage for users’ language usage**

The model trained with the item corpus is not able to cover diverse language usage in the users’ digital traces. This is particularly the case when the item corpus is relatively small. For example, an LDA model trained with meetup descriptions from Meetup.com shows rather poor performance in profiling users’ interests (see Section 7.6.2). This is not only due to a smaller number of items available in the meetup corpus, but also because the meetup descriptions are much shorter and narrower in topic and vocabulary.

**Context-specific noise**

The other issue is that the user profiles generated by Basic LDA are biased towards the context-specific topics that prevail in a certain context but do not represent users’ interests. For example, in email, people tend to use words, “discuss”, “meet”, etc., that are often classified as office- or work-related themes. It is usually the case that a person mentions these terms not because she is interested in the office-related topics, but because email is usually used in work-related contexts. In other words, the occurrences of these words are largely independent of the user’s interests and should be excluded from the user’s profile. Such context-specific noise exists in many kinds of traces. For example, on Twitter, people tend to have “share”, “love”, “video”, etc. social-oriented terms, but they are rarely associated with users’ real interests. When we directly use the topic distributions learned by LDA, this noise overwhelms the real interests of the user.
7.5 Context-Aware LDA Profiler

To address the above-mentioned problems, we propose Context-Aware LDA (CA-LDA). This model originates from the techniques used in the comparative text mining [ZVY04, PG09], where multiple corpora are co-trained in a single model to reveal the commonalities and discrepancies between them. In CA-LDA, we co-train multiple item corpora (news and meetup descriptions) along with the digital trace corpora (Twitter, Facebook, and email). As illustrated Figure 7.1, CA-LDA assumes that all the corpora share a superset of salient topics, i.e. the
topics that reflect users’ interests, and each corpus individually has its own unique background topic that is associated with the context-dependant noise.

The intuition here is that, given a large number of trace instances from a certain context, the context-dependant noise should prevail in these instances regardless of their main topics. To model this intuition, we assume each document to be a mixture of salient topics and the background topic, and the background words are sampled directly from the word distribution of the context’s background topic independent of the document’s topic distribution \( \theta_d \). Specifically, for each corpus \( c \) and the documents in it, the Context-Aware LDA assumes the following generative process:

1. Draw a word distribution \( \phi_c \sim \text{Dirichlet}(\beta_c) \) for the background topic

2. For each document \( d \) in corpus \( c \),
   
   (a) Draw salient words proportion \( \lambda_d \sim \text{Beta}(\gamma_\alpha, \gamma_\beta) \)
   
   (b) Draw topic distribution \( \theta_d \sim \text{Dirichlet}(\alpha_{1...K}) \)
   
   (c) For each word \( n \), draw \( x_{d,n} \sim \text{Bin}(\lambda_d) \)
      
      i. If \( x_{d,n} = 1 \) (a salient word)
         
         A. Draw a topic assignment \( z_{d,n} \sim \text{Mult}(\theta_d) \)
         
         B. Draw a word \( w_{d,n} \sim \text{Mult}(\phi_{z_{d,n}}) \)
      
      ii. If \( x_{d,n} = 0 \) (a background word)
         
         A. Draw a word \( w_{d,n} \sim \text{Mult}(\phi_c) \)

When \( x_{d,n} = 1 \), the generation of the word is identical to LDA except that the word distributions of salient topics are shared across different corpora. When \( x_{d,n} = 0 \), the generation of the word is independent from the document’s topic distribution \( \theta_d \) and directly drawn from the corpus-specific background topic. This design makes the terms that prevail in a particular context more likely to be
Table 7.1: Frequent background terms learned by Context-Aware LDA. The words were stemmed before included in the model.

<table>
<thead>
<tr>
<th>Context</th>
<th>Background Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>pleas, offic, schedul, convers, fax, cellular</td>
</tr>
<tr>
<td>Twitter</td>
<td>awesom, share, tweet, post, video, love</td>
</tr>
<tr>
<td>Facebook</td>
<td>love, night, happi, tomorrow, final, tonight</td>
</tr>
<tr>
<td>Medium.com</td>
<td>happen, idea, actual, experi, hard, reason</td>
</tr>
<tr>
<td>Meetup.com</td>
<td>social, event, network, singl, profession, join</td>
</tr>
</tbody>
</table>

assigned to the background topic and, at the same time, prevents them from diluting the salient topic distribution $\theta_d$. On the other hand, co-training multiple corpora mixes topics in different corpora and allows the smaller corpus (i.e. the meetup descriptions in our case) to benefit from the diverse topics and vocabulary in the larger corpus (i.e. the news articles in our case). The inclusion of a large news article corpus also increases the robustness of the word distribution $\phi_k$ due to the longer documents and more diverse word choices contained in the news articles.

Similar to the Basic LDA profiler, given a user’s digital traces $N$, the trained Context-Aware model is used to infer the topic distribution for each instance $n_{i,m}$. The user profile $u_i$ is again defined as the weighted sum over the instances’ topic distributions $\theta_{i,m}$

$\theta_{i,m}$, but now the $\theta_{i,m}$ are only associated with the salient topics and separated from the context-specific terms. As shown in our experiments, this leads to a user profile $u_i$ that is much more focused on the user’s real interests and has a stronger predictive power.

Training CA-LDA with Public Datasets

We sampled digital traces from the datasets that exist in the public domain to train the model. For example, the Enron email dataset [KY04] and several public

---

2The proper weighting scheme is learned from the existing users’ data. See Section 7.6.2 for details.
mailing lists were used to construct the email corpus that covers diverse topics; 100,000 users’ public Twitter data and 1,200 users’ public Facebook posts were used to construct the Twitter and Facebook corpora. The model is implemented based on Mallet’s parallel LDA implementation; the source code and the trained model are released at [Hsi15]. Some of the learned background terms are listed in Table 7.1.

In addition to the user profile, we also use CA-LDA to generate item profile $v_j$ from the item contents, which will be used in the recommendation as well. Compared to other ad-hoc noise suppressing approaches, such as hand-crafting a list of background terms, or carefully tuning the $tf-idf$ thresholds for different corpora, CA-LDA resolves the context noise issue in a systematic way. For immersive recommendation, where we need to deal with data from a large and increasing number of different contexts, the CA-LDA’s ability to avoid costly hand-tuning is particularly valuable [CS07].

### 7.6 Evaluation with Real-World Data

We conducted a large-scale offline evaluation to study the performance of the proposed profiling algorithm, CA-LDA. We created profiles for individual users based on their public Twitter traces, and used that to predict what news and events they liked on Medium.com and Meetup.com. Note that Twitter data was the only digital trace used in the evaluation as it is one of a few publicly-available yet personally-identifiable data sources that allows for such a large-scale offline study. (This constraint is relaxed in the user study described in Section 8.3.) In the following, we describe our data collection and profile generation strategies.
7.6.1 Data Collection Strategy

Many Medium.com and Meetup.com users share their Twitter account name on their profile page. This enables us to evaluate how well our profiling algorithm can use users’ Twitter data (as a sample small data source) to predict their interests in news (Medium.com) and meetups (Meetup.com). We randomly chose 63,053 Medium.com and 50,000 Meetup.com users who declared their Twitter handle and crawled their public Twitter traces to create the profiles. For each of these users, we crawled: (a) their most recent 3,000 public tweets through Twitter API, (b) the tweets made by the people they followed, and (c) the tweets (made by other people) that were associated with the user’s hashtags (to capture topics the user paid attention to).

The users’ records on Medium.com and Meetup.com are taken as the ground truth for their news and event preferences. For Medium.com, the ground truth is the news articles each user has upvoted. For Meetup.com, the ground truth is the meetup groups in the New York City each user has joined. We crawled 31,000 news stories, and 11,823 meetup groups. On average each Medium.com user has upvoted 13.1 news stories, and each Meetup.com user has joined 5.1 meetup groups. The upvotes and group memberships both follow a long-tail distribution — the top 10% of the most popular items account for 59% and 61% of upvotes and group memberships respectively.

7.6.2 Profile Generation Strategy

We generated a profile for each user based on their Twitter traces. As suggested in [BZG14], rather than treating each individual tweet as an instance, we treated a set of tweets associated with the same source as one instance in order to infer a more robust topic distribution. Specifically, a user’s digital trace set $N_i$ consisted of a

---

3These users were chosen from 300,000 Medium users discovered through Medium.com’s upvote graph and about one million Meetup.com users based in the New York City.
unique instance that was composed of the most recent tweets made by this user and multiple instances that were composed of the tweets made by each of her followees, and the tweets associated with each hashtag she used. The followees’ tweets allowed us to enhance the profile precision in particular for passive users who posted only few or no tweets [PSV12]. The tweets associated with the hashtags allowed us to better understand the topics the user referred to. For robustness, we randomly selected 300 followees, and 300 hashtags to include into a user’s profile.

We defined the user profile $u_i$ as the weighted sum over the topic distributions of these instances as mentioned in Section 7.5. We determined the proper weighting through a grid search. The first instance, composed of the user’s own tweets, was weighted by the number of the tweets in it; the instances made by the followees and those associated with the hashtags were weighted by 5 and 0.2 respectively. The background topic for the Twitter context (i.e. $\phi_{\text{twitter}}$) was used to filter the Twitter-specific background noise in all the instances. We only included words that have more than 3 characters and stemmed the words before including them in the model. The stopwords and URLs were excluded.

The item profiles $v_j$ were generated in a similar fashion. The topic distributions of the article or the meetup description computed with the corresponding background topic (i.e. $\phi_{\text{medium}}$ or $\phi_{\text{meetup}}$) was taken as the profile for each item.

### 7.6.3 Evaluation Strategy

We randomly chose 3,000 Medium.com and Meetup.com users to test the profiling performance. As in [PSV12], for each user we created a set of items $J$ composed of items the user liked (denoted as $J_{\text{like}}$) and the user did not like (denoted as $J_{\text{dislike}}$). A good profiling algorithm should be able to discriminate $J_{\text{like}}$ from $J_{\text{dislike}}$ based on the similarity between the user profile and the item profile generated by the algorithm. Considering the average number of upvotes and group memberships
Figure 7.2: Precision-recall curves for different profiling algorithms. CA-LDA outperformed the prior algorithms by 18.7% and 77.4% in mean average precision for Medium.com and Meetup.com respectively.

per-user, for Medium.com users, we randomly chose 10 liked articles as positive examples, and 190 articles they did not like as negative examples. For Meetup.com users, we chose 5 meetups each of them joined, and 95 meetups they did not join. We ranked the items in \( J \) by the profile’s cosine similarity to the user’s profile and computed the prediction precision for different recall rates.

We compared the profiling performance of the proposed CA-LDA algorithm with that of the standard LDA, and \textit{doc2vec}, which is a state-of-the-art text representation learning algorithm based on neural networks \cite{Le:2014}. We used the LDA implementation from Mallet \cite{McC02} with \( \beta = 0.01 \) and \( \alpha_i = \frac{1}{K} \) for \( i=1,2,\ldots,K \). CA-LDA has additional parameters: \( \beta_c = 0.1 \) and \( (\gamma_\alpha, \gamma_\beta) = (0.2, 0.8) \). The \textit{doc2vec} implementation from \textit{Gensim} was used with the default parametrization \cite{Rud01}. We tested different model sizes for \( K = 50, 100\ldots500 \), and only presented the results for \( K = 200 \) where the performance of all three algorithms saturated. The same weighting scheme is used across different algorithms.
7.6.4 Evaluation Results

Figure 7.2 shows the average precision and recall curves with different profiling algorithms. CA-LDA outperformed both LDA and doc2vec in every case. Specifically, for Medium.com, CA-LDA had 18.7% improvement in mean Average Precision (mAP) over LDA due to its ability to focus on the salient topics referred to in users’ Twitter traces and in the item contents. For Meetup.com, CA-LDA had a much more significant (77.4%) improvement over LDA that was trained with only the meetup description corpus. In addition to the above-mentioned benefit, this demonstrated the advantage of co-training multiple corpora to allow a smaller corpus (i.e. the meetup description corpus) to benefit from the richer linguistic features contained in the others. In general doc2vec had a much poorer performance compared to LDA-based algorithms due to its sensitivity to the location of the words in a document (unlike LDA’s bag-of-word model) that made the model trained with text in one context less-generalizable to the text in another context in this specific task [LM14].

The result above showed that CA-LDA can effectively learn users’ interests from their Twitter data. We further evaluated the predictive power of different types of tweets. Table 7.2 reports the mAP of CA-LDA when different types of tweets were used individually, and when they were combined. Interestingly, the followees’ tweets were the most informative among all three types while the hashtag tweets were the least, which was consistent with the results in [PSV12]. The combination of the tweets only showed marginal improvement. This could be because the inherent correlation among these three signals [PSV12]. In the future work, we will explore if more sophisticated weighting schemes can further improve the performance.

In addition, we found that the length of Medium.com articles was also indicative of the users’ preferences and could additionally improve the CA-LDA’s mAP...
<table>
<thead>
<tr>
<th>mAP</th>
<th>Own</th>
<th>Followees</th>
<th>Hashtags</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>0.237</td>
<td>0.240</td>
<td>0.188</td>
<td>0.245</td>
</tr>
<tr>
<td>Meetup</td>
<td>0.331</td>
<td>0.347</td>
<td>0.282</td>
<td>0.353</td>
</tr>
</tbody>
</table>

Table 7.2: The predictive power of different types of tweets using CA-LDA.

by 13%, but the length of the meetup descriptions did not have such an effect. This was probably because the length of a news article is more related to the article’s quality. We also evaluated the performance of other kinds of information retrieval models. For example, BM25, one of the classical text-retrieval algorithms based on probabilistic relevance framework [RZ09], showed performance comparable to CA-LDA for Medium.com and had about 7% lower mAP than CA-LDA for Meetup.com. However, it is unclear how to combine the BM25 scores with the rating information, which is crucial to the performance of a practical recommendation system as we will see in the next chapter.

7.7 Conclusions

Learning users’ preferences from unstructured and noisy digital traces users generate in their daily life is one of the main challenges for immersive recommendations. In this chapter, we proposed Context-Aware LDA, a multi-context topic model that simultaneously learns users’ interests from multiple contexts and suppresses the noise introduced by each of the contexts. The evaluation results showed that the user profile learned by CA-LDA possesses significantly better predictive power to users’ preferences for news and local meetup events than the traditional topic model approaches. In the next chapter, we will describe how to utilize the immersive user profile learned in this chapter to enable a highly-personalized and practical recommendation system.
CHAPTER 8

Making Recommendations with Immersive User Profiles

8.1 Introduction

In this chapter, we introduce the proposed hybrid recommendation model, Collaborative User-Item Regression, that combines the immersive user profile, item profile, and rating information to enable a highly-personalized and practical recommendation system. We describe two large-scale offline recommendation tasks conducted to evaluate the accuracy of the proposed approach using Medium.com and Meetup.com data. Then, we present a 33-person user study conducted to evaluate the recommendation utilities in a more interactive setting with users’ direct feedback. In the both cases, immersive recommendations showed significant improvement over the state-of-the-art recommendation approaches in both accuracy and utilities, which suggests its promising potential in improving today’s recommendation systems.

8.2 Collaborative User-Item Regression

Given user profile $u_i$ and item profile $v_j$, we are able to identify items that are relevant to a user’s interests. However, the relevance alone is not sufficient in a practical recommendation system. For example, within a large set of items, there may still have a large number of items that are relevant to a user’s interests. Fur-
ther filtering is needed to find the items that will have the highest user-perceived quality. Another issue specific to Immersive Recommendation is that a person’s interests would vary from one platform to another; fine-tuning is needed to better match the recommendations to the user’s specific interests on the target platform.

We propose a hybrid collaborative filtering algorithm, called **collaborative user-item regression**, that carefully fuses the objective user/item profiles and the subjective rating information to predict the ratings \( \hat{r}_{ij} \) that are still unknown. This model is built on the foundation of *regression-based latent factor* [AC09, Ble12]. On top of the user profile and item profile, we introduce *latent user offset* \( \eta_i \in \mathbb{R}^K \) and *latent item offset* \( \epsilon_j \in \mathbb{R}^K \) to capture the preference information that is not captured by the user/item profile, but manifested in the ratings. The model assumes a generative process for ratings \( r_{ij} \) as follows:

1. For each user \( i \), draw user offset \( \eta_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K) \)
2. For each item \( j \), draw item offset \( \epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K) \)
3. For each user-item pair \((i, j)\), draw the rating
   \[
   r_{ij} \sim \mathcal{N}((u_i + \eta_i)^T(v_j + \epsilon_j), c_{ij}^{-1}).
   \] (8.1)

The key ingredient of this model lies in Eq. [8.1] As \( v_j \) and \( u_i \) are fixed, the free parameters \( \epsilon_j \) and \( \eta_i \) will be tuned in a way that the resultant inner product approximates to the rating \( r_{ij} \) and makes up the difference between the user rating \( r_{ij} \) and the user/item profile relevance, i.e. \( u_i^T v_j \). This design allows \( \epsilon_j \) and \( \eta_i \) to capture the preference information that is missing from the user/item profiles.

Specifically, the item offset \( \epsilon_j \) makes up for the hidden characteristics of item \( j \) that are not captured by the item profile \( v_j \). Consider article *The Difference Between Living in New York and San Francisco* from Medium.com as an example [Coo15]. Based on the content, this article is not related to technology and thus
its profile $v_j$ has a small weight on the tech-related topics. However, many tech people actually enjoy reading this article and give it high ratings, probably because New York and San Francisco are the two cities where many tech people live or may consider moving to. As these tech people’s profile $u_i$ tend to have a large weight at the tech-related topics, in order to make the inner product in Eq. 8.1 approximate to the high ratings $r_{ij}$ given by these users, the item offset $\epsilon_j$ will be tuned to have a larger weight in the dimensions that correspond to the tech-related topics, and this, in turn, will make this article more likely to be recommended to the users who are interested in tech (i.e. whose $u_i$ has a larger weight in the tech-related topics) in the future.

On the other hand, the user offset $\eta_i$ makes up for the difference between a user’s initial profile $u_i$ and her specific interests on the target platform. To see this, consider a user who gave high ratings to many rock music articles while her initial profile $u_i$ shows little interest in rock music. Assuming the remaining parameters are fixed, in order to satisfy Eq. 8.1 the user offset $\eta_i$ will be tuned to have a larger weight on the rock music related dimension to match the high ratings, and the other articles that are related to rock music will become more likely to be recommended to this user in the future. This adjustment is important for immersive recommendation as a user may have specific interests on the target platform that are not revealed in her profile. For a new user, her offset $\eta_i$ is a zero vector, and the recommendations are made solely based on her profile. However, once she starts to give ratings, $\eta_i$ will be tuned to compensate for the difference.

The scale of $\epsilon_j$ and $\eta_i$ is controlled by the regularization parameters $\lambda_v$ and $\lambda_u$. For example, when $\lambda_u$ is smaller, a larger $\eta_i$ is allowed, and the recommendation will lean more towards the preferences expressed in user $i$’s ratings than towards her initial profile $u_i$.

The precision parameter $c_{ij}$ serves as a confidence parameter for rating $r_{ij}$, and is set larger if we trust the rating $r_{ij}$ more. This confidence parameter is
useful when dealing with implicit ratings [HKV08]. For example, in the case of Medium.com, users only “upvote” the articles they like, and do not have a means to express their dislike for an article. As such, the case that a user did not upvote an article could be interpreted as either a) the user did not like the story or b) the user was not aware of the story. Therefore, for stories that did not get upvoted, we set \( r_{ij} = 0 \) and have a lower \( c_{ij} \) to capture this uncertainty [HKV08, WB11], specifically:

\[
c_{ij} = \begin{cases} 
a, & \text{if } r_{ij} = 1 
a, & \text{if } r_{ij} = 0
\end{cases}
\]  

(8.2)

where \( r_{ij} = 1 \) if user \( i \) upvoted item \( j \) or otherwise; \( a \) and \( b \) are tuning parameter, and \( a > b > 0 \). On the other hand, for ratings that are made on a 1-5 Likert scale, \( r_{ij} \) can be set to numbers between 0 and 1 that represent different degrees of support to an item, and \( c_{ij} \) can be set accordingly to represent the rating confidence.

8.2.1 Parameter learning

Since directly computing the posterior distribution of \( \epsilon_j \) and \( \eta_i \) is intractable, following Wang and Blei’s formulation [WB11], we use an EM-algorithm [DLR77] to estimate their Maximum A Posteriori probability (MAP). The complete log likelihood given \( \eta_i \), and \( \epsilon_j \) is:

\[
\mathcal{L} = -\frac{\lambda_u}{2} \sum_i \eta_i^T \eta_i - \frac{\lambda_v}{2} \sum_j \epsilon_j^T \epsilon_j - \sum_{i,j} \frac{c_{ij}}{2} (r_{ij} - (\mathbf{u}_i + \eta_i)^T (\mathbf{v}_j + \epsilon_j))^2
\]

(8.3)
We optimize the log-likelihood $L$ by a coordinated ascent. Let $\hat{u}_i = u_i + \eta_i$, $\hat{v}_j = v_j + \epsilon_j$, and $\hat{U} = \hat{u}_{i=1}^T$, $\hat{V} = \hat{v}_{j=1}^T$. We take the gradient of $L$ with respect to $\hat{u}_i$ and $\hat{v}_j$, respectively. Setting the gradient to zero gets:

\[
\hat{u}_i \leftarrow (\hat{V}C_i\hat{V}^T + \lambda_uI_K)^{-1}(\hat{V}C_iR_i + \lambda_uu_i) \tag{8.4}
\]

\[
\hat{v}_j \leftarrow (\hat{U}C_j\hat{U}^T + \lambda_vI_K)^{-1}(\hat{U}C_jR_j + \lambda_vv_j) \tag{8.5}
\]

where $C_i$ is a diagonal matrix with $c_{ij}$ for $j = 1..., J$ as its diagonal elements and $R_i = (r_{ij})_{j=1}^J$ for user $i$. $C_j$ and $R_j$ are defined in a similar way for item $j$.

In each iteration, we update all $\hat{u}_i$ with the latest estimation of $\hat{v}_j$, and update all $\hat{v}_j$ with the latest estimation of $\hat{u}_i$. This estimation process stops when the log-likelihood $L$ converges, and the offsets $\eta_i$ and $\epsilon_j$ can be computed accordingly. The complexity of each iteration is linear to the number of known ratings $r_{ij}$, but each iteration enjoys a high degree of parallelism as all the $\hat{u}_i$ and all the $\hat{v}_j$ can be estimated concurrently.

8.2.2 Prediction and Updating

After offsets $\eta_i$ and $\epsilon_j$ are learned, we use our model to predict an unseen rating $\hat{r}_{ij}$ as follow:

\[
\hat{r}_{ij} \approx (u_i + \eta_i)^T(v_j + \epsilon_j), \tag{8.6}
\]

where $\hat{r}_{ij}$ is the expectation of the rating given every known rating $r_{ij}$ and every user/item profile.

When a user starts to make new ratings, the new ratings are incorporated into the model and update the user offset $\eta_i$ by optimizing the $\hat{u}_i$ with the updated $C_i$ and $R_i$ as in Eq. 8.4. However, computing $\hat{V}C_i\hat{V}^T$ has time complexity $O(J)$ and is too slow for real-time personalization when the number of item $J$ is large.
This, however, can be optimized based on the observation that

$$\hat{V}Ci \hat{V}^T = b\hat{V}\hat{V}^T + (a - b) \sum_{j \in S(i)} \hat{v}_j \hat{v}_j^T,$$  \hspace{1cm} (8.7)

where $S(i)$ is the set of items user $i$ has upvoted. The optimization is done by caching $b\hat{V}\hat{V}^T$, and only computing $(a - b) \sum_{j \in S(i)} \hat{v}_j \hat{v}_j^T$ in the update process. This optimization introduces $\frac{J - |S(i)|}{|S(i)|}$ times of speed-up, which is quite significant as $J \gg |S(i)|$ in most cases.

### 8.2.3 Recommendation Performance

We conducted a large-scale recommendation task to measure the end-to-end performance of immersive recommendations with the proposed collaborative user-item regression model (denoted as ImmRec). The same data described in Section 7.6 were used to conduct the evaluation. We evaluated the recommendation performance for Medium.com users and Meetup.com users, and compared the recommendation accuracy with the following prior approaches:

1. Content-Based (CONTENT) ranks the items by cosine-similarity between the user/item profiles that CA-LDA learned. This algorithm represents pure content-based models. Whenever a user likes an item, the profile is updated by adding the item profile $v_j \times \alpha$, where the smoothing parameter $\alpha$ is set 0.1 for the best performance.

2. Most Popular First (POPULAR) ranks the items by number of upvotes or members an item had, which is a common baseline for the user-cold-start problem. While simple, this algorithm is a strong baseline for comparison in the case of news and events recommendations as the majority of upvotes or group memberships are concentrated in a small subset of items.

3. Probabilistic Matrix Factoring (PMF) makes recommendations without
user or item profiles [MS07]. PMF represents traditional user-user collaborative filtering models.

4. Collaborative Topic Modeling (CTM) uses PMF along with item profiles to improve the recommendations [WB11]. CTM is one of the state-of-the-art document recommendation models, and represents existing hybrid recommendation algorithms in use.

We set $K = 200$, $a = 1$, and $b = 0.01$ for all the collaborative filtering based algorithms, including ImmRec, PMF, and CTM, and set $\lambda_u = \lambda_v = 10$ for ImmRec, $\lambda_u = \lambda_v = 0.01$ for PMF, and $\lambda_u = 0.01, \lambda_v = 10$ for CTM according to the prior work [Ble12, MS07].

8.2.3.1 Training/Testing Data Segregation

We randomly chose 5,000 Medium.com and 5,000 Meetup.com users (out of 63,053 Medium.com and 50,000 Meetup.com users in the datasets) to create testing user sets whose upvotes and membership information were excluded from the model. The models for Medium.com and Meetup.com were trained with the rest of the users’ data. We evaluated the recommendation accuracy for the testing users starting with the time at which they had made no feedback (i.e. user-cold-start phase) up to the time that they had made 10 upvotes or had joined 5 meetup groups. To evaluate the recommendation accuracy under a more realistic settings, for Medium.com, we included the upvotes each testing user made before January 1st 2015 in a chronological order, and made recommendations only for the stories published after January 1st 2015 and before October 1st 2015, which were 22,863 stories in total. This segregation was to ensure that the recommendations were made based on the same pool of items in every case. For meetups, where such temporal information was not available, we randomly chose 2,000 meetups for recommendation and included users’ ratings to the rest meetups in a randomized
8.2.3.2 Evaluation Metrics

We generated the top-50 news stories and meetup groups based on the predicted ratings $\hat{r}_{ij}$, and computed the following metrics based on the items the users actually upvoted or joined [WKL07].

- **Average Recall Rate**: Recall rates measure the proportion of positive items that the algorithm was able to identify in a top-M recommendation task. The recall rate for each user is defined as below:

$$\text{Recall}@M = \frac{\text{number of items the user liked in top } M}{\text{total number of items the user liked}}.$$  

- **Mean Reciprocal Rank (MRR)**: MRR measures the ranking of the first correct item and averages over all the users. This measure provides insight into the ability of the algorithm to return a correct recommendation at the top of the ranking. It is defined as follows:

$$\text{MRR} = \frac{1}{|I|} \sum_{i=1}^{I} \frac{1}{\text{rank}_i},$$

where rank$_i$ is the rank of the first correct item for user $i$.

Note that, as in most prior work, we were not able to compute the recommendation precision. This was because when a user did not upvote or join an item, we do not know if it was because she did not like it or because she was not aware of it [WB11]. This drawback was addressed in the user study described in Section 8.3.
Figure 8.1: Average Recall@50 and Mean Reciprocal Rank when a user had made 0 to 5 or 10 feedback signals. Immersive recommendation (ImmRec) significantly outperformed the second best approach in every case by up to 57.9% in Recall@50 and 42.6% in MRR, and was able to smoothly improve the performance when more feedback was available (post cold start).

8.2.3.3 Evaluation Results

The evaluation results in terms of the Recall and MRR are shown in Figure 8.1. Note that the content-based algorithm (CONTENT) performed over 5x and 3x worse than the popularity-based baseline (POPULAR) for Medium.com and Meetup.com respectively in both cold-start and post-cold-start phases. It is not shown in the figure and omitted hereafter to focus the discussion, but its poor performance underscored the importance of incorporating rating information into
the recommendations.

Compared to the remaining algorithms, ImmRec significantly outperformed every other approach in both average recall rate and MRR for Medium.com users, and maintained at least a 14.7% to 42.6% margin over the second best algorithm. For example, when a user had not made any upvote (i.e. in the user-cold-start phase), our approach was able to make recommendations that were even more accurate than the recommendations PMF and CTM were able to make after 10 upvotes (post cold-start). According to the data we collected, it would take an average user 261 days to make that many upvotes. The results also demonstrated the ImmRec’s fine-tuning performance. When a user started to make upvotes, the algorithm was able to incorporate these signals and smoothly improve the recommendations over time. One exception was when a user made fewer than 2 upvotes. In those cases, the user offset $\eta_i$ leaned too much towards the profile of those few items, and degraded the overall accuracy. This drawback can be addressed by putting a smoothing coefficient in front of $\eta_i$ in Eq. 8.6 when the number of upvotes is small, or more systematically, by learning the $\eta_i$ at different stages from the data as in [YRS14].

Another noteworthy result is the superior MRR of ImmRec. ImmRec’s MRR was always the highest among all the algorithms while other collaborative filtering algorithms (i.e. PMF, CTM) were not able to surpass the baseline until later in the user’s lifetime. This demonstrates ImmRec’s ability to push the relevant items further up into the ranking, which is an important requirement for many recommendation systems [SKB12].

The results for meetup recommendation followed a similar trend. As shown in Figure 8.1b, the ImmRec outperformed the second best algorithm in every case by a 9.3% and to 42.5% margin. However, the gap between ImmRec and PMF or CTM shrunk quickly particularly in the recall rates. This was probably due to the fact that a user usually had much narrower preferences in terms of joining
meetups than reading news articles (manifested in an 5.1% lower average topical entropy according to our topic model). Therefore, PMF and CTM could quickly learn a user’s meetup preferences with only a small amount of feedback. Even so, the high-quality cold-start recommendations and a high MRR still make ImmRec a more desirable choice.

8.2.4 Model Limitations

While demonstrating promising performances, our recommendation model has two known limitations we intend to explore in the future work. First, as in CTM and PMF, the proposed collaborative user-item regression model lacks the user and item bias terms. The user bias term is a scalar variable associated with each user, used to capture a user’s tendency of giving higher or lower ratings than other users. Similarly, item bias is associated with each item, used to capture an item’s tendency of receiving higher or lower ratings than other items. We will explore whether adding these bias terms to Equation 8.1 can further improve the accuracy.

Another more critical issue with our model is that the dimensionality of the user/item offsets tends to be much higher than that in the more traditional matrix factorization models (whose dimensionality is usually between 10 and 50). This is because that the dimensionality of user/item offsets is coupled with that of the user/item profiles, and both of them need to be of the same dimensions. In the user profiling phase, we tend to choose a rather high-dimensional user profile in order to capture users’ finer-grained interests. However, in the recommendation phase, we may want to keep the dimensionality low to avoid overfitting.

This tension can be addressed by creating a learnable transformation function $f$ that projects the user/item profiles into a lower-dimensional space before including them into the recommendation model. Specifically, we replace $u_i, v_j$ in
Equation 8.1 with $f(u_i), f(v_j)$. Then, the user/item offsets and the recommendation will operate in this new lower-dimensional space rather than in the original topical space. In this way, we decouple the dimensionality of the profile variables and the offset variables, which, in turn, results in a much smaller model size. The transformation function $f$ can be a simple linear function, e.g. $f(u_i) = W u_i + b$, where $W \in \mathbb{R}^{N \times K}$, $b \in \mathbb{R}^N$, and $N < K$. More advanced transformation functions, such as Multilayer Perception, can be used as well.

8.3 User Study

We conducted an initial, small user study ($N=33$) to explore the utility of immersive recommendation in an interactive setting [SG11]. Moreover, in this study we included not only users’ Twitter data, but also Facebook and email traces. The personal communication in email is worthy of exploration for its potential representation of a broader range of interests than what is expressed in social media alone, where impression and audience management dominate [Hog10, Mb11].

The goal of the experiment was to compare the performance of ImmRec to other algorithms using direct evaluation by users of the recommendations given by each. As in the offline evaluation, we studied the ImmRec’s performance for both the user-cold-start and post-user-cold-start phases and compared its performance to that of Most-Popular, Random, PMF, and CTM. To avoid bias due to between-subject differences [SG11], we adopted a within-subjects design, in which each participant rated items recommended by each of the algorithms, and the performance of the algorithms were compared on a per-user basis [Kep91, SG11].

8.3.1 Experiment Design

The experiment consisted of two sessions, one for news and one for meetup. Each session consisted of two phases: a Cold-Start phase and a Steady-State phase. In
In the **Cold-Start Phase**, we compared three algorithms: ImmRec, Most-Popular, Random. We presented to the user the top six recommendations generated by each algorithm. The users were not told which items were recommended by which algorithm. As a treatment to the carryover effect, where items presented earlier cause bias in rating later items, the recommended items were presented one at a time, in a randomized order. If two algorithms’ recommendation coincide, the same item would only be presented once. For each item, the participants were asked to specify how interesting the presented item is on a 1 to 5 Likert scale. The descriptions of the levels were assigned based on [DSM11] from "Not at all interesting" to "Extremely interesting".

In the **Steady-State Phase**, we used some of the ratings provided by the participant in the first phase, and compared four algorithms: ImmRec, Most-Popular, PMF, and CTM. We presented the top six recommendations from each to each participant in the same fashion as described above. In this phase, of course, the algorithms (other than Most-Popular) could generate recommendations based on the ratings the participants made earlier. Since recommendations made by ImmRec earlier would contain information about the user’s profile, for these previous ratings, we only used the ratings for the items recommended by the Most-Popular and Random algorithms in order to prevent other algorithms from benefiting from this information. For a Likert scale rating $l = 1$ to 5, we set $r_{ij} = 0.25 \times (l - 1)$, and $c_{ij} = 1$ to update the recommendation models as described in Section 8.2.2.

The participants were asked to use a web app to connect to at least one of their Gmail, Facebook, and Twitter accounts for the system to access their traces. For email, we removed the signatures and treated each email thread the user sent or forwarded as an equally-weighted instance. For Facebook, similar to the strategy for Twitter (Section 7.6.2), we treated all the posts made by the user

---

1. We observed significant fatigue in a pilot study when more than six recommendations from each algorithm were shown.
as one instance weighted by the number of posts in it, and those made by each Facebook Page the user liked as one instance weighted by 5 due to their similar natural as Twitter followees. The profiles from the three different contexts were normalized to their \( L_1 \)-norm and summed together. Due to limited data access, we were not able to further fine-tune the parameterization prior to the experiment (in particular for email); this is an important area for future research. For the recommendations themselves, we used 13,250 articles from Medium.com published between July 1st 2015 and October 1st 2015 (news), and all 11,823 meetup groups in the New York City (meetup).

The participants were recruited through mailing lists and flyers on a university campus in New York City. The participants included 1 faculty member, 5 staff members, 2 undergraduates, and 25 graduate students. The study was approved by Cornell Institutional Review Board Protocol #1507005739.

8.3.2 Evaluation Metrics

We performed a robust examination of the experiment results using numerous strategies for treating the rating data and for the statistical analysis of the differences between conditions. For lack of space, we focus on one set of strategies here, and note the alternative methods when relevant.

As a common treatment to the ordinal data, we first transformed each level in the Likert scale to a value between 0 and 1 using their average cumulative proportion \( \text{Agr10} \), given by:

\[
a_j = \sum_{k=1}^{j-1} p_k + 0.5p_j,
\]

where \( p_j \) is the proportion of level \( j \) among all the ratings. This step is critical as
the differences between two consecutive levels were not necessarily uniform. For example, in prior studies of movie recommendation, users had a higher chance to switch their ratings between 2 and 3 than to switch their ratings between 4 and 5, which indicated that there was a larger user-perceived distance between 4 and 5 [APO09]. We devised the transform schemes for the news articles and meetups separately based on all the users’ ratings. The ratings of news stories showed the similar phenomenon as in movie recommendations described in [APO09], while the meetup ratings were more uniformly distributed. For robustness, we performed the analysis using other functions, including directly using the raw 1–5 Likert scale, or cutting off the ratings at 3.5 and assigning binary scores of 0 and 1 [HKT04]. We found similar results with these alternative functions.

For each participant $p$, we took $U_{p,a}$, the average of the (transformed) ratings that the participant gave to algorithm $a$’s recommendations, as the utility of that algorithm for the participant. We used $U_{p,a}$ to compare the different algorithms in the analysis below.

In addition, to assess the scale of the average improvement of ImmRec, for each competing algorithm $a$, we computed the average utility improvement ImmRec had over algorithm $a$ across all the users, and normalized it by the respectively global mean of news and meetup recommendations based on all the users’ ratings, which is:

$$\frac{1}{M|P|} \sum_p (U_{p,\text{ImmRec}} - U_{p,\text{Other}}),$$

where global mean $M = 0.48$ and 0.47 for news and meetup respectively. We used the improvement metric in the figure below.

### 8.3.3 Evaluation Results

We compared the recommendation performance using the metrics described above. The results, in terms of the improvement, are shown in Figure 8.2. The figure
shows the performance improvement of ImmRec over the other algorithms in
different settings (news on top, in green; meetup on bottom, in orange) and in
cold-start (left) and post-cold-start settings. For example, the left-most bottom
bar shows that ImmRec improved over Most-popular by 20.7% in the cold-start
Meetup settings. Note that the Random algorithm had an 2x worse performance
than the ImmRec and so is not shown in the figure and omitted hereafter to focus
the discussion.

We used the $U_{p,a}$ scores to evaluate whether ImmRec outperformed the other
algorithms. The statistical significance of the improvement was evaluated using
the paired Student’s t-test\footnote{We also used ANOVA for repeated measures and Wilcoxon signed-rank test and found
similar results.} as suggested in \cite{SG11}, and the effect size was evalu-
ated using Cohen’s $d$ \cite{Coh13}.\footnote{Cohen’s $d = \frac{\bar{X}_d}{S_d}$, where $\bar{X}_d$ and $S_d$ are the mean and standard deviation of the differences
between two algorithms’ performances.}

As demonstrated in Figure 8.2, ImmRec had an improvement in utility over
the other algorithms in every case. In the cold-start phase for the meetup re-
commendation, ImmRec had the greatest improvement. The improvement was
statistically significant with $p = 0.00018$ over the Most-Popular algorithm, and
the effect size Cohen’s $d$ was 0.78.\footnote{An effect size $d = 0.5$ is considered to be a medium effect, and $d = 0.8$ is a large effect.}

This large improvement was probably due to
the fact that people tend to join meetups that are closely related to their daily
activities or major interests, which can be more effectively learned from their
digital traces (in relative to the more diverse interests in news articles). On the
other hand, a smaller improvement (6.3%) was observed in the cold-start phase
for the news recommendation. This low performance was probably due to the
fact that extremely popular news articles, while not directly relevant to a user’s
interests, may still have a high chance to appeal to the user. Even so, ImmRec was
still able to consistently outperform the Most-Popular algorithm with statistical
significance $p = 0.009$ and effect size $d = 0.50$.\footnote{We also used ANOVA for repeated measures and Wilcoxon signed-rank test and found
similar results.}
Figure 8.2: Expected improvement of ImmRec in utility over other algorithms in the user experiment ($N=33$). ImmRec outperformed other algorithms by up to 20.7%, but had less significant improvement during the cold-start phase for news and the steady-state phase for meetups. (Asterisks represent the significance level of the paired t-test. *: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)

Across the board, ImmRec performed better than all other algorithms, though in two cases (compared to CTM and PMF in the steady-state phase for Meetup recommendation) the performance improvement was not statistically significant. This result is consistent with the results we observed in the offline evaluation, where the advantage of ImmRec shrunk quickly after the users made more feedback (see Section 8.2.3.3). It is also noteworthy that, in our analysis, only ImmRec showed statistically significant improvement over the Most-Popular baseline, while CTM and PMF did not for either news or meetup recommendations. Future work can use human ratings to verify that the performance characteristics would remain for items that are beyond the top six results explored in this study, as suggested by the analysis in Section 8.2.3.3.
8.3.4 Error Analysis

Through a closer examination of the cases where ImmRec performed relatively poor, we observed a strong correlation between the topical entropy of a user’s profile\(^5\) and the utility of ImmRec in the cold-start phase \((r=0.49, p < 0.003)\) while no correlation was observed in the steady-state phase. The topical entropy of the profile also strongly correlated with the number of different types of traces used to generate the profile \((r = 0.70, p < 0.001)\). These correlations suggested that when a user’s profile was biased towards few topics (i.e low entropy), ImmRec’s recommendations tended to be less satisfactory for some users in the cold-start-phase, and ImmRec was able to mitigate this issue in the steady-state phase through the fine-tuning. Based on this, we suggest practitioners of ImmRec to elicit as many diverse traces as possible from users and mix cold-start recommendations with some popular items in order for ImmRec to adjust for the potential discrepancies between the user profile and the user’s interests. In future work, we will explore whether more sophisticated weighting schemes can increase the profile entropy. Moreover, questions, such as what kinds of traces contribute most to the recommendations, and how the amount of data changes the recommendation performance, will be explored in the future study as well.

8.4 Conclusions

In Chapter 6-8, we presented Immersive Recommendation, a user-centric recommendation model that empowers individuals to benefit from their diverse digital traces and enable a richer, more personally-relevant and desirable recommendation results. We proposed a topic-model-based algorithm to simultaneously profile multi-context user digital traces and a hybrid collaborative filtering model to improve the recommendation quality beyond the user-cold-start phase. We

\(^5\)Topical entropy is defined as \(H(u_i) = -\sum_k p(y_{ik}) \log p(y_{ik})\)
targeted news and local-event recommendations for their utility and societal importance and conducted a large-scale offline evaluation with Medium.com and Meetup.com users based on their publicly-available Twitter traces. We further verified the results with a 33-person user study with more diverse digital traces and the direct user evaluation. In almost every case, immersive recommendation showed significant improvement over the prior algorithms, which establishes the feasibility and justifies the potential efficacy of this new recommendation model.

While we only focus on text data in the present work, the promising results suggest a fruitful research avenue for the investigation of other digital traces, including location and travel histories, personally-created or viewed images, and the online purchase and consumption histories. An important potential benefit of immersive recommendation is to turn a recommendation system into a tool for awareness and aspiration. For example, the techniques proposed in the present work can be used to create a pathway for user interaction with the personalization/recommendation system to intentionally bias the system toward not only the user’s observed behaviors, but the user’s aspirational goals. Future work will explore how users can inform recommendation systems with their intentions so as to break out of their behavioral loops.
CHAPTER 9

Conclusions

In this thesis, we introduce the notion of small data, which are the digital traces we generate in our everyday life when interacting with a myriad of digital devices and services. Small data systems are user-centric systems that combine the diverse small data of individual users to create services and applications that are more personalized and individualized for each individual user. We focus on two application contexts for small data systems in this thesis.

In Chapter 2-4, we discussed the potential of using small data, in particular the smartphone data to enable personal behavioral analysis. We presented our contributions, Lifestreams, a modular data analysis toolset for small data behavioral analysis. Lifestreams’ 4-tiered software stack and extensible set of analytical building blocks facilitate the exploration of diverse personal data streams by extracting important behavioral indicators and identifying key patterns in individuals’ everyday life behaviors.

We evaluated the Lifestreams’ ability to analyze multiple heterogeneous data streams by applying it to a dataset collected during a 6-month study with 44 young mothers. Lifestreams was able to identify important patterns and major behavior-changing life events that potentially were not otherwise identified by the individuals themselves or researchers. Lifestreams’ visualization component was also used in the interviews with the study participants guided by a research coordinator trained with the system. We explored the generality and extensibility of the Lifestreams by applying it to two additional studies, one with continuous
audio data and another of much larger scale.

In Chapter 5, we discussed the shared system challenges arising in small data systems. We proposed a modular system architecture to address these challenges and help facilitating the development of small data systems. The central component of the proposed architecture is Lifestreams DB, an RDF-based database system that provides a unified interface for querying, combining, and fusing diverse small data streams. Lifestreams DB adopted a soft-state design and a chunk-based data management scheme to significantly improve the storage efficiency, query performance, and security of the small data applications built on top of it. We evaluated the performance of Lifestreams DB through three sample applications and real-world users’ small data traces.

In Chapter 6-8, we turned our attention to the second application context — personalizing online experiences using small data. We proposed immersive recommendations, a new recommendation paradigm that utilizes a wide range of individuals’ small data to make highly-personalized recommendations, satisfying users’ diverse interests. We attacked two major technical challenges in this application context — namely how to profile users’ diverse digital traces and how to make effective recommendations with immersive user profiles. Our key contributions to this application context are 1) Context-Aware LDA, a topic model based user profiling algorithm that simultaneously learns users’ interests from multiple small data streams and suppress the noise introduced by different contexts and 2) Collaborative User-Item Regression, a hybrid collaborative filtering algorithm that is able to combine immersive user profile with item profile and rating information to make high quality recommendations and allow rapid fine-tuning based on users’ feedback.

We conducted large-scale offline evaluations, a 33-person user study, and a real-world service deployment to explore the feasibility, efficacy, and practicability of immersive recommendations. We focused on two application domains: news
and local meetup event recommendations. The results suggest the promising benefits of applying immersive recommendations to improve the future recommender systems. For example, for news recommendations, our approach outperforms the prior in-channel recommendation approaches by up to 42.6% in terms of recall@50, and the recommendation accuracy for the cold-start users (i.e. users with no rating) was even better than that of the other approaches after they included 10 feedback signals from the users. These promising results also call for further investigation to include broader small data sources and the refinement of the techniques through larger-scale real-world experiments.

To summarize, this thesis discusses the increasing opportunities of using small data to produce deeper and more comprehensive insights across the union of a user’s small data. We touch on the issues of extracting useful information from small data and constructing rich user models and robust systems to enable novel applications in the contexts of enabling personal behavioral analysis and highly-individualized online experiences.

Of course, there is much work to be explored beyond this thesis. For example, we only touched on a small subset of our small data in this thesis. The promising results in this thesis suggest a fruitful research avenue for the investigation of other digital traces, such as the image data we capture or like on social media [YHE15], our interaction traces with Internet-of-Things or smart home devices, our purchase and transactional history captured by smart wallets, and our working traces, such as the code and documents we write. There is a great potential of using these traces to enable novel small data applications, and new information extraction algorithms are required to unlock the potential of these data.

One challenge of conducting small data research is that many machine learning models we use today (e.g. topic models mentioned in Chapter [7]) require a large amount of data to construct a robust and generalizable model. It may seem challenging to collect personal data at such a scale and variety. However,
as we explored earlier in immersive recommendations, we were able to address this problem with a *big-to-small strategy* — we pre-train the model with the *big* publicly-available datasets, and apply the model to the users’ personal data at runtime and make necessary fine-tuning with the users’ data. As this pre-training process is required by many small data systems, we would like to explore the important consequent research question of how to prevent the biases and stereotypes contained in the public datasets from leaking into the users’ personal models during the pre-training process [BCZ16].

Furthermore, there is also much future work for each specific application context. For example, in Lifestreams, we developed a system that can systematically identify trends and patterns in users’ daily behaviors. Ultimately, we would like to help researchers and care givers to develop interventions that can greatly improve the personalization and precision of patient-centric care. For example, for patients who had hip surgery, the care giver can use a tailored instance of Lifestreams to check their data weekly to identify whether patients are having any trouble complying with the recovery procedure. After a few months, the care giver can use Lifestreams to study the longer-term interaction between patients recovery progress and their daily life behaviors, such as working schedules, sleep patterns, stress and mood, and adjust the treatment accordingly. Assisted with the visualizations generated by Lifestreams, the care giver can use this information to guide the discussion with patients and families to provide feedback and to help patients with problem solving, goal setting, and monitoring progress towards goals.

For immersive recommendations, an immediate extension we would like to explore is to extend the idea for individual users to a group of people to enable group immersive recommendations (See [WHY16] for a sample application). Moreover, we would also like to explore the notion of *aspirational recommendations* in which we allow users to review and modify their user profile learned from their small data
and create a pathway for them to intentionally bias the system toward not only their observed past behaviors (as in immersive recommendations), but also their aspirational behaviors and future goals. Today’s recommendation systems tend to create effective positive feedback loops that optimize recommendations based on a model of the user built on their observed behavior [Par11, NHH14]. We will explore how users can inform recommendation systems with their intentions so as to break out of behavioral loops, in particular, in the context of news recommendations, civil engagements, and food consumption personalization, where such behavioral loops tend to have undesirable consequences.

Granted, small data systems are still very much in its infancy. Some of the most useful data streams, such as transaction history or health records, are still locked in their data silos with no direct programmatic access; and significant privacy issues need to be addressed before a wider adoption and utilization of small data can be possible. Nonetheless, the results of this work suggest that many small data applications are not only readily feasible today, but also can enable services that are tremendously beneficial to individual users and would not be possible otherwise with the traditional provider-centric model that has access to only narrow view of individuals’ behaviors. We believe that by creating methods, tools, and systems with the notion of small data in mind and recognizing the role of the individuals as beneficiary of their own data, ultimately, we can build a future in which small data systems can address the complement of the big data problem: rather than drawing generalizable insights about populations across broad swaths of data for purposes of similar scale (e.g. corporate, governmental, etc.), we can draw insights about the individuals and help them use their own data to better manage their well-being, to improve their productivity, and ultimately to enable them to achieve their personal goals.
References


[YRS14] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. “Personalized entity recommendation: A heterogeneous information network approach.” In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pp. 283–292, 2014.


