DEMOCRATIZING INTERACTION MINING

BY

DENIZ ARSAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Doctoral Committee:

Associate Professor Ranjitha Kumar, Chair
Professor Darko Marinov
Assistant Professor Reyhaneh Jabbarvand
Dr. Jeffrey Nichols, University of Chicago

ABSTRACT

In the digital landscape, defined by a multitude of mobile applications spanning various platforms, our daily lives are shaped by the quality of digital experiences. The impact of these experiences extends beyond individual satisfaction and directly influences the success of organizations, driving the need for data-driven methods to evaluate and enhance user interfaces. Traditional approaches like analytics and A/B testing, while valuable, require access to an application's codebase, limiting their utility for external applications. In response to these limitations, interaction mining has emerged as a potent technique. Interaction mining entails the capture of design and interaction data as users engage with an application, resulting in the creation of interaction traces. However, existing interaction mining systems rely on intricate OS-level interventions to enable comprehensive data capture.

This dissertation introduces **O**n-**D**evice **I**nteraction **M**ining (ODIM), a framework that democratizes interaction mining. ODIM enables data capture by enabling in-the-wild interaction data collection from any Android app on personal devices, without the need for specialized hardware or modifications to the operating system. ODIM empowers researchers, designers, and industry practitioners to enhance task automation, ensure robust user privacy, and bridge the gap between analytics and UX testing. These contributions provide the tools and methodologies needed to innovate digital experiences while safeguarding user privacy. ODIM is a step towards a more accessible, principled, and privacy-conscious future for interaction mining.

To my wife,

ACKNOWLEDGMENTS

This work would not have been possible without the unwavering support of my amazing advisor, Ranjitha Kumar. Her warm welcome into her research group when I was in the process of changing research areas, coupled with her constant trust and reassurance, formed the foundation for the realization of this work. Ranjitha not only believed in me but consistently championed my confidence, irrespective of external opinions. The shared passion we experienced is truly invaluable. My heartfelt gratitude goes to her for being more than an advisor, for shaping this academic journey in profound ways.

In addition, I extend my sincere appreciation to my esteemed committee members – Darko Marinov, Reyhaneh Jabbarvand, and Jeffrey Nichols. Their essential feedback, encouragement, and guidance have played a pivotal role in shaping the depth and perspective of this work. Fond memories of Darko's support during my academic soul-searching in the first year still linger, a testament to his generosity and assistance provided by my academic mentors.

To my incredible teammates and friends – Carl Guo, Ali Zaidi, Aravind Sagar, Rizky Wellyanto, and Oliver Melvin – your camaraderie and support have transformed this academic pursuit into an immensely rewarding experience. Scott Hutchins, thank you for being a mentor who not only believed in my ideas but also encouraged me every step of the way. I also extend my gratitude to many others – mentors, mentees, University of Illinois Urbana-Champaign faculty, who have generously shared their wisdom.

I would like to thank my family, a source of perpetual support and positivity; even though I sometimes feel like they do not completely understand my research, which I think is pretty normal. They were always there, even when I was not there myself. My mother Saadet, "Annem", is my first teacher, mentor, and biggest fan. My father Mustafa, "Babam", has always believed in me and is the most virtuous man I know. My sister Esin, "Abram", is my very first role-model, never afraid to lead the way and always thinks what is best for me. She has been my mother in the States, and perhaps this whole thing would not be possible without her. My brother-in-law Efe, "Enişte", who always understands and supports my work, and makes the best eggs in the world. From the depths of my heart, thank you all.

Most importantly, I would like to thank Ahu Yolaç, my partner, my wife, my best friend. Your attentive listening, encouragement, and uplifting spirit, especially during moments of uncertainty, have brought profound meaning to this journey. You always shared and encouraged my excitement, and lifted me up when I felt down. With you, everything falls into place, and I cannot imagine my life without you.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The digital landscape we navigate daily is a tapestry of diverse mobile applications, each offering its own unique user interface (UI) and user experience (UX). Whether it's the convenience of a weather app, the productivity boost of a task manager, or the entertainment provided by a game, our daily interactions with these applications span across a multitude of platforms, including smartphones, tablets, smartwatches, and more. The quality of these digital experiences plays a pivotal role in determining user satisfaction, and in turn, has a direct impact on the success and profitability of the organizations responsible for creating and maintaining these applications. As a result, businesses and researchers alike have increasingly turned to data-driven methods to evaluate and enhance user interfaces. In pursuit of this goal, they rely on two primary tools: analytics and UX testing, often leveraging specialized platforms dedicated to improving digital experiences.

Analytics provides comprehensive, real-world usage data across user segments, enabling professionals to scrutinize critical user journeys, compute conversion metrics, and testing the effectiveness of design alternatives [2, 4, 58, 62, 67, 120, 133, 135]. These tools work by instrumenting an app's source code to capture clickstream information and perform action reporting. However, they necessitate access to an application's underlying codebase, rendering them impractical for collecting data from applications not under one's ownership. In contrast, UX testing tools offer the ability to gather nuanced behavioral and affective data through scripted tests [72, 77, 114, 146, 156]. Yet, their effectiveness hinges on elaborate test plans and flow design, particularly when conducting tests on external applications.

These limitations have spurred the development of an alternative approach, commonly known as *interaction mining*. Interaction mining is a popular technique for capturing design and interaction data while a mobile app is being used [33]. As a user interacts with an app, interaction mining systems capture the user's journey in the form of an *interaction trace*. This trace comprises sequences of UI screenshots, enhanced with structural composition and render-time properties (i.e., view hierarchies), alongside user actions (e.g., gestures) performed on these screens. Since its inception [33], interaction mining has catalyzed the creation of invaluable mobile UX datasets [13, 15, 34, 91, 121], empowering researchers to investigate UI patterns at scale [22, 53, 118, 141, 148, 168], train generative design models [9, 89, 94, 100, 102, 108, 174], and automate user actions [5, 97, 98, 99, 101, 144].

Interaction mining frameworks operate by utilizing physical devices or emulators equipped with a customized operating system designed to facilitate interaction capture. These frameworks shift the responsibility of trace capture from individual apps to the operating system

itself. Consequently, they enable the capture of interactions from any app, eliminating the need for integration prerequisites.

However, while the promise of interaction mining is immense, existing systems are not without challenges, primarily related to their reliance on OS-level interventions to facilitate comprehensive data capture. One of the primary challenges has been the complexity associated with developing, maintaining, and distributing a custom mobile operating system. Earlier approaches sought to mitigate this challenge by harnessing dedicated hardware with predefined specifications, capable of running (or emulating) the modified operating system. These systems pixel-streamed app content to end-users through a zero-configuration web interface [33]. While effective, this infrastructure's intricacy discouraged the release of its software or source code, limiting widespread adoption. Consequently, the current constraints in interaction mining architectures confine data collection efforts to supervised, in-laboratory settings, limiting the feasibility of launching such initiatives due to high upfront costs. The reliance on in-lab data collection results in out-of-date interaction mining repositories with the rapid updates of popular apps [126]. The limitations of existing architectures also highlight the need for collecting in-the-wild data to improve the ecological validity of interaction traces.

This dissertation introduces the **O**n-**D**evice **I**nteraction **M**ining (ODIM) framework, which stands as the first fully-functional interaction mining framework, facilitating real-world data capture from any Android app, directly on personal devices, without necessitating specialized hardware or operating system modifications. ODIM empowers researchers, designers, and industry practitioners to easily collect and leverage interaction mining data to build systems that benefit everyone.

1.1 CONTRIBUTIONS

This dissertation advances the field of interaction mining and envisions a sustainable ecosystem for interaction mining in the future by making the following contributions:

A Framework for Capturing In-the-Wild Interaction & Design Data: Addressing the limitations of existing interaction mining frameworks that restrict their applicability primarily to controlled lab environments, this dissertation introduces ODIM, an interaction mining framework requiring no integration or infrastructure setup, thereby democratizing the practice of interaction mining.

Principles for Interaction Mining System Design: While prior work has explored various techniques for implementing interaction mining systems, this dissertation delineates fundamental design principles that should underpin any interaction mining framework. These principles serve as a guidepost for the development of future interaction mining systems.

Enhancing Task Automation Through Interaction Data: In recognition of the central role of task automation in contemporary mobile devices, this dissertation introduces SAVANT, an unsupervised search system that automates the generation of task shortcuts for virtual assistants by mapping user tasks to relevant UI screens within apps, complementing existing programming-by-demonstration (PBD) systems. SAVANT serves as an example on how end-users can benefit from having direct access to interaction data through ODIM.

Insights for User Privacy Concerns in Mobile Apps: In a digital landscape where user privacy is of paramount concern, this dissertation sheds light on sensitive user information beyond personally identifiable data (PII). Drawing from a user study, it identifies additional types of sensitive content that users often encounter in digital user interfaces. These insights guide interaction mining systems like ODIM, enabling a privacy-first approach.

A Framework for Bridging Analytics and UX Test Data: Companies rely on two key data sources to evaluate digital experiences: behavioral analytics for quantitative insights and UX tests for qualitative feedback. Although both capture behavioral data, the techniques and tools for working with them are disparate. By applying interaction mining principles on these data sources, this dissertation bridges them to support richer workflows for optimizing digital design.

These contributions drive interaction mining toward a future that is more accessible, principled, and privacy-conscious, providing researchers, designers, and industry practitioners with innovative tools and methodologies to enhance digital experiences and safeguard user privacy effectively.

1.2 OVERVIEW

This dissertation is structured into seven chapters. These chapters encompass an exploration of the existing body of work in design and interaction mining, introduce the ODIM framework for democratizing interaction mining, and offer real-world use cases that are enabled by a more accessible interaction mining ecosystem.

Chapter 2 provides background on interaction mining by presenting the existing body of work, the datasets, and the applications in the field. In addition, it provides a review of the prior work on user privacy in the realm of mobile applications and an overview of existing evaluation methods for digital UIs.

Chapter 3 unveils the ODIM framework, which enables capturing in-the-wild interaction data from any Android app without any infrastructure or integration requirements. Additionally, we outline design principles that should serve as guidelines for future interaction mining systems.

Chapter 4 introduces SAVANT, an interaction data fueled system that generates task shortcuts for virtual assistants, elucidating how end-users stand to benefit from improved access to interaction data.

Chapter 5 presents the results of a study on the privacy concerns of Android users, conducted specifically for gaining a better understanding on how user perceive their interaction data.

Chapter 6 showcases how UX researchers can benefit from systems that leverage interaction mining principles. It provides an example system that bridges disparate data sources and provides opportunities to enhance the optimization workflows for digital design

Chapter 7 encapsulates considerations on how ODIM should be deployed in the real-world, while also providing the vision for a democratized and sustainable interaction mining ecosystem.

1.3 STATEMENT ON PRIOR PUBLICATIONS

The SAVANT system was previously published and presented at the ACM User Interface Software and Technology (UIST) Conference in 2021 [5]. Some of the materials included in the remainder of the dissertation went through the peer-review process within the top HCI venues, at least once. This dissertation presents the latest versions of these works, guided by the comments and the suggestions of generous reviewers.

CHAPTER 2: RELATED WORK

This chapter aims to provide a foundational understanding of the key concepts and existing work associated with interaction mining. We navigate through prior research and existing technologies in order to lay the groundwork for the relevant core concepts. We first provide a brief exploration of interaction mining, underlining widely used datasets and its applications. As interaction mining deals with personal data, we then provide an overview of user privacy in mobile apps and discuss the developments and research on the topic. Additionally, we explore existing methods for evaluating digital UIs which, similar to interaction mining, deal with behavioral data.

2.1 INTERACTION MINING

The realm of mining design patterns from digital user interfaces has been extensively explored [52], leading to systems designed for extracting website design information and facilitating data-driven web design tools [84, 85]. However, with the proliferation of mobile devices, the landscape of interaction mining evolved to accommodate the unique characteristics of these platforms [92]. Notably, the introduction of *ERICA* marked a significant milestone in the domain, kick-starting the field of mobile interaction mining [33].

Interaction mining is a technique for capturing design and interaction data while an app is being used [33]. As users interact with an app, an interaction mining system records their journey, encompassing sequences of UI screenshots, the structural composition, render-time properties of these screens, and user actions. These recorded user journeys are encapsulated as *interaction traces*.

Interaction mining frameworks function through physical devices or emulators equipped with a custom operating system, specially tailored to facilitate interaction capture. This approach eliminates the necessity for app-specific integration and enables the collection of interaction data from any app. By installing these frameworks on devices and pixel streaming app screens through a web interface, they offer a versatile means of capturing interactions within digital UIs.

Interaction mining serves three primary user groups: UX researchers, machine learning (ML) practitioners, and end-users. For UX researchers, it provides valuable insights into design patterns, inspiration for UI creation, and performance evaluation. ML practitioners leverage interaction data to train models for predicting user interactions, offer personalized suggestions, and enhance design comprehension. End-users benefit from improved digital

experiences, including personalized features, automation tools, and overall enhanced UX.

2.1.1 Datasets

The advent of interaction mining frameworks has spurred the creation of several datasets, each offering valuable design and interaction data resources. Notable examples include:

***Webzeitgeist* [85]:** As one of the first large-scale design repositories, *Webzeitgeist* comprises over 100k web pages and 100 million design elements. This dataset was collected to pioneer design mining for the Web, as traditional data mining techniques did not include render-time properties for web pages.

***Rico* [34]:** This extensive repository encompasses over 66k UI screens from over 9.3k apps spanning 27 categories, serving as the largest collection of mobile app designs. *Rico* exposes various design properties and supports five data-driven applications, including design search, UI layout generation, UI code generation, user interaction modeling, and user perception prediction.

***ReDraw* [121]:** To support the transformation of graphical user interface (GUI) mockups into code, researchers have curated the *ReDraw* dataset. This dataset plays a pivotal role in training deep convolutional neural networks (CNNs) for the classification of GUI components. It encompasses over 14k screens and more than 190k GUI components, serving as a valuable resource for UI design automation.

***Enrico* [91]:** An enhancement of *Rico*, the *Enrico* dataset aims to delve deeper into UI functionality through layout design categorization. It comprises human-supervised and high-quality data for 1460 UIs and 20 design topics.

***VINS* [13]:** Designed to facilitate the training of models for detecting similar UIs, the *VINS* dataset offers an annotated collection of UI screens at two distinct design stages: abstract wireframes and high-fidelity fully designed interfaces. With a total of 4800 images encompassing UI designs (257 wireframes and 4543 screens), it provides a rich resource for similarity detection across diverse UI screens.

***MoTIF* [15]:** Catering to the domain of vision-language navigation, the *MoTIF* dataset captures interaction traces for over 6.1k natural language tasks across 125 Android apps.

This dataset serves as a valuable asset for research into vision-language interaction and navigation systems.

***WebUI* [167]:** As a dataset of 400k web UIs associated with their relevant metadata, the *WebUI* dataset was used to incorporate semantic information in web pages to build more performant mobile UIs understanding models.

While these datasets have significantly advanced interaction mining research, they are not without limitations. Existing interaction mining systems with specialized infrastructure often miss data from certain apps. Additionally, as apps frequently undergo updates, existing datasets can quickly become outdated. Furthermore, concerns about the ecological validity of interaction mining data persist, with some datasets relying on web interfaces or inorganic executions of mobile apps to collect data, potentially impacting the authenticity of the captured interactions in unfamiliar mobile environments.

2.1.2 Applications

The proliferation of design and interaction datasets has catalyzed the development of an array of data-driven systems and applications. Researchers and practitioners have harnessed the wealth of data obtained through interaction mining to create innovative solutions that span various domains:

Researchers have developed systems that can autonomously extract semantic information from UI components [108]. These annotations enhance the understanding of UI structures and functionalities, aiding in various downstream tasks. Design and interaction data have been instrumental in automating the generation of UI designs. Researchers have developed systems that can analyze interaction mining data to propose novel UI layouts and design elements [9, 94, 149]. Moreover, interaction traces serve as a valuable resource for retrieving existing UI designs based on specific criteria, streamlining the design process [74, 121]. Interaction mining has also paved the way for transforming UI screens into human-readable natural language descriptions [100, 159], augmenting accessibility features within mobile apps [176]. Researchers have also leveraged interaction traces to automate UI testing and evaluation [35, 104].

Interaction mining has presented rich opportunities for the development of task automation systems. Various programming-by-demonstration systems, including *PUMICE* [99], *KITE* [97], *SUGILITE* [98], and *VASTA* [144], have capitalized on interaction mining data. These systems operate on a supervised learning paradigm, where users demonstrate specific tasks within apps. Subsequently, the systems generalize these interactions and can

autonomously execute similar tasks with varying parameters. Other systems have also been devised to automatically navigate users through app interfaces by analyzing interaction traces to autofill relevant information on task-relevant screens, enhancing user convenience and efficiency [5, 101]. Emerging research has explored the application of task automation to create functional app prototypes and perform dynamic analysis [65, 83].

Finally, interactive ML systems [3, 54] harnessed interaction mining data to build systems that cater to diverse domains. These applications range from aiding in the creation of musical instruments [57] to automating complex data science projects [160], and to interactive correction for grounding [95], enhancing the interaction between users and ML systems.

In essence, interaction mining has fueled a wide spectrum of applications. From automating UI design to enhancing accessibility and enabling task automation, the potential of interaction mining is vast, offering valuable solutions to researchers, developers, and end-users across diverse domains.

2.2 USER PRIVACY IN MOBILE APPS

Interaction mining data comprises user journeys within a mobile application. Since the data is captured while apps are being used, interaction traces inevitably contain personal data. In the rapidly evolving landscape of mobile applications, users already face critical decisions regarding the extent to which they are willing to share their data.

From the user's perspective, this decision-making process occurs at two primary levels: permissions and settings. Upon the installation of an Android app, it commences with zero permissions. Subsequently, when the app requires access to certain functionalities (such as location data, camera usage, or contact information) that could potentially impinge on the user's privacy, it solicits permission. Existing research in this realm has concentrated on assisting users in managing their permission preferences [106], automating permission decisions to bolster privacy protection [163], and elucidating which components of an app utilize specific permissions [103], thereby facilitating informed privacy-related decisions.

Concurrently, privacy settings, which govern the sharing of information, are accessible through the app itself and are accompanied by default values that determine the level of shared data. Notably, Chen and colleagues introduced *Hound*, a system capable of autonomously identifying concealed privacy settings using a classifier trained through semantics-based UI tracing [25].

Moreover, there have been research endeavors aimed at aiding app developers in recognizing potential privacy concerns during the app development process. *UIPicker*, for instance, is designed to identify potentially sensitive user inputs, facilitating security analyzes of mo-

bile apps [124]. Similarly, *Coconut* offers feedback to developers regarding potential privacy issues, encouraging heightened awareness of privacy considerations throughout the app development lifecycle [96]. While app permissions serve as an essential initial layer of defense in safeguarding user privacy, it is imperative to acknowledge that they may not offer comprehensive protection.

2.2.1 Visual Privacy

Detecting privacy risks within visual media has been extensively studied. Recent research has focused on the automatic detection of privacy-sensitive information in images using classifiers [130, 173], neural networks [153, 172], and crowdsourcing [82]. Datasets containing visual privacy concerns have been compiled to support ongoing research in detecting private information within images [64, 130, 131]. Although digital UIs are primarily visual, they convey more than mere pixels, making the detection of privacy-sensitive elements a challenging endeavor.

2.2.2 Designing for Privacy

Designers hold a pivotal role in shaping the treatment of user privacy within interactive systems. The design process fundamentally determines the extent to which user information is required and how this necessity is effectively conveyed to users. As a response to this critical role, researchers have dedicated efforts to offer guidance for privacy-conscious design practices.

Lederer et al., for instance, shed light on potential privacy pitfalls that designers should remain vigilant about when crafting interactive systems [88]. Building upon these insights, recent research by Hong et al. introduced a comprehensive design rationale aimed at enhancing privacy considerations within Android user interfaces [71].

Furthermore, the evolving concept of "privacy by design" has garnered significant attention. This approach advocates for the integration of privacy concerns into the initial phases of product design rather than treating privacy as an afterthought [19, 122, 143]. Consequently, researchers have explored methods and strategies to assist designers and HCI practitioners in incorporating privacy considerations into their work [164].

2.2.3 Perceptions and Expectations

Prior research has explored user perceptions of privacy, particularly in the context of social media and interactive systems [138]. These studies investigate how socioeconomic factors relate to user privacy concerns [137] and what users expect from interactive systems regarding privacy [105]. Additionally, researchers have proposed methods for measuring user privacy expectations and comparing them with the reality of how systems handle privacy [113]. Conventional definitions of privacy and private information often fall short in capturing what users may consider private, highlighting the importance of aligning design and functionality with user perceptions and expectations.

2.3 EVALUATING DIGITAL UIS

To evaluate how digital products perform, companies use many professional systems to collect user interaction data. They examine two major data sources for this purpose: analytics and UX assessments. Behavioral analytics provides large-scale, coarse-grained data which is often used to identify *how much* a certain behavior occurs within an experience. UX assessments yield smaller-scale affective data focusing on *why* a behavior occurs. Both methods are directed at understanding user behavior, which interaction mining can provide data for.

2.3.1 Behavioral Analytics

Early works in web analytics outlined the significance of gaining insights into how customers interact with products [26, 158]. Over time, the industry has widely embraced the value of behavioral analytics data, recognizing it as pivotal in understanding customer behavior within digital products. A multitude of services offer comprehensive behavioral analytics data. These platforms provide information about user interactions, including screen navigation, button clicks, device specifications, session duration, self-reported opinions through surveys, and more. Notable platforms such as Google Analytics [62], Adobe Experience Cloud [2], Flurry [58], Heap Analytics [67], Mixpanel [120], Amplitude [4], Qualtrics [135], and Pendo Analytics [133] empower businesses to track user activity, identify drop-off points, and measure key performance indicators (KPIs).

Transforming raw usage data into meaningful and actionable insights has been a subject of considerable research. Studies have explored various approaches, including the evaluation of different flow chart shapes in web analytics using eye-tracking methods [87]. Additionally,

novel visualization techniques have been proposed to represent user navigation in web pages, offering diverse aggregate metrics to enhance decision-making [69, 86].

Recent advancements have introduced session recordings and replays as part of behavioral analytics services. These recordings allow businesses to gain deeper insights into user interactions by programmatically replaying user sessions or providing video recordings from the user's perspective. Services like FullStory [60], Contentsquare [29], and Quantum Metric [117] offer these capabilities, providing a more detailed understanding of user behavior.

Although these advancements provide a deeper insight into what users are doing, existing behavioral analytics solutions are still limited to only tracking the organic behavior, without having access to user intent. These tools are still largely provide quantitative data, lacking a coherent link with qualitative information on the human context.

2.3.2 UX Testing

Commercial platforms like UserTesting [77], UserZoom [156], Maze [114], Hotjar [72], and Sprig [146] have become essential tools for organizations seeking to gain valuable user insights through UX testing. These platforms facilitate tests in which representative users are tasked with completing typical actions within a digital product while their journeys and feedback are recorded. The collected data aids in improving UX design, identifying potential new features, and assessing overall user satisfaction. Typically, these tools provide visual data in the form of screenshots or screen recordings along with think-aloud data through which users share their mental models.

Beyond commercial platforms, innovative research systems have augmented traditional UX testing methodologies. Examples include *WebQuilt*, which introduced remote UX testing capabilities and data augmentation to visualize participant paths [70, 161, 162]. *UX-Handle* enhances the experience of user researchers by providing a semi-automatic web-based system for the analysis of UX testing data [11]. *InteracDiff* offers UX visualization tools to enhance the interpretation of collected data [47]. Researchers have also investigated the integration of machine learning systems into the workflows of UX designers [171].

Researchers have explored automating the inclusion of additional data sources during UX tests. For instance, Ferre et al. proposed leveraging Google Analytics to collect quantitative metadata during usability evaluations, enriching insights [56]. Jeong et al. introduced a framework for collecting interaction logs during asynchronous UX assessment tests, subsequently generating visualizations based on these logs [79]. Buono et al. developed flow visualization techniques that leverage recorded user interaction data during UX tests to identify "usability smells" and areas of improvement [14].

CHAPTER 3: ON-DEVICE INTERACTION MINING

This chapter introduces **O**n-**D**evice **I**nteraction **M**ining (ODIM), a framework designed to make interaction mining accessible to everyone. ODIM empowers users to capture in-the-wild interaction data directly on their Android devices, entirely eliminating the need for custom mobile operating systems or specialized hardware (Figure 3.1). By simply downloading the ODIM app, users can record interaction traces as they naturally interact with their favorite apps. ODIM enables efficient workflows that benefit UX researchers, machine learning practitioners, and end-users alike, democratizing interaction mining.

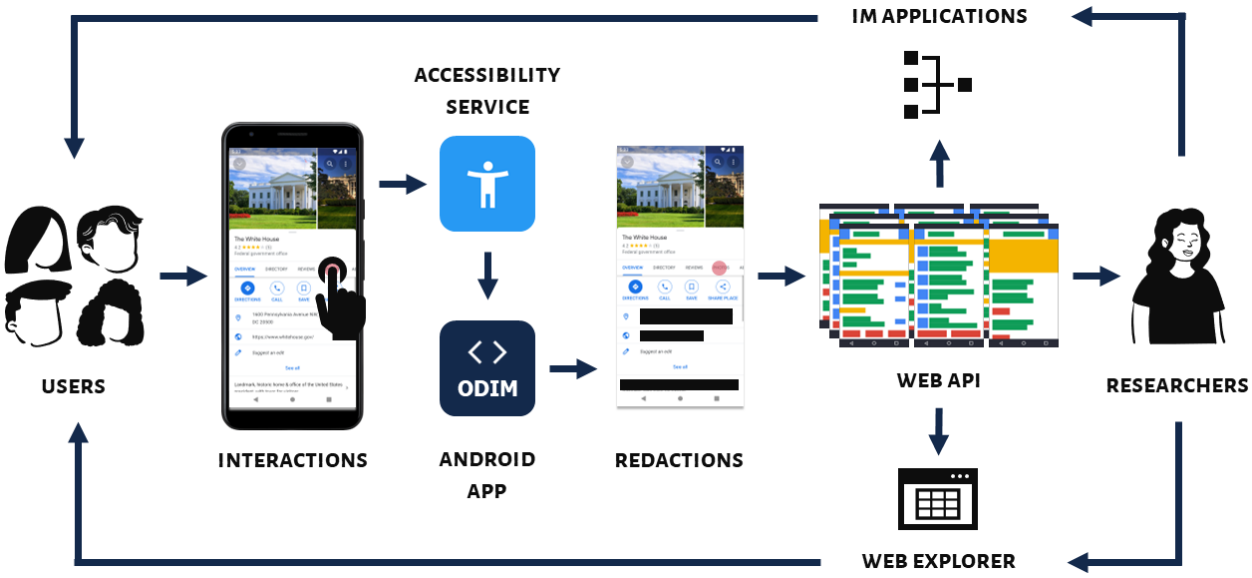

Figure 3.1: ODIM *is the first fully-functional interaction mining framework that enables in-the-wild capture of any Android app without requiring a custom OS or hardware.*

Users can collect interaction data simply by downloading the ODIM app and toggling its capture feature. As users interact with various apps, ODIM records these interactions in the background. Users maintain full control over their data; they have the option to review and redact any sensitive information before deciding to upload a trace to the cloud. ODIM comprises an Android accessibility service for recording interaction traces, an Android app for managing these traces, a web application for exploring interaction data, and a web API for allowing programmatic access to collected data. The core principles driving ODIM's design are rooted in accessibility, simplicity, and privacy. ODIM allows collecting high-quality, ecologically valid interaction data while maintaining user privacy and control.

In the sections that follow, we first explore the key design principles that guide ODIM's implementation. We then walk through how the ODIM system is designed, outlining each

of its components. We also provide insights into the evaluation of ODIM, underpinned by a live repository of interaction traces contributed by users from a broad spectrum of apps. Additionally, we showcase the various workflows that ODIM facilitates, catering to the distinct needs of users across different domains. ODIM presents a leap towards democratizing interaction mining, enhancing digital experiences for all, and catalyzing innovation in the fields of UI research, machine learning, and app development.

3.1 DESIGN PRINCIPLES FOR INTERACTION MINING

To democratize interaction mining, ODIM embodies five fundamental design principles that collectively underpin to its efficacy and versatility in enabling interaction data capture:

Native Interaction Capture: To ensure the quality and ecological validity of interaction data, ODIM captures interactions as they naturally occur on Android devices. This is achieved through an Android accessibility service, integrated within the ODIM app. This service operates unobtrusively in the background, allowing for the capture of interactions without imposing any notable performance overhead on their devices. By capturing on-device interactions, ODIM enhances the value of collected data for both UX researchers and ML practitioners, ensuring that it remains ecologically valid.

Zero Integration and Infrastructure Setup: Overcoming the challenge of requiring specialized infrastructure or code integration is a key aspect of ODIM's design philosophy. While behavioral analytics platforms like Google Analytics [62] require some code instrumentation but no setup, systems like ERICA [33] demand infrastructure setup but do not require code changes. ODIM removes these barriers by imposing no integration or infrastructure setup requirements; users simply need to download an app to commence recording interaction traces. Once recorded, these traces are uploaded to the cloud and made publicly accessible through the ODIM explorer. ODIM's minimal reliance on external components ensures that the system remains agile and adaptable, with rare updates required when backwards-incompatible alterations are made to the Android accessibility service.

Crowdsourcing Simplicity: ODIM is intentionally designed to facilitate crowdsourcing efforts. Researchers can collect interaction data by directing participants to download the ODIM app. Researchers can monitor participant progress by reviewing interaction traces via the web explorer. Scaling up and adding new participants to ongoing projects incurs minimal overhead. After data collection, researchers can download organized interaction

traces. Users interact with ODIM through an extensible web explorer that allows for potential advanced features like user accounts, workspace creation, and worker tracking, simplifying the crowdsourcing of interaction data further.

Privacy-First Approach: Given the inherent need to capture natural interactions, safeguarding user privacy becomes a paramount concern in interaction mining systems. Without built-in mechanisms to protect user privacy, many recorded traces would potentially include sensitive personal information. ODIM addresses this challenge by providing two essential features: a start/stop toggle that allows users to pause recording when entering sensitive information, and a redaction interface that permits users to review traces before uploading and redact any screen regions they consider sensitive. These redactions automatically extend to the recorded programmatic screen representations, effectively eliminating any data contained within the redacted regions from the corresponding view hierarchies.

Versatile Data Access: To cater to diverse use cases, ODIM ensures that the interaction data it collects is accessible and adaptable for various purposes. Practitioners may require interaction data for research, UX testing, session inspection, or large-scale training data. ODIM's codebase is designed to accommodate all these access patterns. By making a simple adjustment to the ODIM Web API, users can collect a private corpus of interaction traces instead of uploading to the public cloud. Additionally, the ODIM Web API supports both serial and batch access to trace data, providing versatility in data retrieval methods.

3.2 SYSTEM DESIGN

ODIM comprises four components (Figure 3.1):

- An **Accessibility Service** responsible for capturing screens and user interactions on the user's device, ensuring the comprehensive collection of interaction data.
- An **Android Application** designed to efficiently organize the collected data into traces, equips users with the capability to review, edit, and upload their interactions.
- An explorer **Web Application** empowering users to delve into the uploaded traces, fostering deeper insights and understanding.
- A **Web API** facilitates streamlined access to the collected data, enabling a range of interaction data-driven applications to harness its insights.

3.2.1 Accessibility Service

The Android platform's `AccessibilityService` API is designed to support the development of accessibility services that enhance the usability of Android devices and applications, particularly for users with disabilities [39]. Once enabled, accessibility services actively monitor user interface actions and generate `AccessibilityEvents` in response to significant interactions [37]. These `AccessibilityEvents` furnish valuable insights into the UI, encompassing details about the involved UI elements, their content, the current UI state, and any interactions that transpired. Key to accessing this information is the `AccessibilityNodeInfo`, which facilitates the extraction of structural and visual particulars from these events [38]. The structural composition and render-time attributes of a screen are encapsulated in a tree formed by `AccessibilityNodeInfo` nodes. This tree commences with the root node, serving as the entry point for extracting screen representations.

In the context of ODIM, an Android accessibility service assumes a pivotal role in detecting user actions. This service also creates an overlay that captures user touches on the device screen. These two data streams merge to form an **event** – the fundamental building block of an interaction trace. An event combines both the visual and structural representations of a UI screen with any concurrent user interactions (*gestures*). When the accessibility service is active, ODIM triggers a transparent overlay that spans the entirety of the device screen, monitoring user touches. Upon detecting a touch event, ODIM captures a screenshot and retrieves the root `AccessibilityNodeInfo` of the active screen. Through a recursive preorder depth-first traversal of the screen's node tree, ODIM gathers the visual properties of all UI elements. This gathered information is then structured into a JSON-formatted string, referred to as the **view hierarchy**.

To ensure the efficient transformation of the `AccessibilityNodeInfo` tree into a JSON view hierarchy string, ODIM employs a streaming approach facilitated by the Gson library [63]. Initially, we utilized a recursive process to extract node properties into a tree of maps, which was subsequently deserialized into a JSON string. However, this method encountered memory constraints when dealing with extensive view hierarchy trees, as the entire map tree had to be loaded into memory for conversion. The current streaming approach improves parsing efficiency and robustness by writing node properties into a stream token by token during the recursive traversal of the `AccessibilityNodeInfo` tree. ODIM maintains the latest screenshot and view hierarchy data from a user touch as global variables.

Simultaneously, while the overlay monitors user touches, the accessibility service listens for `AccessibilityEvent`s. Upon detecting an event, ODIM performs an in-depth analysis of its properties to discern whether it signifies a user interaction. This analysis encompasses

the identification of the interaction type, the specific UI element implicated, and the package name of the app where the event unfolded. `AccessibilityEvents` also encapsulate gesture information, including a pair of (`x`, `y`) coordinates for *touch* interactions and a quartet of (`x`, `y`, `scrollDx`, `scrollDy`) for *scroll* motions. Gesture coordinates are calculated based at the center of the UI element involved in the interaction. When ODIM identifies a user interaction within an `AccessibilityEvent`, it integrates the event data with the most recent stored screenshot and view hierarchy, and generates a complete event.

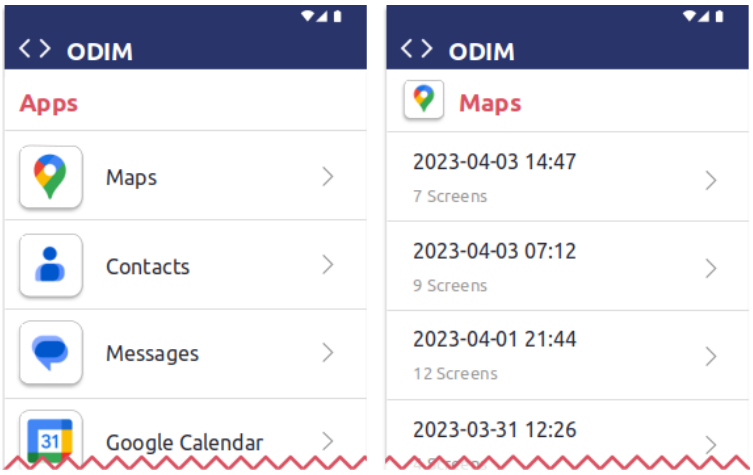

Figure 3.2: *The* ODIM *App groups recorded interaction traces by app.*

3.2.2 Android App

ODIM consolidates interaction data into *traces*. These traces consist of sequential events, including screenshots, user gestures, and view hierarchies for each UI screen. They represent continuous interactions within an app, with a new trace initiated when users launch an application and concluding when they tap either the "Home" or "Back" button, or simply when they return to the home screen. When the accessibility service generates a new event, it is incorporated into the ongoing trace or triggers the initiation of a fresh trace. Traces are organized by the associated app and presented in a reverse-chronological order (Figure 3.2). Users can review screens within each trace by selecting the trace of interest. The trace view offers a sequence of event screenshot thumbnails, while clicking on these thumbnails reveals the complete screen view, accentuated with red box borders that delineate the UI elements derived from the recorded view hierarchy. Positioned at the bottom of the trace view, a button invites users to upload the complete trace data to the connected database, preceded by a prompt soliciting a trace description for contextualization.

Recognizing that ODIM possesses the capability to capture intricate user data, it is prudent to address potential concerns regarding data sharing, particularly when the app in question may contain personally identifiable information. In response to this, the ODIM application introduces a redaction interface within the full screen view. This interface empowers users to review and redact sensitive portions of the recorded content. The redaction interface is designed with red box highlights overlaid on privacy-sensitive UI elements, accompanied by a floating action button that toggles between "draw" or "erase" modes, as well as another button for confirming and "saving" (Figure 3.3) the redaction changes.

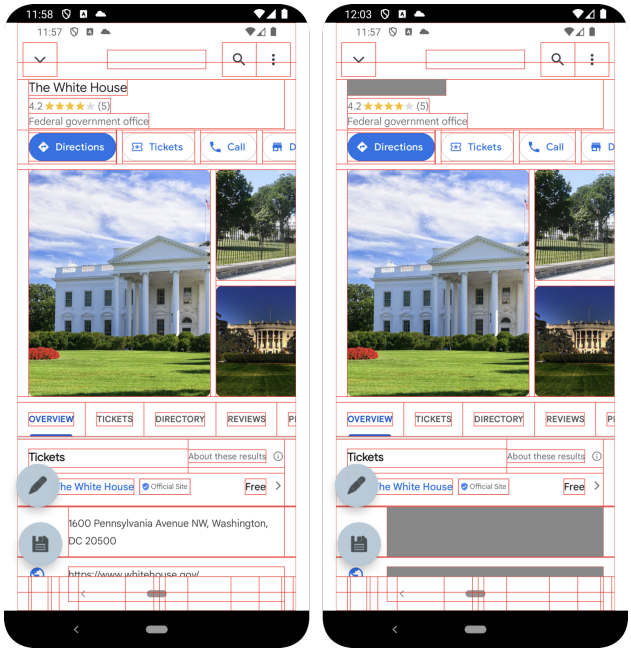

Figure 3.3: *The* ODIM *app provides users with a redaction interface to review traces and redact screen regions that contain private information.*

To redact, users simply touch and drag over the privacy-sensitive area, creating a protective and opaque gray rectangle. Each redaction prompts users to provide a descriptive label, ensuring transparency and context. In the event of an accidental redaction, the "erase" mode permits users to rectify the situation by touching the affected area. Upon completion, users can switch back to "draw" mode and confirm the redaction changes by clicking the "save" button. The redacted areas within the screenshot bitmap are marked with black boxes, while the redacted UI elements within the view hierarchy undergo a transformation that replaces fields potentially containing sensitive information (e.g., text or content description) with "REDACTED". Importantly, this entire redaction process transpires exclusively on users' mobile devices, thereby affording users full control over the data they choose to display or conceal before any uploading occurs. The motivation behind and design choices for this

redaction interface are described more in detail in Chapter 5.

3.2.3 Web Application

The ODIM Explorer[1], provides easy access to the collection of recorded interaction traces. The homepage of this explorer presents users with a grid view, showcasing apps for which interaction traces have been captured (Figure 3.4). Upon selecting an app card, users are taken to a paginated list of these recorded traces (Figure 3.5). These traces are displayed as horizontal sequences of screens, with user gestures represented as orange circles. These circles convey *touch* interactions, including their precise (`x`, `y`) coordinates, as well as *scrolling* actions—depicted by repeated orange circles featuring gradually decreasing radii that elucidate `scrollDx` and `scrollDy` values. In accordance with data privacy norms, any redacted UI screens are obscured with black boxes, accompanied by user-provided descriptions located at the bottom of each trace, affording context to the viewer.

The ODIM Explorer further enhances the user experience with a search feature, enabling users to efficiently filter and search for traces by app names. Once a user uploads a trace and receives confirmation, it immediately becomes accessible to a wider audience via the web explorer. Additionally, users are able to download the raw interaction data, thereby equipping them with the flexibility and control they require for further analysis or use.

3.2.4 Web API

The ODIM Web API is crucial for storing and making collected interaction data accessible. The ODIM app communicates with the API through HTTP requests. When it sends a POST request with local trace data, the API saves the metadata in a dedicated MongoDB instance in the cloud. Simultaneously, it uploads the corresponding screenshots and view hierarchies to a designated AWS S3 bucket.

When a trace is uploaded, the ODIM app provides the package name for the app it corresponds to, along with the interaction data. If a trace is from an app not in the database, the ODIM API dynamically scrapes the Google Play Store using the app's package name. This fetches additional details such as descriptions, ratings, and promotional screens, which are then used to create a new trace collection for the app.

The ODIM Explorer serves as a user-friendly interface for accessing the stored interaction data and allows users to interactively explore it. Additionally, it serves as a proof-of-concept for potential interaction data-driven applications. In the future, the ODIM Web API can

[1]The ODIM Explorer is available at `https://odim-explorer.vercel.app/`

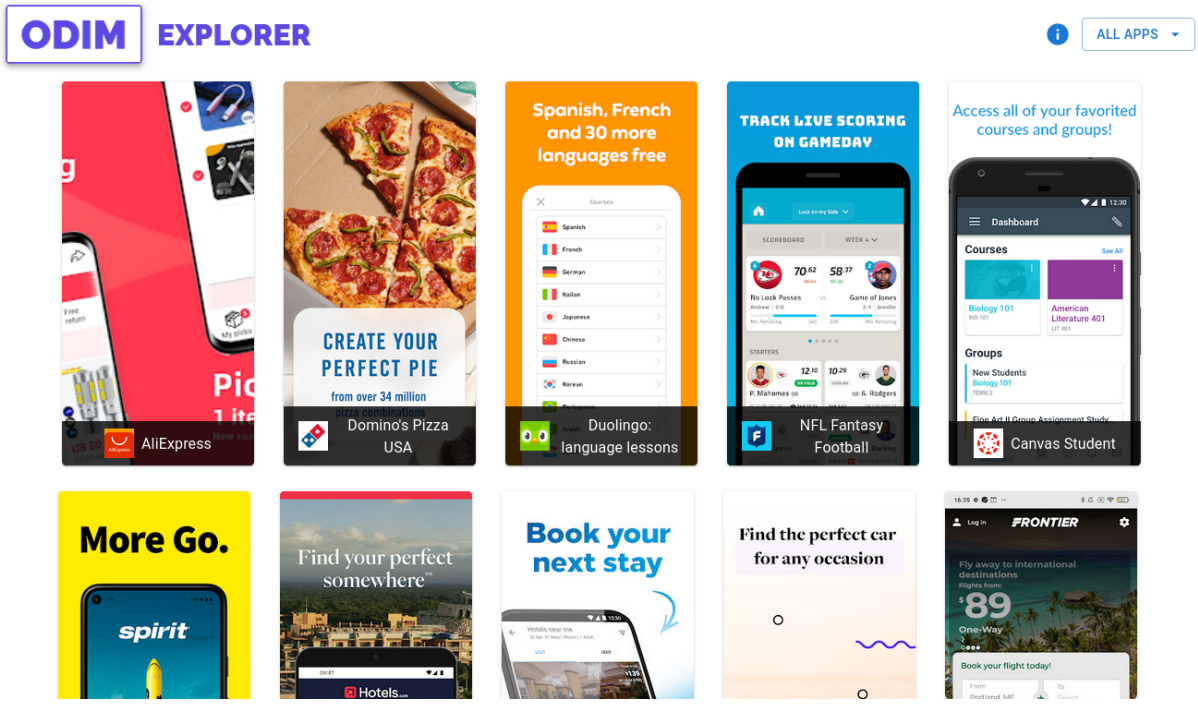

Figure 3.4: *The homepage of the* ODIM *Explorer presents a grid view of apps from which traces have been captured.*

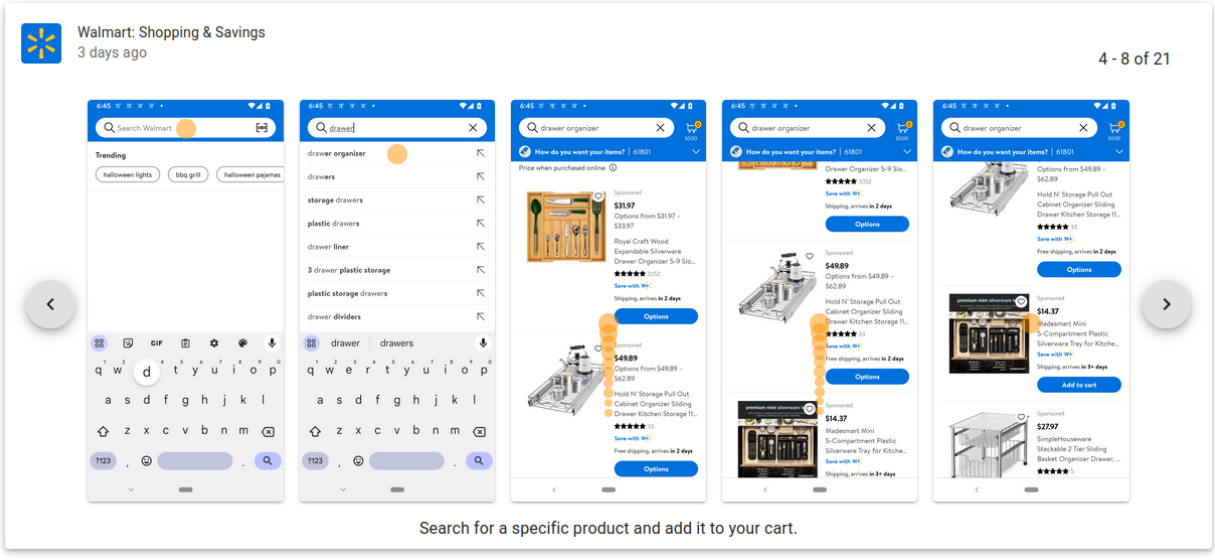

Figure 3.5: *The detail view of the* ODIM *Explorer showcases interaction traces recorded with the* ODIM *app.*

be enhanced with advanced search capabilities, allowing users to search for specific screen content or recurring UI concepts within traces. This adaptability makes the ODIM Web API a valuable tool for interaction data analysis.

3.3 EVALUATION

To assess the usability and effectiveness of ODIM, we conducted a pilot data collection to establish the initial public repository. Ten users participated in this data collection effort. We selected 32 app categories from the Google Play Store, excluding categories like games, library demos, and watch faces that were unlikely to yield substantial interactions. From each category, we retrieved the top 20 apps, resulting in a total of 640 target apps. For each app, we generated five target tasks by using ChatGPT [129] with the prompt "Give a concise list of 5 user tasks for the `[APP NAME]` Android app (`[APP PACKAGE NAME]`)". This process yielded a set of 3.2k target traces. Our users were then instructed to record traces from this list using sock puppet accounts. In instances where task descriptions were unworkable or poorly defined (for example, due to the necessity of pre-existing account data), the task was skipped. In just six days, our users successfully collected over 1,000 labeled traces from more than 500 apps, creating a valuable initial dataset that is now accessible via the ODIM Explorer 3.2.3. Unlike previous interaction mining repositories, anyone can contribute new traces to ODIM simply by downloading the ODIM Android Package (APK).

During this process, we encountered a single major usability issue with ODIM. Some users reported crashes in the accessibility service while interacting with certain apps. These crashes stemmed from an early design decision to maintain the entire view hierarchy in memory during capture. To mitigate this issue, we transitioned to a streaming memory model using the Gson library, as detailed in Section 3.2.1, and implemented persistent storage within the app to prevent data loss.

For comparison, the prior state-of-the-art interaction mining system, ERICA[33], invested six months in collecting interaction traces from 9.3k to form the *Rico* dataset [34]. They relied on paid participants recruited from Upwork [155] for this task. This initial comparison underscores the efficiency, reduced setup complexity, cost-effectiveness, scalability, and sustainability of ODIM. It is worth noting that ODIM's interaction traces are task-oriented and include descriptions, augmenting the value of the collected interaction data when compared to the *Rico* dataset.

3.4 WORKFLOWS

ODIM offers the capability to directly capture in-situ data on Android devices without the requirement of complex setups. Its design ensures that interaction mining is accessible to a broad spectrum of users, facilitating efficient workflows for collecting interaction data. To illustrate its adaptability, we present three distinct workflows, each tailored to empower individuals across diverse domains: UX researchers, ML practitioners, and interactive ML end-users.

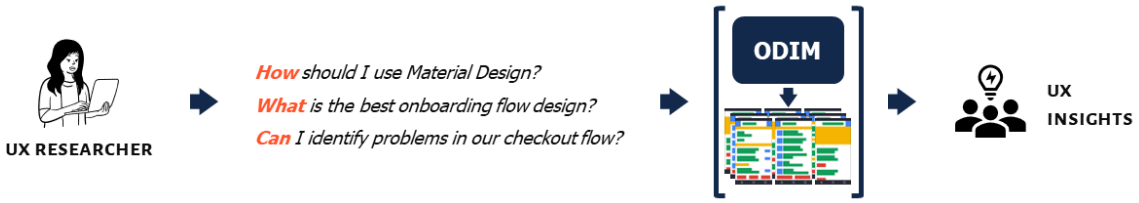

Figure 3.6: ODIM *gives UX researchers easy access to interaction data, enabling them to understand design systems, find inspiration for designing new UIs, and analyze the performance of existing UIs.*

UX Researchers: With ODIM, UX researchers gain access to valuable interaction data, empowering them to study design principles at-scale, seek inspiration for innovative UIs, and evaluate the effectiveness of existing ones (Figure 3.6). Let's consider the example of Olivia, a UX researcher on a mobile team at an e-commerce company. Olivia is tasked with enhancing the checkout process of their mobile app. She selects three apps for comparison and enlists the help of five UX test participants. These participants utilize the ODIM app, with the accessibility service enabled, to record interaction traces. Olivia instructs them to start recording from the home screen of each app. The participants record interaction traces for the checkout flows. After recording, they use the ODIM app to redact any private UI elements before uploading the traces. Olivia collects and analyzes these 15 traces to identify user-friendly UI features, highlight potential pain points, and ascertain preferred interaction elements. She then shares her findings at the upcoming UX design meeting, contributing to her team's informed design decisions.

ML Practitioners: ODIM empowers ML practitioners by providing them with interaction data for training ML models for predicting user interactions, assisting designers, and enhancing design comprehension (Figure 3.7). Take, for example, Nick, an ML practitioner working on the development of a mobile task automation tool. Nick's wants to train a model capable

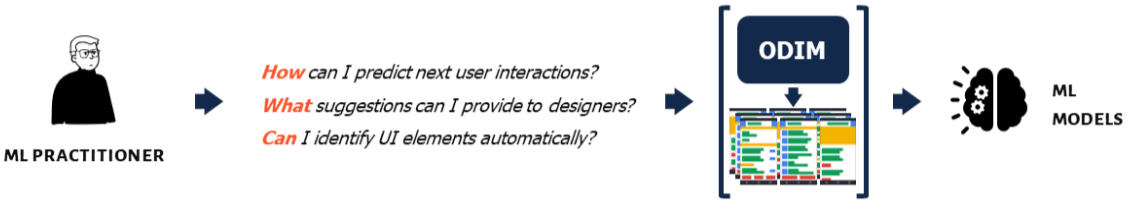

Figure 3.7: *With* ODIM*, ML practitioners can collect interaction data for training ML models. These models help other researchers, designers, and end-users.*

of autonomously deducing user actions for common tasks. To achieve this, he runs a study to identify prevalent mobile tasks, initially selecting a set of 10 such tasks. Nick recruits a group of 100 participants through a crowdsourcing platform, guiding them to download and configure ODIM. These participants continue their regular device usage for a period of three days before reviewing the recorded interaction traces. Nick provides them with instructions to upload traces corresponding to the designated 10 tasks. Participants choose relevant traces, redact sensitive information as necessary, and securely upload the interaction data. Nick leverages this raw interaction data, which encompasses redacted UI screens, gestures, and structural representations, to effectively train his model. Subsequently, he releases his task automation tool, `Task-o-Tron 3000`.

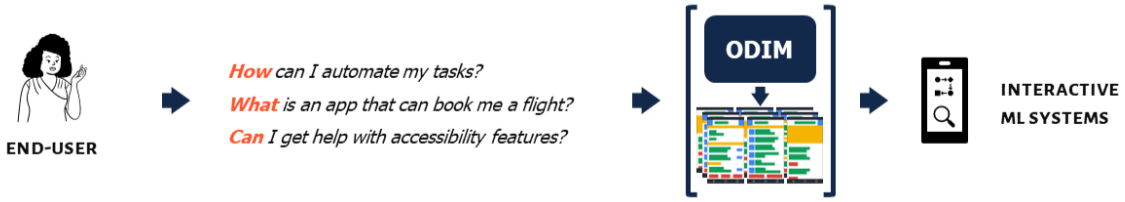

Figure 3.8: *End-users benefit from* ODIM *as they are presented with new features, automation capabilities, and better user experience.*

Interactive ML End-Users: End-users also enjoy the advantages of ODIM's interaction data, providing them with personalized features, automation tools, and enriched digital experiences (Figure 3.8). Let us take Patricia as an example –a regular end-user who downloads `Task-o-Tron 5000`, an updated iteration of Nick's task automation tool, now equipped with interactive ML capabilities. After a week of using the tool, Patricia decides to enable automation for a task that isn't officially supported within `Task-o-Tron 5000`. She leverages ODIM to capture specific interaction traces tailored to her unique task. Patricia then directs `Task-o-Tron 5000` to integrate the locally captured interaction data from ODIM, thereby enhancing the tool's task automation capabilities to accommodate her new task. Later on,

Patricia stumbles upon a public repository of ODIM-recorded traces, further elevating the functionality of Nick's tool. To contribute to this growing resource, users like Patricia share their own redacted interaction traces, enabling them to access an even broader range of automation features. In Patricia's case, she gladly contributes her redacted interaction traces to enrich the repository.

CHAPTER 4: TASK SHORTCUTS FOR VIRTUAL ASSISTANTS

In a rapidly evolving digital landscape, virtual assistants like Google Assistant, Siri, and Alexa have become indispensable companions in our daily lives, expanding the boundaries of what users can accomplish through voice and text-based interactions. These assistants have evolved to automate a multitude of everyday tasks, providing users with information, scheduling appointments, and even controlling smart devices in their homes. However, despite their growing repertoire of skills, their capabilities still lag behind the vast landscape of functionalities offered by the multitude of mobile apps.

Recognizing this limitation, virtual assistant platforms have started interfacing with mobile apps to extend their capabilities. Through mechanisms such as *Android App Actions* [41] and *Siri Shortcuts* [50], developers are empowered to programmatically expose the functionalities of their apps to these virtual assistants. This, in turn, enables the virtual assistants to accomplish a broader range of tasks, ultimately enhancing their utility to end users. Additionally, end users themselves have the power to augment the skills of virtual assistants by creating logic flows using platforms like IFTTT [75], Workflow [165], and Shortcuts [147]. While these options undoubtedly expand the horizons of virtual assistants, they all share a common requirement: *manual* development effort.

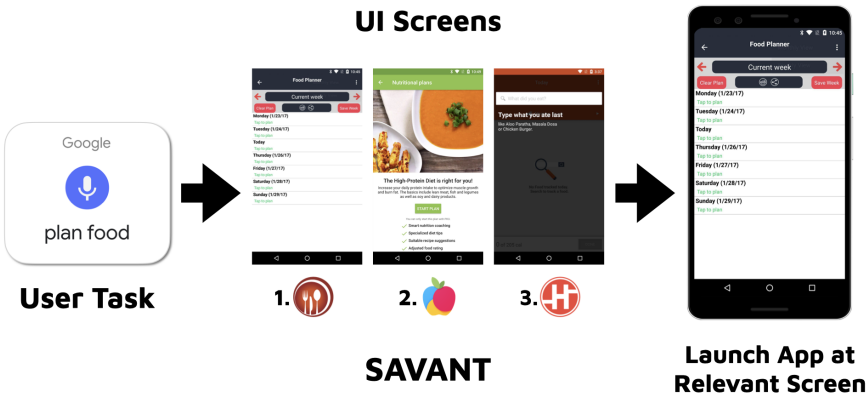

Figure 4.1: *SAVANT automatically generates* ***task shortcuts*** *for virtual assistants by mapping user tasks to relevant UI screens in apps.*

This chapter introduces SAVANT (**S**hortcuts in **A**pps for **V**irtual **A**ssistant **N**ew **T**asks). By leveraging interaction mining data, SAVANT automatically generates task shortcuts for virtual assistants by mapping user tasks to relevant UI screens in apps (Figure 4.1). SAVANT then uses these shortcuts to launch apps at the screens best suited to the task at hand. While users can directly engage with SAVANT using voice and text-based input, virtual assistants can also interface with this system, gaining the ability to autonomously explore and initiate

UI states within installed apps. This capability opens doors to perform tasks that virtual assistants might not currently handle, enhancing the overall user experience.

When presented with a natural language task (e.g., "send money to Joe"), SAVANT utilizes text and semantic cues within UIs to identify relevant screens. It employs intent modeling to map entities (e.g., "Joe") to UI inputs, typically involving an action (e.g., "send") and its object (e.g., "money"). UI elements like *icons* and *buttons* denote actions, while object inference occurs from nearby *text fields*. SAVANT combines UI data (icons, buttons, text) with app metadata from Google Play Store to efficiently match tasks to UI screens, generating *task shortcuts*. These shortcuts reference Android app UI screens (*activity* names) for launching. They also include a pre-recorded interaction trace, capturing the path from an app's home screen to the shortcut's UI screen. In cases where direct access is unfeasible, SAVANT navigates by replaying the interaction trace.

SAVANT, at its core, functions as an unsupervised search system, sifting through vast datasets of interaction traces to find the most fitting app screen for a given task. Beyond this fundamental capability, SAVANT extends its prowess to identify not just the apps relevant to user tasks, but also the specific screens within those apps that can serve as a launching point for task execution. In this way, it not only streamlines task automation but also enriches the user experience by bridging the gap between virtual assistants and the myriad functionalities offered by mobile applications. Importantly, SAVANT can also integrate with existing programming-by-demonstration (PBD) systems, providing them with valuable starting points for task demonstrations, fostering adaptability and generalization in automation.

In this chapter, we explore how interaction mining, facilitated by the ODIM framework, plays a pivotal role in enabling systems like SAVANT that benefit end-users. We examine the technical details of SAVANT, its approach to automating task shortcuts, and the results that validate its effectiveness.

4.1 BACKGROUND & MOTIVATION

Virtual assistants enhance everyday task automation through voice commands, with popular technologies like Google Assistant for Android and Siri for iOS now integrated into smartphones. These virtual assistants offer extensive opportunities for task automation by leveraging installed apps. However, their capabilities are limited by the functionalities exposed by app developers. If developers do not explicitly make specific tasks accessible for virtual assistants, these valuable features remain untapped.

Even when developers make such efforts, there are constraints in place on both Android

and iOS platforms. In Android, an *intent* serves as a description of an operation to be executed [44], and "built-in intents" encompass predefined operations that Google Assistant can perform. However, this system only supports a finite set of built-in intents, effectively limiting the scope of interactions with apps. On the iOS platform, while developers can encode custom intents to interact with Siri, end-users must still manually configure Siri to enable per-app automations.

To illustrate, consider the PayPal app, which permits users to send money to one another. Android developers for PayPal can use *App Actions* [41] to inform Google Assistant of their app's ability to handle the built-in intent for "create money transfer" [43]. To activate an *App Action*, developers must add a specific file (`actions.xml`) to their app, essentially declaring to Google Assistant that PayPal is equipped to manage a predefined intent. Although the list of built-in intents is expanding, Google Assistant remains unable to automatically use apps for entirely custom tasks.

The process on iOS for enabling Siri to send money via PayPal mirrors the Android counterpart. iOS developers must include an `Intent Definition File` within their app, detailing the tasks the app supports, known as iOS intents. Unlike Google Assistant, Siri is compatible with custom intents in addition to system intents, offering more flexibility. However, developers must create a mechanism to *donate* the shortcut to Siri every time a user conducts a money transfer through PayPal [49]. Only once the app has donated the shortcut does Siri begin to use it automatically. Alternatively, developers can expose app capabilities to Siri via *Suggested Shortcuts* [51], which still requires end-user configuration for task automation.

In contrast to these approaches that necessitate extra development effort, SAVANT capitalizes on the information inherent in app UIs to correlate screens with tasks, facilitating the automatic generation of shortcuts that launch task-relevant screens within apps. SAVANT is motivated by the recognition that task descriptions often exhibit semantic relationships with UI components within apps. Therefore, it harnesses data gathered through interaction mining to pinpoint UI screens that are semantically related to a given task description. Importantly, SAVANT eliminates the need for developers to manually modify an app's code to expose its capabilities externally. Instead, it autonomously discovers an app's capabilities based on recorded interaction traces. For instance, SAVANT can automatically discern the UI state in PayPal where a user initiates a money transfer and generate a shortcut for that task. Subsequently, it utilizes this shortcut to directly navigate the virtual assistant, or the end user, to the relevant UI screen.

4.1.1 Related Work

Virtual Assistants: With the growing prevalence of virtual assistants [119], there has been a notable trend in enhancing their functionality [55, 178] and enhancing the user experience [30, 140]. Virtual assistants have advanced in their capacity to furnish users with additional information regarding on-screen content [68, 127]. Furthermore, solutions such as *JustSpeak* have enabled virtual assistants to access labels for UI elements provided by developers [178]. Nevertheless, these existing systems primarily focus on offering information about screen content and enhancing the user interface. In contrast, none of these systems are explicitly designed to evaluate a screen in terms of its potential to assist users in completing specific tasks. SAVANT, on the other hand, harnesses the power of semantic annotation techniques to forge connections between UI screens, their individual elements, and the tasks they are apt for, effectively bridging the gap between virtual assistants and task-related functionality.

App and Feature Recommendation: Conventional systems focus on personalized app discovery based on usage patterns [170], context (e.g., location, time) [80], and privacy preferences [107, 179] but do not establish a direct linkage between user tasks and recommended apps. In contrast, SAVANT distinguishes itself by conducting a semantic analysis of UI elements on app screens to offer task-specific app recommendations. Traditional methods largely rely on predicting and recommending already installed apps for launch, leveraging spatio-temporal [8], sensory [145], and usage pattern [150] data, which enhances quick access to relevant apps. However, these approaches lack the capacity that SAVANT possesses to directly launch apps at screens tailored to specific tasks. Furthermore, while some systems have explored translating natural language task descriptions into application capabilities [1, 59] or providing conversation-relevant shortcuts for messaging apps [21], they are limited in their scope, as they primarily align tasks with specific app features, unlike SAVANT, which can operate with any app using pre-recorded interaction traces.

Mobile Deep Links: Deep linking conventionally refers to providing a uniform resource identifier (URI) that provides direct access to a specific page on a website, rather than the homepage. In the mobile application context, deep links extend this idea to direct users to particular UI screens within apps instead of web pages. The significance of mobile deep links has grown considerably, particularly in the context of evolving interaction methods like virtual assistants. While virtual assistants rely on third-party apps to fulfill various user queries, the responsibility still falls on app developers to manually expose and provide

deep links for their apps. Android, for instance, offers developers the ability to manually create deep links using intent filters, targeting navigation to specific content within apps [42]. To expand the range of actions available for use with virtual assistants, Google introduced *App Actions*; however, these actions necessitate manual developer support [41]. Various research initiatives have explored methods to assist developers in integrating deep links into their apps, including RESTful-style app models [111], developer libraries like *uLink* for user-defined deep links [7], and *Aladdin*, a static and dynamic program analysis approach for generating deep link APIs within apps [112]. However, these existing methods continue to place the onus on developers for connecting virtual assistants with app functionality. In contrast, SAVANT offers an automatic task shortcut solution that streamlines navigation to task-specific screens within an app, making it a zero-integration alternative to traditional deep links that often require significant developer effort and exhibit low adoption rates [73, 112].

Intent Modeling: Intent modeling is a critical aspect of task automation, involving the automatic parsing and mapping of expressions to specific desired tasks or outcomes, allowing for the extraction of relevant entities. Earlier task automation systems [98, 99] and conversational bots employ intent matching for interfacing with third-party applications [97]. Notably, SAVANT adopts *Dialogflow*, enabling it to train a robust agent [28] with only a few example phrases for each intent-entity pair. SAVANT also maintains flexibility through *Dialogflow*'s "automated expansion" feature for entities, which recognizes new values for entities based on pattern matching, enhancing its adaptability.

4.2 SYSTEM DESIGN

SAVANT maps user tasks to relevant app screens, and then launches those apps directly at those screens. SAVANT's screen search system leverages *interaction mining* data, which can be collected with systems like ODIM 3. This data comprise screen sequences — interaction traces — that capture each screen's visual (i.e., screenshots) and structural (i.e., view hierarchies) render-time properties. For each screen in the set of available interaction traces, SAVANT computes a representation, which is used to find screens that best match a given user task and ultimately generate a **task shortcut**. SAVANT leverages the shortcut to launch the most task-relevant app screen and populates the screen's UI elements with extracted entity values associated with the task intent (Figure 4.2).

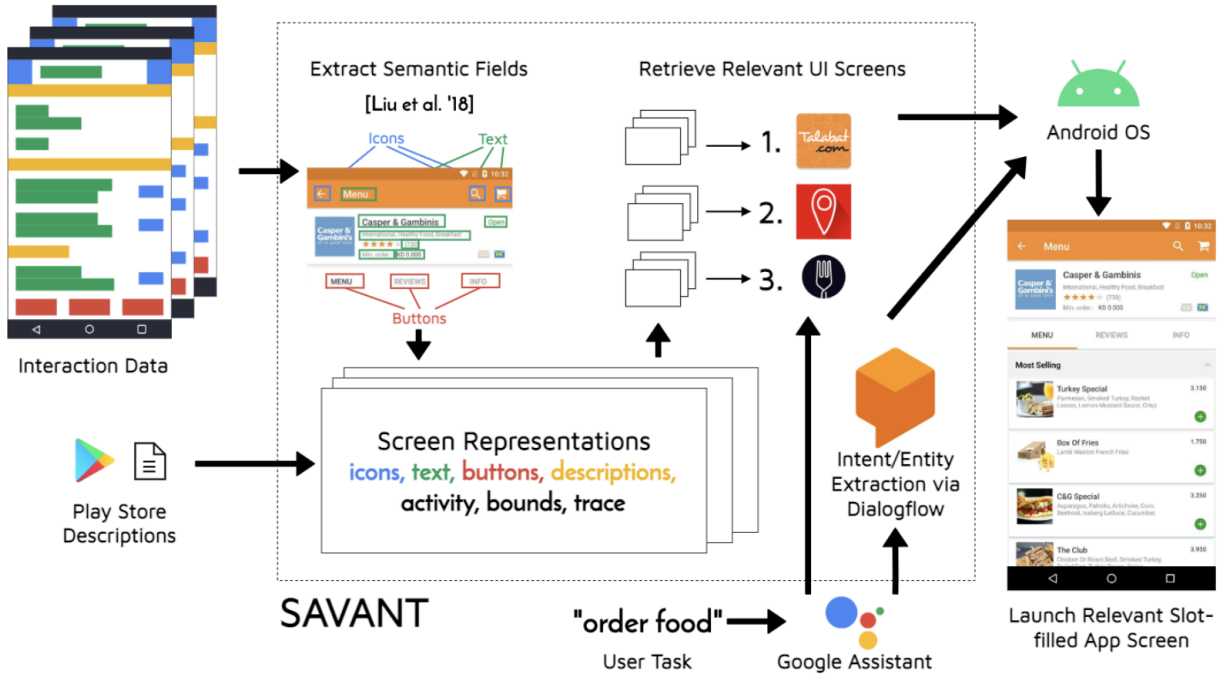

Figure 4.2: *SAVANT maps user tasks (e.g., "order food") to app screen representations generated through semantically annotated interaction data and app descriptions sourced from the Google Play Store. SAVANT then instructs the Android OS to open the most suitable app at the screen most relevant to the given task, and fills the screen's UI components with the relevant entity values linked to the task's intent.*

4.2.1 Generating a Task Shortcut

SAVANT uses semantic information contained within the UI components to compute screen representations, a kind of digital fingerprint for each screen. These screen representations encapsulate the semantic attributes of UI elements, such as icons, buttons, and text fields. Following Liu et al.'s approach, SAVANT augments these representations with semantic annotations, making it easier to match UI elements with user queries. [108].

For instance, *icons* and *buttons* are often indicative of the main action, or verb, in a user's query, while *text fields* are more likely to contain objects or nouns in the query. However, this approach isn't without its challenges, as sometimes text within these components can trigger false matches. For example, when a user queries "check email", many screens may contain the word "email" for various reasons. To refine these matches, SAVANT incorporates Google Play app descriptions, processed to eliminate common stop words.

SAVANT's screen representations include programmatic references that enable the direct launch of an app at a specific UI state. A view hierarchy, in addition to encoding the structural and rendering attributes of a UI, also contains the name of the screen's Android

activity, essentially "an app component that provides a screen with which users can interact in order to do something" [40]. In cases where a screen requires input parameters and cannot be launched directly, SAVANT retains a pointer to the relevant interaction trace to ensure navigation through replay.

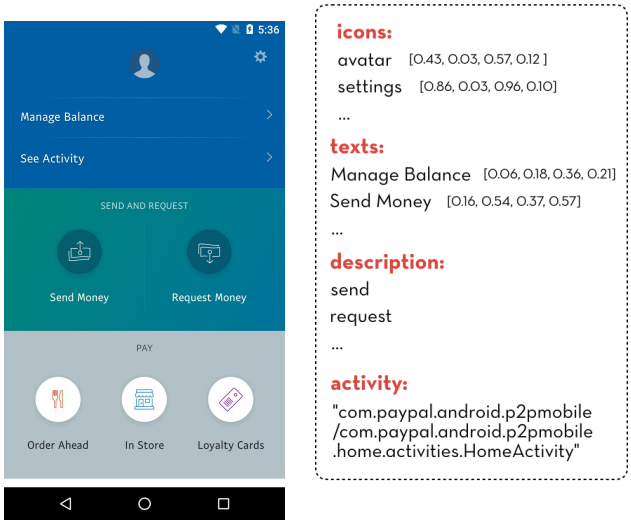

Figure 4.3: *Home screen for the PayPal app (a) and its computed screen representation (b), which encodes semantic information — `icons`, `texts`, and app `description` fields — and a programmatic reference for launching PayPal at this home screen — the `activity` name.*

To match user task queries with these screen representations, SAVANT employs an off-the-shelf cloud search service. The system pre-processes task queries by applying stemming and filtering out common, non-informative words. Additionally, SAVANT incorporates a set of synonyms for common actions in mobile apps, derived from UX concepts presented in Liu et al.'s work [108]. This approach ensures that task queries related to actions like "chat" also match screens containing phrases like "message" and "comment". The result is a collection of screen representations, each with a relevance score indicating its suitability for the query. These results are grouped by app, and an overall score is assigned to each app, emphasizing those with more matching screens. Finally, SAVANT presents the user with the top-rated screen representation from the highest-scoring app, referred to as a **task shortcut**.

```kotlin
val intent = Intent()
intent.component = componentName
intent.addFlags(FLAG_ACTIVITY_NEW_TASK)
context.startActivity(intent)
```

Figure 4.4: *Kotlin code required for creating an explicit intent to launch a new activity.* `componentName` *contains the target app and the activity.* `FLAG_ACTIVITY_NEW_TASK` *allows creating the activity as a new task on the same history stack.*

4.2.2 Using a Task Shortcut

A task shortcut contains the necessary information to launch an app launch an app at the precise screen required for a specific task. To showcase how these shortcuts can be used, we integrated SAVANT with a *system-level* Android app. Android apps often impose restrictions on launching their *activities* externally from other apps. However, system-level apps can request a unique privilege – the `START_ANY_ACTIVITY` permission. This enables them to override these restrictions and initiate *activities* within other apps.

Android apps primarily employ *intents*, essentially messaging objects [44], to request actions from components in other apps. There are two types of intents: *explicit* and *implicit*. *Explicit intent* specify the exact component within an app that will handle the action and are typically used to launch *activities* within the same app. In contrast, *implicit intents* declare a general action but lack specific information about the component. They delegate the action to another app chosen by the user. While *implicit intents* are apt for launching other apps to perform tasks, they often necessitate user intervention to choose the preferred app.

As a result, SAVANT uses an *explicit intent* to start the *activity* specified in the `activity` field of the task shortcut. This information includes the app (before the "`/`") and the specific *activity* within the app (after the "`/`") (Figure 4.3). Combined, these elements constitute a *component name*. SAVANT leverages this name to craft an *explicit intent*, dispatching it to the Android system (Figure 4.4).

However, some *activities* require specific user inputs for launching because they demand a particular internal state within the screen to be accessible. In theory, these inputs could be supplied externally via various `putExtra` methods of intents, enabling the inclusion of serializable data. Regrettably, task shortcuts do not currently support this approach, primarily due to the challenges of understanding the intricacies of each *activity*'s implementation. In such cases, SAVANT turns to the `trace` field within the task shortcut, enabling the replay of interactions leading up to the target component.

The interaction trace data is contains gestures, representing user actions performed on

the UI state, leading to transitions within the app. This data not only holds information about the matched UI state but also encompasses the starting state and the sequence of transitions that culminate in the desired UI state. By initiating these transitions and simulating gestures on the screen starting from the initial state, SAVANT navigates the user to the intended UI state. This is possible through another permission exclusive to system-level apps: `INJECT_EVENTS`, which allows simulating gestures on the screen, offering support for two primary types of gestures: touch and swipe. These gestures cover most of the transitions between UI states. However, for exceptional cases like pinch-and-zoom, additional gestures might be needed. SAVANT deploys these gestures as events on the screen, drawing from static parameters in the interaction trace. To ensure users are aware of the choices that lead to the task-relevant screen, SAVANT presents a step-by-step replay. Should the replay process encounter difficulties, or in cases of altered app interfaces, SAVANT offers a fallback solution: launching the app at its home screen. This ensures that the user can still access the app even when direct navigation to a specific UI state is challenging.

In addition to its primary goal of assisting virtual assistants with unfamiliar tasks, SAVANT's system-level app implementation supports voice- and text-based inputs directly from the user.

4.2.3 Slot-Filling and Intent Modeling

A typical user task, like "send money to Joe", consists of an intent ("send money") and associated entities (in this case, "Joe"). To further improve the utility of a task shortcut, apart from launching a relevant app screen, SAVANT aims to autofill UI elements with user inputs related to the intent's entities. This problem, known as the slot-filling problem, is addressed through the implementation of a *Dialogflow* agent [28]. The primary function of this agent is to match user expressions to intents and entities, which can then be linked to the corresponding UI elements on the app screen (Figure 4.5).

The *Dialogflow* agent employs a combination of *rule-based grammar matching* and *machine learning* techniques to identify the most likely intent for a given natural language phrase. An intent in *Dialogflow* is defined as an "end-user's intention for one conversation turn". In the context of SAVANT, a "conversation" involves a single user task phrase, followed by SAVANT launching a specific application at a relevant UI screen. For instance, the trained agent can identify a user utterance as the "book flight" intent when the user intends to book a flight. In this analogy, intents are akin to verbs in sentences, while entities are used to represent nouns and adjectives within natural language phrases. For example, based on the user's utterance, the agent may associate a "destination" entity if it contains information

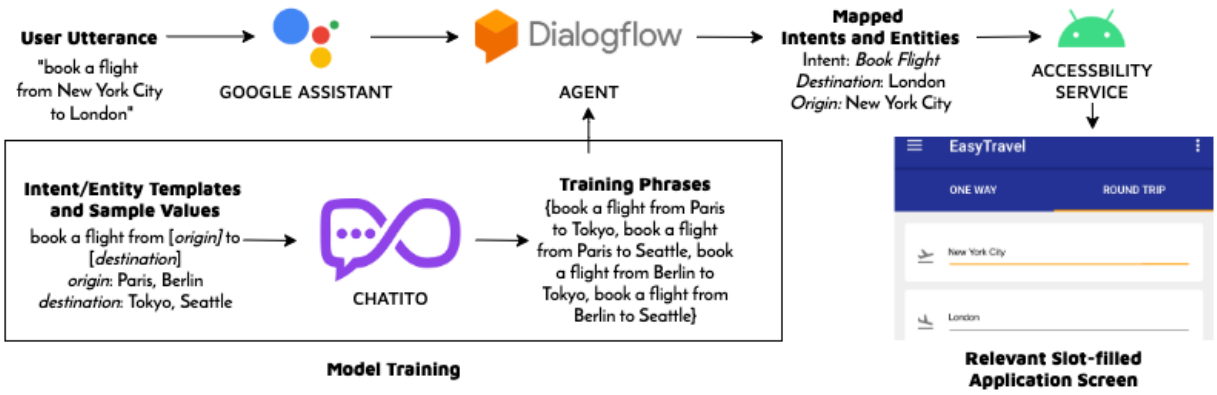

Figure 4.5: *SAVANT's Dialogflow agent matches user utterances to intents and entities; the extracted entities are used to autofill the values of any corresponding UI elements on task-relevant screens. The agent is trained on phrases generated by Chatito.*

about where the user wants to fly.

We predefined the initial intent and entity catalog of the agent, drawing from application concepts derived from prior research [108] and a formative study that identified common smartphone tasks (Section 4.3.1). This manual approach proves to be versatile and efficient since it establishes rules globally across all apps, rather than on a per-app basis. Automatically deriving intents and entities from UI elements found in user traces would be challenging due to the inherent lack of order among these elements. UI elements are more akin to a "bag of words" representation rather than a structured language utterance, making them less suited for training an agent. Extracting intents like *book flight* solely from analyzing the UI screen of an app like Expedia is a complex task.

We've developed a robust and adaptable set of intents and entities that serve as the foundation of SAVANT's *Dialogflow* agent. These sources collectively encapsulate common intents across various Android applications, including actions like *Add*, *Checkout*, *Forgot Password*, and *Terms of Service*. Some of these concepts can be directly expressed as *Dialogflow* intents, while others, like *Terms of Service*, cannot. Intents that fall within this subset of concepts also have corresponding entities. For instance, the *Add* intent may be associated with an entity for adding a *song* to a *playlist*.

Dialogflow agents are trained using example phrases and values for each intent and entity, respectively. For instance, a *greeting* intent might include a training phrase like "Hi, how are you", and a *restaurant* entity could have a sample value like "Olive Garden". To streamline the generation of *Dialogflow* training phrases, we employ *Chatito*, a widely used Domain Specific Language (DSL) for dataset generation [134]. Instead of manually crafting training phrases, we provide *Chatito* with template intent phrases like "book a flight from [*origin*]

to [*destination*]" along with a list of values for the entities (such as *origin* and *destination*) present in the template. *Chatito* then generates various combinations of the phrase with different entity values, significantly reducing manual effort. For a DSL that encompasses 25 intent templates and 20 entities, *Chatito* produces approximately 100 training phrases for each intent. We subsequently upload these generated phrases to *Dialogflow* and manually annotate the intent and entities within the phrases using *Dialogflow*'s web interface to train the agent.

Once SAVANT identifies the most relevant screen for a user task, it forwards the task phrase to the trained *Dialogflow* agent, which returns the most probable intent and associated entities extracted from the user's input. These entities are then mapped to UI elements on the application screen, and the values of corresponding components are set to the respective entity values. Since Android OS restrictions prevent regular applications from accessing elements in third-party apps, SAVANT employs accessibility services to access and modify elements on screens.

We employed Python scripts for the initial processing of interaction mining data and the calculation of semantic annotations. To carry out the search and ranking of screen representations, we used AWS CloudSearch. A Node.js API is responsible for tasks like stemming task descriptions, querying data within AWS CloudSearch, generating task shortcuts, and consolidating them within applications. The system-level Android application, along with the accessibility service used for slot-filling, was developed using Kotlin.

4.3 EVALUATION

We conducted a comprehensive user study to assess the effectiveness of SAVANT's semantic search in identifying task-relevant UI screens. While there have been numerous studies examining smartphone usage at the *app level* [10, 17, 27, 48, 90, 93, 169, 177], the exploration of smartphone usage at the *task level* is limited and lacks publicly available resources. To address this gap, we first conducted a formative study aimed at identifying common user tasks performed on smartphones. Subsequently, a second study was carried out to evaluate SAVANT's performance, focusing on the task set identified during the formative study.

4.3.1 Identifying Common User Tasks

The existing studies on common tasks performed on smartphones, such as the motivating study for *SUGILITE* [98], while insightful, did not offer publicly accessible results, providing only eight user tasks. Therefore, we conducted a new formative study involving 24 partic-

ipants. Participants were recruited through a university web programming course mailing list and were rewarded with a $10 Amazon gift card for their participation. Each session lasted approximately 15 minutes, during which participants were asked to list five tasks they typically perform on a smartphone. The tasks submitted by users predominantly featured higher-level intents (e.g., "message friends") rather than detailed, instance-level instructions (e.g., "tell my roommate I'm running late"). SAVANT's purpose is to act as an intermediary between virtual assistants and apps, primarily dealing with higher-level intents. Virtual assistants are already adept at extracting higher-level intents from detailed instructions, so SAVANT primarily focuses on higher-level intents.

The initial set of 120 tasks was grouped based on the Google Play Store categories they corresponded to, forming the initial task sets. Further examination of these initial groups led us to categorize highly specific tasks into their unique classes. For instance, "video chat" was separated from the initial "Communication" tasks category since the remaining tasks all pertained to text-based communication. Additionally, we merged certain thematically related task sets, such as "Tools" and "Weather". This process resulted in 20 distinct task sets, with the most frequently occurring task within each set serving as the representative task. In the event of ties, we selected the task phrase that shared the most common words with other tasks in the set (Table 4.1).

4.3.2 Measuring Task Relevance Precision

To evaluate the quality of the shortcuts recommended by SAVANT, we enlisted three participants, who were recruited from the same mailing list as the formative study (Section 4.3.1) and rewarded with a $10 Amazon gift card.

We initiated SAVANT with the *Rico* dataset [34, 108]. For each representative task identified in the formative study, we computed the top-three task shortcuts. We opted to display only the top three results, as some representative tasks had a maximum of three shortcuts, consistent with how virtual assistants operate when multiple apps can assist with a given task.

For each task, participants were presented with the top-three results generated by SAVANT and tasked with assessing the task relevance of each screen. Participants were only informed that a relevant task shortcut should guide them to a UI screen conducive to completing the specified task. This methodology was akin to previous research by Chen et al., where participants were asked to identify the 'top-3 relevant shortcuts for apps" based on the context of a text message [21].

Measuring *task success rate* was not deemed relevant for SAVANT, as it does not aim to

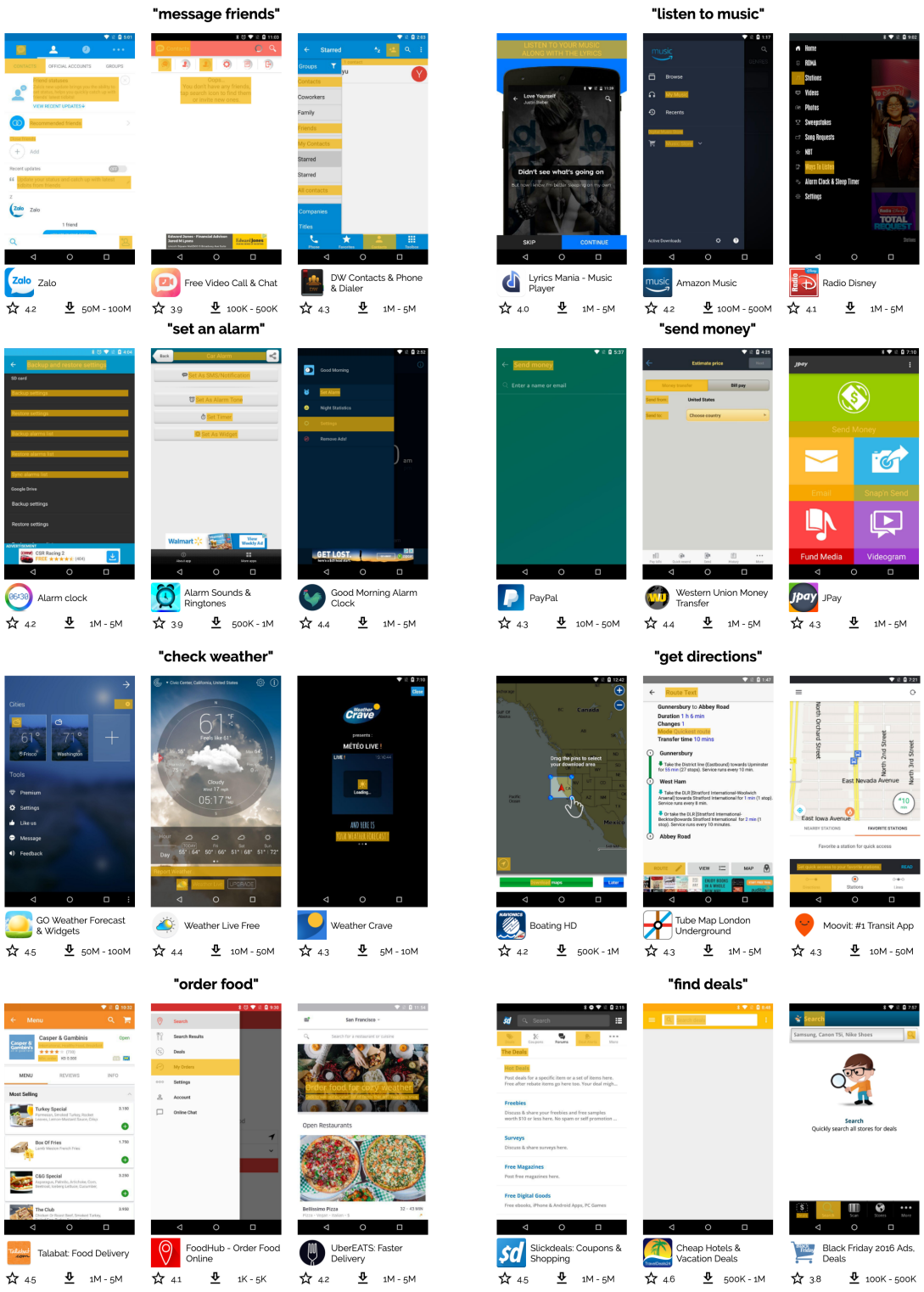

Figure 4.6: *SAVANT's screen matching results for representative tasks, in descending screen relevance order from left to right. Matching elements are highlighted for clarity.*

TASK SET	PRECISION (%)	TASK SET	PRECISION (%)
message friends (5), text friends (3), check email (2), communicate with people (2), chat with friends, group chat, message people, messaging, respond to emails, send messages, write email	55.6	**listen to music (6)**, play music (3), watch videos (2), browse forums, listen to podcast, recipe videos, watch movies	88.9
set an alarm (6), set a reminder (2), set alarm (2), set alarms, set reminder	55.6	**send money (3)**, check bank account (2), deposit a check, make a payment, make payments, mobile payments, paying people, track spending	100
check weather (3), check the weather (2), calculator, get sports updates, identify a song, search online, show tickets, take measurements, what time is it	77.8	check calendar (2), **view calendar (2)**, add tasks, calendar, calendar to do list, create events, to do list, view calendar events	66.7
get directions (2), find directions, navigate, navigating to a destination, navigation, transit time	77.8	**take notes (4)**, taking notes, write down notes	44.4
take photos (2), scan a document, take a picture, take pictures	100	**post a picture**, post pictures, post updates	11.1
track diet, **track workout**, view workout activity	66.7	**order food (2)**, buy groceries	100
read books (2), view document	33.3	**video chat (2)**	66.7
find deals (2)	100	**translate word**, translate words	88.9
book a cab, make reservation	77.8	**turn off lights**	33.3
practice flashcards	100	**play games**	66.7

Table 4.1: *20 unique task sets identified through the formative study along with SAVANT's precision at 3 values for each set. The representative tasks are highlighted. The numbers in the parenthesis indicate the number of occurrence in the original 120 tasks.*

fully automate tasks. Additionally, measuring *recall* was impractical, as it would necessitate exhaustive labeling of over 66k UI screens for each task, rendering it infeasible. Moreover, recall is not particularly advantageous for the general use case, where providing users with top-k task shortcuts suffices to facilitate task completion, aligning with SAVANT's primary objective.

We calculated the top-three precision for each of the 20 task sets. The average precision across all task sets was 70.1%. Impressively, SAVANT achieved 100% top-three precision for five tasks: "“send money”, “take photos”, “order food”, “find deals” and “practice flashcards”. In 13 out of the 20 task sets (65%), SAVANT provided at least two relevant task shortcuts, with an average top-three precision exceeding 66% (Figure 4.6).

However, it is essential to note that SAVANT encountered challenges in certain task sets.

For "take notes" (44.4%), SAVANT returned shortcuts from domain-specific apps like myPill® Birth Control Reminder and Blue Letter Bible, which offered note-taking as a side feature. Additionally, SAVANT suggested a shortcut from the FlightView Free Flight Tracker app since it interpreted the verb "take" as a synonym for "flight", "take off". The "post a picture" task yielded the poorest results (11.1%) because SAVANT identified only shortcuts related to posting on a forum or taking a picture. For "read books" (33.3%), two of the results were from unrelated apps like Hotels.com and SpiceJet, highlighting SAVANT's current inability to distinguish between verbs and nouns. The sole relevant shortcut for this task was from the Free Books app. Finally, for the "turn off lights" task (33.3%), SAVANT suggested shortcuts related to screen lighting configuration for the night, such as Night Light and Blue Light Filter apps, while participants expected shortcuts for managing smart home devices. The only relevant shortcut identified for this task was from the Color Lights Flashing app.

4.4 DISCUSSION

To ensure a robust real-world deployment of SAVANT, several key challenges must be addressed. While bootstrapping SAVANT with public interaction data repositories, such as *Rico*, served as a convenient strategy for proof of concept, real-world implementation necessitates a more sustainable approach. These challenges encompass sourcing app interaction traces, security and privacy concerns, and expansion to platforms beyond Android.

Sourcing Traces: While crowdworkers have been valuable for sourcing high-quality app traces, this approach is not scalable in the long term. Relying solely on crowdworkers is costly and limits trace collection to popular apps. To overcome this, interaction traces can be sourced in two additional ways. First, app developers could provide sample interaction traces along with app updates. These traces, especially in subsequent updates, need to cover new and updated UI screens. App developers could also source traces from crowdworkers or users to ensure they are up-to-date. App developers would be motivated to contribute traces to SAVANT, to enhance their app's visibility and utility through virtual assistants. Secondly, end users could opt to share their own app interaction traces. Leveraging user-contributed traces can result in more personalized task shortcuts and user experiences. For new apps, public interaction data could serve as a temporary solution until users generate their personalized traces. Both of these methods of providing interaction traces can be streamlined through ODIM.

Security and Privacy Concerns: Utilizing a system-level app like SAVANT for UI automation raises security and privacy concerns. System-level apps possess unique permissions, such as installing applications and simulating screen interactions, which are not available to regular apps. SAVANT uses these permissions exclusively for generating task shortcuts based on interaction data, making the likelihood of security issues minimal. SAVANT's design inherently requires malicious UI screens and interactions to be included within the interaction trace for a security problem to manifest. This is further mitigated by Google Play Store's own measures to detect and prevent the installation of apps containing malicious content [45].

Privacy concerns center around the exposure of personally identifiable information (PII). Interaction mining, by its nature, captures sensitive data. One proposed solution involves providing sock puppet accounts for trace contributors to protect their identity. For real user data, automated PII detection and removal mechanisms could be applied to ensure privacy compliance. Existing systems can identify conventional PII, like email addresses or phone numbers, while additional research is needed to develop user-centered privacy models addressing broader privacy concerns. Chapter 5 provides details on an initial exploration of such privacy concerns in the context of Android apps.

Expanding to Other Platforms: While the core approach of leveraging previously recorded app interaction traces for virtual assistant enhancement is platform-agnostic, SAVANT is currently tailored to Android apps and Google Assistant. Expanding to iOS or other platforms introduces additional challenges. iOS, despite offering some gesture automation capabilities, lacks a public interface for programmatically launching UI states, requiring knowledge of unexposed mechanisms within iOS. Nevertheless, this expansion remains a possibility. Moreover, SAVANT's applicability could extend beyond smartphones to encompass a variety of devices, including smart TVs, speakers, and appliances with non-touchscreen UIs, as long as mining and programmatically interacting with these interfaces is feasible, expanding its reach beyond smartphones and further augmenting its utility.

CHAPTER 5: EXPLORING PRIVACY CONCERNS IN MOBILE APPS

Interaction mining systems, like ODIM, collect data during app use, capturing screen contents and user actions. This data naturally contains personal information, introducing an obstacle that any given interaction mining system aiming to sustainably collect and disseminate interaction data must overcome: preserving user privacy. With the emergence of mobile applications, This ever-growing concern in the digital age has raised new questions about safeguarding users' personal information.

The traditional focus on Personally Identifiable Information (PII), covering data like names, email addresses, and phone numbers, has expanded with regulations like GDPR, now defining PII as "any information which is related to an identified or identifiable natural person" [139]. In the evolving realm of mobile applications, where people spend nearly four hours daily [128], a more nuanced approach to privacy is crucial. As apps integrate into various aspects of life, the definition of sensitive information transcends conventional PII safeguards. In apps like Venmo, users often choose to obscure payment memos from other users by intentionally writing vague statements or using emojis [16]. Similarly, fitness tracking app Strava offers "privacy zones" for users to protect sensitive locations such as homes and offices, but even these safeguards can be breached [66]. These instances underscore the need to acknowledge and protect user-defined sensitive data beyond traditional PII boundaries.

This chapter presents an exploration of this emerging class of sensitive information in digital user interfaces, **Beyond PII**. It presents the findings of a user study that goes beyond the conventional scope of PII to identify types of content that users consider sensitive when interacting with mobile apps. The study employs an earlier implementation of the redaction interface within ODIM, which we refer to as the *privacy probe*. This probe allows users to record their interactions with mobile applications and selectively redact regions within their traces. By leveraging established interaction mining techniques, this research offers insights into the elements users find sensitive and extends our understanding of sensitive data beyond typical PII categories.

In the following sections, we detail the methodology of our study, provide an analysis of the results, and discuss the implications for future privacy models. This chapter contributes a comprehensive list of common UI elements that users consider sensitive, empowering a more effective app design and development with user privacy in mind.

5.1 BACKGROUND & MOTIVATION

Mobile applications have transformed the way we manage our tasks, offering us a wide array of capabilities and conveniences. However, to unlock the full potential of these apps, users are often required to share their personal information. One pattern is, through system permissions at runtime, to ask users to allow access to private information that already exists on their phones. More common way is for users to provide this information through creating accounts and organically using the app. As many apps today have social features like profiles and feeds to provide a digital presence for their users, this information is not always fully kept private.

Consider the example of Venmo, a popular app used for monetary transactions among its users. Venmo allows users to send and receive money, request payments, and even offers a social component where transaction descriptions are publicly visible. While the app does provide the option to keep transactions private, the default setting is to share them publicly, a design choice that may not always align with users' privacy preferences. Novice users, in particular, may unknowingly share their transaction history with a broader audience, potentially including individuals they do not intend to share this information with.

To illustrate, let's follow the scenario of two coworkers, Matt and Rachel, who've just had lunch together. Rachel, having covered the bill, asks Matt to reimburse her through Venmo. Matt opens the app and navigates to Rachel's profile to make the payment (Figure 5.1). As he scrolls through Rachel's transaction history, he encounters entries related to her interactions with a person named Lee. These transactions reveal details about the type of transaction, the individuals involved, the date, and even suggest shared living arrangements. Crucially, Matt is not friends with Lee on Venmo, yet he gains access to private information related to Lee.

In this context, Rachel may not even be aware that her transactions are shared publicly, and yet, this inadvertent sharing shouldn't extend to Lee's private information. While Rachel has control over her own data, she cannot govern the sharing of information that pertains to other individuals. Lee's privacy is compromised, as he lacks the means to influence the data-sharing dynamics within the app. This dilemma underscores the importance of apps being able to discern what users would consider private, extending beyond traditional definitions of privacy.

To address this issue, our research presents the outcomes of a user study involving 19 participants. We explore the nuances of privacy in digital user interfaces by investigating what types of information users regard as sensitive. Our findings serve as a valuable complement to existing systems and pave the way for the development of future user-centric approaches

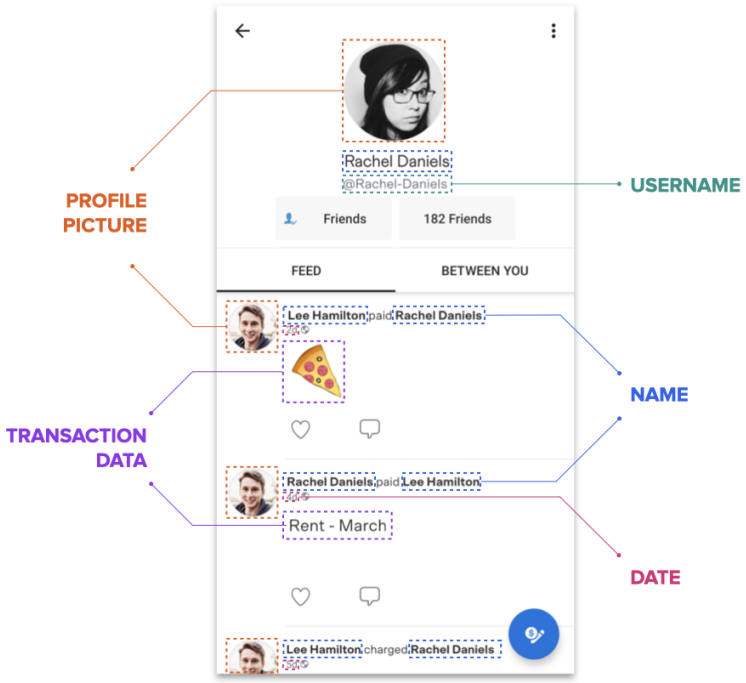

Figure 5.1: *Profile page of a Venmo user. Users can see their friend's feed showing all the public transactions they have made including ones between other users who are not their friends.*

to privacy in app design and functionality.

5.2 PRIVACY PROBE FOR COLLECTING PRIVACY-SENSITIVE UI REGIONS

In our pursuit of understanding users' privacy concerns within mobile applications, we harnessed the capabilities of the *ERICA* infrastructure [33]. To empower a comprehensive exploration of privacy considerations, we extended ERICA with a dedicated web interface that permits users to identify privacy-sensitive areas within the captured app screens. The outcome is a *privacy probe*, essentially a two-tiered system encompassing an infrastructure for recording traces and a web-based client for marking areas of privacy sensitivity (Figure 5.2).

The data captured by the interaction mining infrastructure is subsequently made available to the web client for the process of identifying and redacting privacy-sensitive areas. The web client, in turn, displays all the screens encompassed within a trace, allowing users to designate areas that they consider private by means of creating opaque rectangles over the pertinent regions. The redaction interface provides a comprehensive view of screens overlaid with user-defined privacy redactions. To redact a screen, users can select the screen they

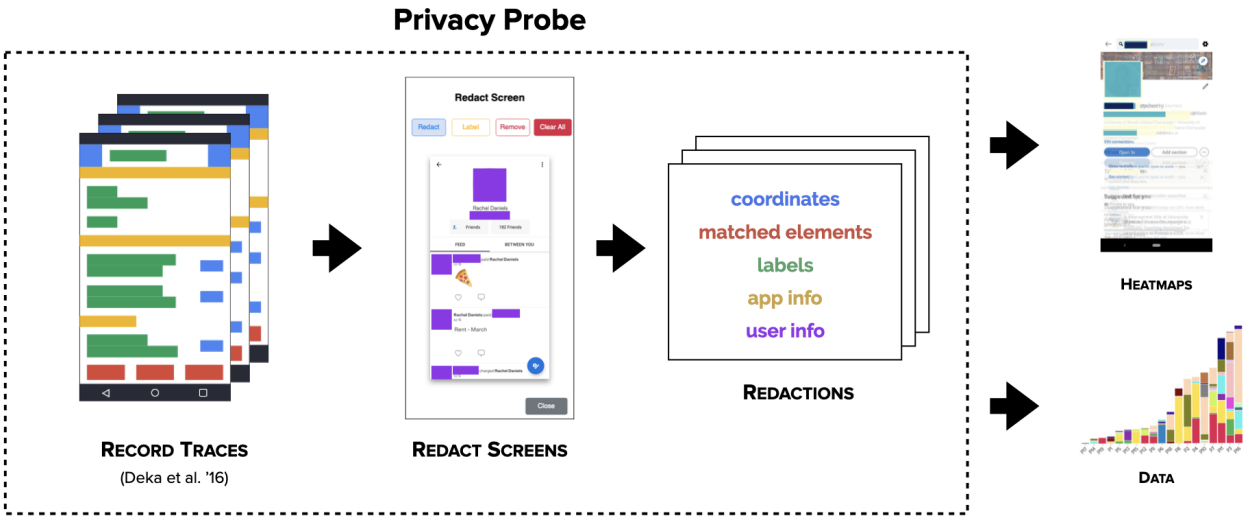

Figure 5.2: *The privacy probe leverages existing interaction mining work to enable recording and redacting traces to facilitate research on user privacy.*

wish to edit, which then opens the redaction editor. This editor presents a view of the app screen and offers four distinct modes: "Redact", "Label", "Delete", and "Clear All". In the default "Redact" mode, the system's interface is displayed as a canvas, enabling users to redact areas of concern by clicking and dragging the mouse to create an opaque rectangle, symbolizing the redacted segment.

After redacting an area, users have the option to label the redacted region to convey the type of information concealed. To achieve this, users can switch to the "Label" mode and click on the redacted region, triggering an input box for them to specify the label they wish to assign. If a user needs to remove a specific redaction from the editor, they can switch to "Delete" mode and simply click within the covered area to eliminate the redaction. For a more expedient option to clear all redactions from a screen, users can select the "Clear All" button within the editor.

Notably, when a redaction is created, the web client employs a positional comparison to align the redacted region with the corresponding element in the underlying view hierarchy. This facilitates the identification of the closest matching element within the structural composition of the screen, and this linkage becomes an integral part of the redaction data. As a result, a single user trace encompasses an extensive dataset, combining UI screenshots, programmatic representations (view hierarchies), user interactions (gestures), and a comprehensive set of privacy-sensitive regions, each endowed with their render-time properties, for every UI screen encountered within the trace.

5.3 EXPLORING PRIVACY CONCERNS

We conducted a user study to explore what users might find private in mobile app UIs. We first ran a preliminary study to determine a set of apps and then asked participants to record and redact interactions using the privacy probe. We grouped the resulting redactions into 22 total categories.

5.3.1 Methodology

We first conducted a preliminary survey that aimed to identify app categories containing apps that are more likely to be dealing with private or sensitive user data. This survey asked participants to select at least five categories from a pool of 35 available on the Google Play Store. Garnering a total of 85 responses through Amazon Mechanical Turk, each participant was compensated with $0.50 for their input.

Next, we selected the top five apps from the leading seven categories, as identified through the survey. These categories included *Finance*, *Dating*, *Medical*, *Business*, *Social*, *Communication*, and *Health and Fitness*. Ensuring that our privacy probe was compatible with these selections, we put each of the 35 chosen apps to the test. For any app found incompatible with our probe, the subsequent app in the same category, based on popularity, was chosen. Subsequently, we conducted an in-person user study with 19 participants, recruiting them through university mailing lists and networks. In appreciation of their involvement, each participant received a $20 Amazon gift card. Each study session spanned approximately 60 minutes and was composed of two phases: recording interaction traces and redacting privacy-sensitive areas within these traces.

Participants were then asked to choose two apps from the pool of 35 that they felt comfortable using their personal accounts with. If a participant found none of the provided apps suitable, we extended the flexibility of suggesting an alternative (Table 5.1). As a result, participants selected a total of 15 unique apps, with only three apps falling outside the originally defined categories. Upon selection, they were instructed to perform their typical daily tasks in these apps, replicating their everyday interactions using our privacy probe. Interaction trace recording commenced after participants logged into their accounts, ensuring the non-disclosure of login credentials. Our study's design aimed to minimize interference with the participants during the recording process, with guidance provided only upon their explicit request.

The redaction process invited a wide range of participant activity. While most participants chose to redact personal information present in screenshots, some opted for minimal

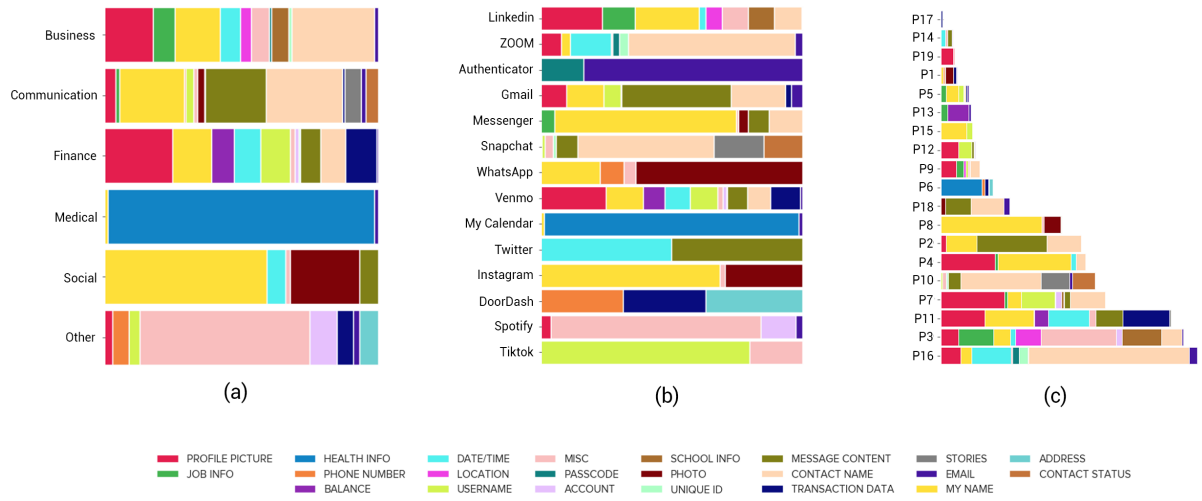

Figure 5.3: *Distribution of redaction categories by Google Play Store category (normalized) (a), app (normalized) (b), and by participant (c). Categories are sorted alphabetically, "Other" consists of the three apps outside of the predetermined categories. Apps are sorted by category. Participants are in ascending order with respect to number redactions.*

or no redaction, contending that their data was already publicly accessible (Figure 5.4). This variance in redaction activity highlights the diverse interpretations of what constitutes "private".

5.3.2 Results

With 19 participants, our study yielded 38 traces, 1416 screens, and a total of 3133 labeled privacy-sensitive regions across 15 unique apps. Employing open iterative coding, we established 22 primary categories for privacy-sensitive regions (Table 5.2). Alongside conventional PII, such as *names*, *email addresses*, and *physical addresses*, our study unveiled novel categories, including *contact names*, *message content*, and *date/time* (Figure 5.3). Notably, the *contact name* label category emerged as the most frequent, signifying users' tendency to consider their connections as personal information. This bias is due in part to the default "confidential" label, which some participants utilized when categorizing their redactions, thus contributing to the high frequency of this classification. We also observed a bimodal distribution concerning the number of redactions per participant, underlining the subjectivity surrounding the definition of private information in digital UIs (Figure 5.3c). Our findings are as follows:

Application	Category	# Traces	Participants
Venmo [157]	Finance	7	P2, P3, P6, P8, P12, P14, P16
LinkedIn [125]	Business	7	P4, P5, P6, P8, P10, P13, P15
Gmail [61]	Communication	6	P1, P2, P13, P17, P18, P19
Messenger [116]	Communication	4	P9, P14, P16, P19
Snapchat [76]	Communication	2	P11, P12
Twitter [154]	Social	2	P15, P18
Instagram [78]	Social	1	P9
WhatsApp [109]	Communication	1	P1
ZOOM [115]	Business	1	P17
Authenticator [6]	Business	1	P11
Flo [151]	Health & Fitness	1	P5
My Calendar [152]	Medical	1	P7
DoorDash [36]	Food & Drink	1	P7
Spotify [123]	Music & Audio	1	P4
TikTok [110]	Video Players	1	P10

Table 5.1: *15 unique apps selected by the 19 participants. Apps are first sorted by their Google Play Store categories, then by the number of recorded traces. Apps suggested by the participants, outside of the predetermined categories, are listed below the double line. Health & Fitness and Medical categories only has a single trace each while there is a healthy distribution between other categories.*

Mobile Apps Protect Conventional PII Adequately: We anticipated conventional PII labels to be dominant within our redaction labels. However, labels related to common PII, such as *email addresses*, *phone numbers*, and *physical addresses*, represented less than 2% of all redactions. This suggests that users do not often encounter typical PII during their everyday app usage, even though this data is retained by the apps. Furthermore, when such information is present in the app's UI, users tend not to regard it as private. Approximately 14% of redactions pertained to *profile pictures*, which included images of both users and their connections. Given the extensive study of visual privacy within the field, mobile app UIs can certainly benefit from the integration of existing privacy solutions in this domain.

Users Find Their Contacts Private: Users exhibit a greater inclination to keep their connection-related information private. A total of 21.6% of redactions pertained to *contact names*, with 8.3% linked to *message content*. This indicates that individuals tend to be more guarded about whom they share content with on these apps, in contrast to the content

CATEGORY	LABELS	CATEGORY	LABELS
CONTACT NAME (676)	connection name, who i am snapchatting, connection, group name, recommender name, schoolmate, friend name, name (search), sender, receiver name, recipient name, added me, contact name, recents list, who i recently snapped, best friend list, other user's name, connection 2, group, confidential recipient, contact info, connection names	BALANCE (62)	crypto balance, account balance, venmo balance
MY NAME (589)	name, first name, full name, personal name, person name	PHOTO (52)	photo, friend photo, picture
PROFILE PICTURE (441)	connection pic, connection profile, contact photo, other user's profile pic, profile picture, my profile picture, avatar, profile pic, profile photo, user face, connection profile pic, user profile pic, connection photo	STORIES (52)	private stories, friends' stories, private story, my private stories, friend's private story, my private story
MESSAGE CONTENT (260)	direct message, message content, email contents, beginning lines of email, text, message content preview, message, text message, convo history, my message, file link, subject, email preview, email content	LOCATION (48)	location, connection location
DATE/TIME (167)	meeting start time, dm timestamp, timeframe, am/pm time, meeting end time, meeting time zone, time, year, meeting date, date, date + time	EMAIL (42)	user email address, email id, email, user email, contact email, confidential email, personal email, contact email/net id
USERNAME (110)	username, receiver username	CONTACT STATUS (40)	friend status, digital relationship to me, snap friend status
JOB INFO (109)	job preference, position, past employee, job location, job avatar, company name, company photo, company, past career, past company, connection career, connection role, connection experience	UNIQUE ID (21)	personal id, personal meeting id, some id, meeting id, unique qr code, snapcode
TRANSACTION DATA (99)	transfer amount, internal transfer, transaction amount, personal transfer, amount, transaction date, transaction id, payment method, confidential card, debit card info	ACCOUNT (20)	user account, account, account 1, account 2, account 3, account 4, account 5, other account 1, other account 2
SCHOOL INFO (75)	school, school name, school avatar, personal major, school + title, connection education	PASSCODE (13)	passcode, one time password code
HEALTH INFO (71)	personal health info, body weight	PHONE NUMBER (10)	phone number, phone no., mobile number
		ADDRESS (7)	home address
		MISC (169)	personal info, private analytics, user input, private playlist title, title, private playlist cover, confidential

Table 5.2: *22 privacy-sensitive region categories determined through open iterative coding on 3133 redactions. The colors match the category colors in Figure 5.3. Numbers in parentheses indicate the label frequency in study results.*

itself. Notably, the majority of connection-related redactions were identified in Business, Communication, and Finance apps (Figure 5.3a). About half of the participants placed redactions on *contact names*, irrespective of the app category they selected. Interestingly, Instagram, a prominent Social app, witnessed no redactions pertaining to *contact names*, while roughly half of Snapchat's redactions were in this category. This variance highlights that users exercise different privacy perspectives concerning communication on Communication apps, compared to Social apps. Users also redacted profile pictures of both themselves and their connections, thereby consolidating data from both sources within the *profile picture* category.

Health/Medical Data is Highly Subjective: While apps like Flo and My Calendar offer comparable functionality as period trackers, they are categorized differently on the Google Play Store. Remarkably, only two traces, one for each app, were observed in our study (P5 — Flo, P7 — My Calendar). Although both apps address health-related data, their users exhibited contrasting perspectives on privacy. P5 refrained from making any redactions for Flo, whereas P7 redacted a substantial amount of *health data* from their My Calendar trace,

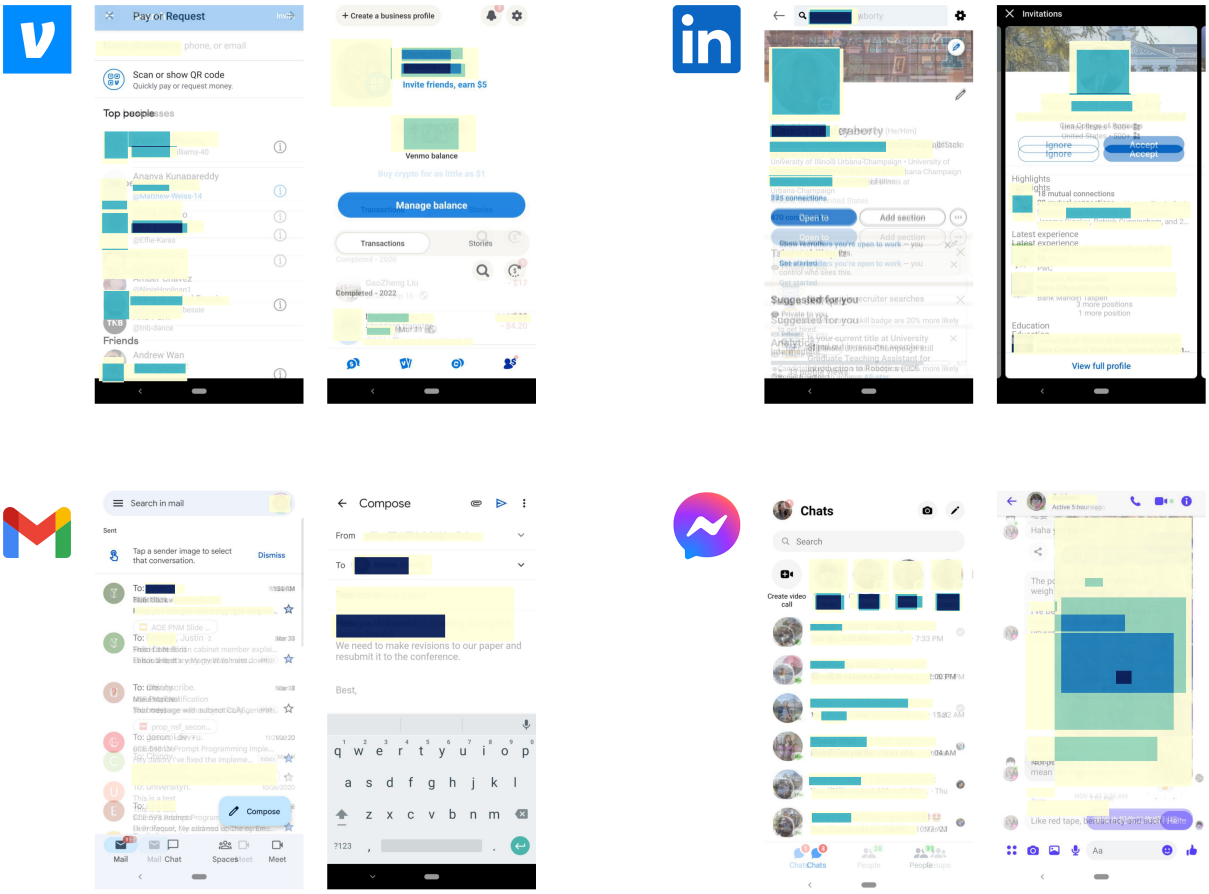

Figure 5.4: *Heatmaps generated for UI screens of the top-4 apps with respect to number of recorded traces. Similar screens are layered on top of each other. Darker regions indicate more redactions.*

including 71 of 73 redactions (Figure 5.3b). These divergent approaches underscore the diverse user conceptions of health and medical data privacy.

Social Features Drive Privacy Concerns: Apps with prominent social features elicited increased redaction activity. This trend was particularly prominent in Business apps, where we noticed a variety of redaction patterns. For instance, LinkedIn and ZOOM, both offering interaction capabilities, showed unique redaction trends. LinkedIn experienced a higher prevalence of redactions in categories *profile picture*, *job info*, and *my name*. Conversely, ZOOM featured redactions for *contact name* and *date/time*, reflecting the dominance of these categories. In contrast, Microsoft Authenticator, an app primarily focused on individual users, had redactions mainly associated with *email* and *passcodes* (Figure 5.3b). Intriguingly, Venmo, categorized as a Finance app, displayed the greatest diversity in redaction cat-

egories, including *transaction data*, *profile picture*, *my name*, *date/time*, and *contact names* (Figure 5.3a). This diversity may be attributed to apps like Venmo, which combine financial transactions with social and communication features.

Users Treat Their Own Data More Privately within Social Apps: Roughly half of the participants redacted their own names (Figure 5.3c). Specifically, within Social apps, the *my name* label accounted for 59.3% of all redactions, compared to 23.7% for Communication apps and 16.7% for Business apps. Instagram, a leading Social app, featured the majority of redactions labeled *my name* and *photo*, with no redactions for *usernames*, a category present in Finance and Communication apps (Figure 5.3b). This discrepancy may be attributed to the prevalence of social media usernames online, leading many participants to redact their own names and photos on Instagram, even though this information is publicly accessible. This suggests that individuals still value control over who can access their data, even when it is openly available.

5.4 DISCUSSION

In this chapter, we have explored the nuanced realm of privacy within digital user interfaces, particularly focusing on content that extends beyond the boundaries of PII. We have accomplished this through a comprehensive user study, facilitated by a privacy probe tailored for mobile apps. While this study has yielded valuable insights, there are inherent limitations that warrant acknowledgment, and avenues for future research that promise to further enrich our understanding of privacy considerations.

The participation pool in our study consisted of 19 individuals, a number that might appear modest at first glance. Nonetheless, this participant cohort has proven to be sufficient for the collection of substantial data, enabling in-depth analysis and the extraction of actionable insights pertaining to privacy in mobile apps. Despite this, one promising trajectory for future research is the expansion of participant numbers. A larger and more diverse group of participants would enhance the generalizability of findings and uncover deeper layers of insight into how users perceive privacy-sensitive content.

Although the study's focus was on Android, future work should endeavor to adapt similar systems for other platforms, notably iOS. The intricacies of the iOS platform, coupled with its unique mechanisms and APIs that are not publicly accessible, pose distinct challenges. Consequently, creating an iOS-specific tool would be a distinctive undertaking. This platform inclusivity is vital for gaining a holistic understanding of user privacy concerns across different digital environments.

Future research can pivot toward developing automated systems that detect and process interaction traces while concurrently safeguarding user privacy. These automated solutions would serve as invaluable additions to the ever-evolving landscape of digital privacy. Conventional PII can already be automatically detected; perhaps solutions based on Large-Language Models (LLMs) are the answer for automating the detection of Beyond PII highlighted in this chapter.

CHAPTER 6: BRIDGING ANALYTICS AND UX TEST DATA

In the realm of digital design and user experience optimization, two distinct but invaluable sources of data often converge to offer a comprehensive understanding of user interactions. These sources are *behavioral analytics*, which provide quantitative metrics, and *UX tests*, which gather qualitative feedback. While both capture behavioral data, they differ significantly in terms of techniques and tools, creating a gap that can be bridged for more profound insights. This chapter seeks to establish a meaningful connection between them through interaction mining.

Behavioral analytics tools, such as Google Analytics [62] and Adobe Experience Cloud [2], excel at aggregating usage data across sessions, providing quantitative insights and facilitating the computation of conversion metrics. On the other hand, UX tests, as facilitated by platforms like UserTesting [77], focus on gathering qualitative feedback through scripted scenarios, offering a more nuanced understanding of user interactions.

These data sources both originate from user interactions, encapsulating user journeys within digital products. Leveraging interaction mining principles, our goal is to unify these sources, recognizing their shared foundation. The convergence of behavioral analytics and UX tests enhances digital design workflows by providing a comprehensive understanding that blends quantitative and qualitative perspectives. By creating bidirectional mappings between these two data sources, we aim to explain behavioral patterns in analytics and validate insights from smaller-scale UX tests (Figure 6.1). To facilitate this confluence, we consider the application state information captured during UX test sessions.

We introduce a bidirectional mapping framework that defines four types of mappings to accommodate different use cases. Additionally, we present two design optimization workflows supported by interactive visualizations, namely *UX Maps* and *UX Path Flows*. UX Maps project UX test data onto the usage graph of an application, revealing opportunities for untested paths and highly-visited UI states. Meanwhile, UX Path Flows summarize user journeys through UX tests and align them with analytics funnels, offering a means to assess the importance of identified issues.

To validate the utility of these visualizations, we provide a case study involving a Software as a Service (SaaS) company's marketing website. Through this case study, we illustrate the practical application of our approach and its potential for enhancing UX analysis.

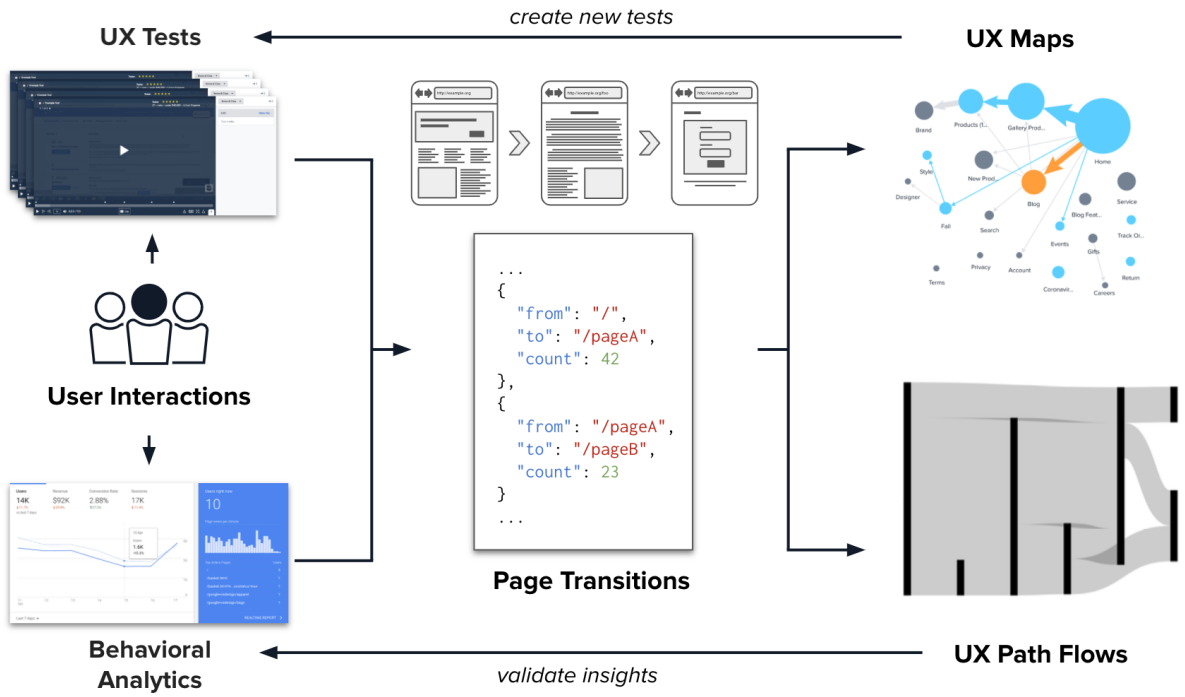

Figure 6.1: *Bridging behavioral analytics and UX test data through the interface of page transitions enables stakeholders to evaluate behavioral data by combining both qualitative and quantitative insights.*

6.1 BACKGROUND & MOTIVATION

To reify a UX analysis workflow that bridges behavioral analytics and UX testing, we present a concrete scenario. Tina works at a large e-commerce company as a UX researcher. The company has just released a new product line with a marketing campaign that includes external ads, promotional emails, changes to their homepage, and blog posts. Tina's UX team has been tasked with identifying UX improvements to the website that could increase product sales.

Tina starts by taking a look at the UX Map for her company's website (Figure 6.2). The UX Map visualizes analytics data as a network diagram, where the nodes represent clusters of similar pages (e.g., product pages, blog pages), and the edges capture the traffic flow between the clusters. Each node's size is scaled based on the sum of all incoming traffic to the pages in its cluster; Tina observes that the biggest nodes are the homepage and the product page cluster.

Additionally, the UX Map projects data from prior UX tests onto the analytics network, and highlights clusters and traffic flows that have been previously tested in blue. Tina notices that her company has previously tested flows from the homepage to product gallery pages

Figure 6.2: *A UX Map visualizes analytics data as a network diagram, where the nodes represent clusters of similar pages, and the edges capture the traffic flow between the clusters. Tina notices that the flows from the homepage to product gallery pages to individual product pages have already been tested (blue). Tina clicks on the highlighted (orange) blog cluster signifying untested high traffic to examine the detail view.*

to individual product pages.

Tina also observes that the cluster containing blog pages is highlighted in orange, which signifies a highly trafficked cluster that has not previously been tested. Tina clicks on the blog cluster to examine the detail view, and sees that there is very little traffic flowing from the blog cluster to the product page cluster.

Based on this data, Tina decides to launch a UX test to better understand how users navigate from blog posts to their corresponding product pages. She creates a test plan that asks participants to read through a blog post about the new product line, and then asks them to navigate to a new product's page where it can be purchased.

She launches this test to 20 participants, and leverages a UX Path Flow to analyze the results (Figure 6.3). A UX Path Flow aggregates behavioral data from a UX test's multiple sessions into a Sankey diagram. The nodes of the Sankey diagram represent pages visited by the test participants, and the edges between the nodes delineate their transitions between pages. The size of a node is proportional to the number of participants that visit that page, and the thickness of an edge is proportional to the number of participants that navigate between the edge's starting and ending pages. In aggregate, a UX Path Flow reveals common and anomalous paths taken by participants while performing a task.

Based on the UX Path Flow, Tina determines that 18 out of the 20 participants were able to complete the task: the two unsuccessful participants navigated to product pages that

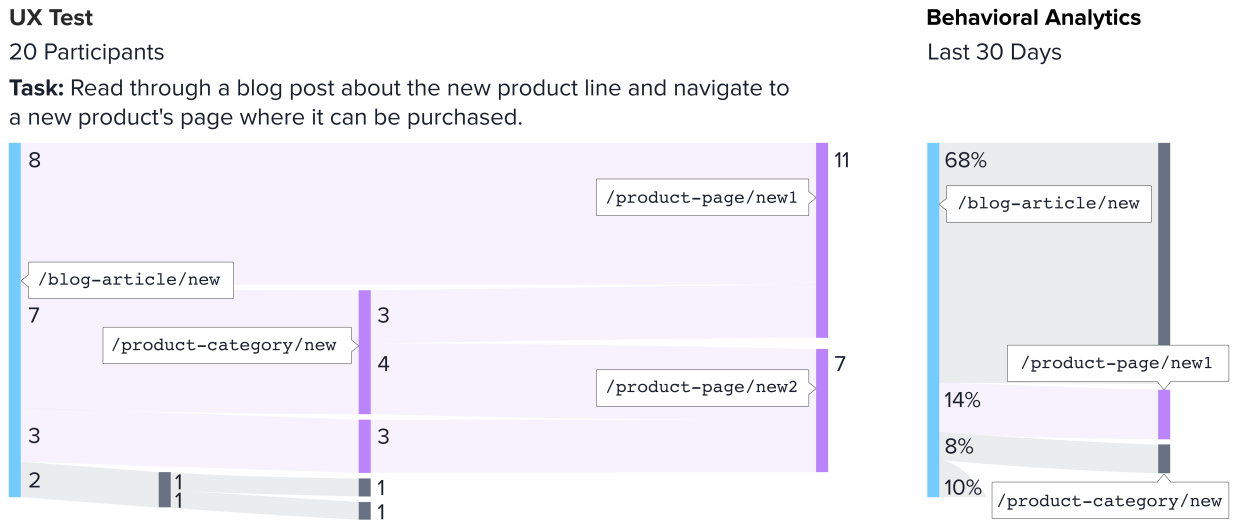

Figure 6.3: *A UX Path Flow aggregates behavioral data from a UX test's multiple sessions into a Sankey diagram. Tina sees that 18 out of the 20 participants were able to complete the task: the two unsuccessful participants navigated to product pages that were not part of the blog post. She selects these page nodes in the Sankey diagram to display their corresponding analytics data to confirm that the blog posts on her company's website should have more discoverable links.*

were not part of the blog post. To navigate successfully to the new product pages, eight of them clicked on the product images which were links to the product pages, seven selected the "New" category from the drop-down menu in the top navigation bar, and three searched for the new products by name.

Tina is surprised that 35% of the participants used the top navigation bar to find the product pages, and only 40% discovered that the product images were links to the product pages. She wants to see if this behavior is also borne out in the analytics data, so she selects the page node in the Sankey diagram corresponding to the blog post to see its corresponding analytics data.

The analytics data for the blog post shows that 523 people had visited the page in the last 30 days. The outgoing traffic visualization reveals that 68% of those page visitors go back to the homepage, 14% of the visitors visit one of the new product pages, and 8% of the visitors go to the product "New" category page. Although she cannot infer exact intent behind users navigating to the "New" category page from the blog post, she hypothesizes that most of them were looking for the new product pages. Therefore, the analytics data gives Tina more confidence in her analysis of the UX test: the blog posts on her company's website should have more discoverable links to the product pages, so that customers can more efficiently navigate to them.

Tina shares her findings and recommendations about the blog pages with the design team, so they can improve the user experience and increase the likelihood of sales conversions. The data from the UX test that Tina ran will be automatically reflected in the UX map, so that she and others in the organization will know that they tested this part of the site in the future.

6.2 THE BEHAVIORAL DATA BRIDGE

The two main methods for capturing how users interact with web applications are examining larger-scale **behavioral analytics data**, and evaluating smaller-scale **UX test data**. Analytics data provides aggregated quantitative metrics for in-the-wild usage data while UX test data reveals targeted, qualitative feedback on how users perform specific tasks. Although these two data sources have different scopes, they both capture behavioral data.

Many companies guide their business decisions based on behavioral analytics data as it contains at-scale information gathered through the organic use of their product. The raw analytics data often comes in the form of a list of page URLs detailing, for each page, how many times it was viewed (page views), which page the users came from (page paths), and what users did on the page (events). Analytics vendors offer customizable dashboards that aggregate this raw data into tables and diagrams. Using these dashboards, stakeholders can create funnels to filter out specific flows and gather at scale information on organic use.

As for UX tests, they play a crucial role in identifying UX issues and improving the user experience of a product. Users perform specific tasks during UX tests, and they interact with the application through its UI, generating interaction traces. These traces are often observed and recorded by researchers facilitating the test. There are also platforms such as UserTesting [77] which enable gathering behavioral and affective data from scripted tests. Although the outputs of UX tests might vary, generally they include a set of observations based on the specified task, a recording of the testing process, and user feedback directed towards any questions that might have been asked by the facilitator during the test.

Our aim is to enable *all* stakeholders to evaluate these two distinct data sources together by creating a bidirectional mapping between analytics and UX test data. Both methods are focused on obtaining insights through analyzing user interactions; but the tools and the workflows for evaluating the insights from these two behavioral data sources are very different. Using these together presents a myriad of possibilities for stakeholders. Since both are populated when users interact with an application, we can link together the qualitative outcomes from UX tests with the aggregated quantitative data from behavioral analytics by using these user interaction traces as an interface. We identify four types of mappings

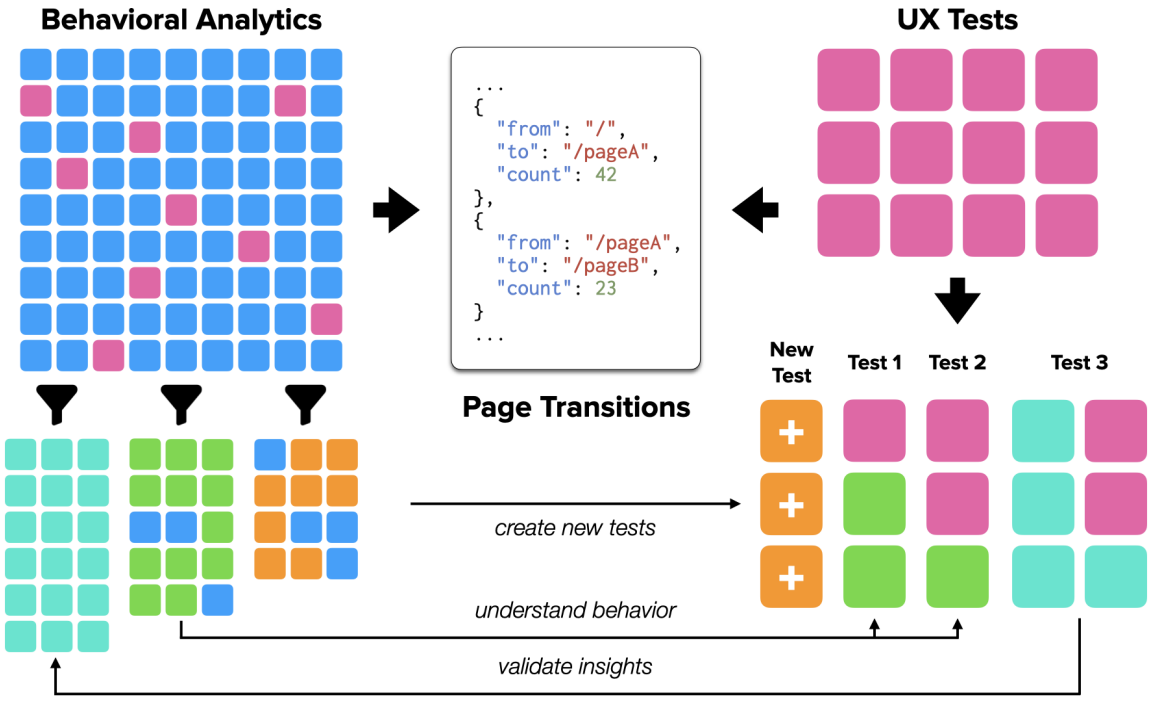

Figure 6.4: *The qualitative outcomes from UX tests can be mapped to aggregate quantitative data from behavioral analytics via page transition data. There are four types of mappings: (pink) a one-to-one mapping between scripted test sessions in a UX testing platform and those same sessions captured by an analytics platform; (teal) a mapping from an observed behavior in a UX test to similar behavior found in behavioral analytics; (green) a mapping from an observed behavior in an analytics funnel to UX test sessions comprising similar behavior; and finally, (orange) a mapping from an observed behavior in an analytics funnel to a new UX test that is launched to better understand that behavior.*

between behavioral analytics and UX test data (Figure 6.4). The first type is a direct, one-to-one mapping between scripted test sessions in a UX testing platform and those same sessions captured by an analytics platform. These mappings could allow a user in an analytics dashboard to identify sessions with additional metadata such as user think-alouds. The second type of mapping links an observed behavior in a UX test to a set of analytics sessions capturing similar behavior, validating if a pattern observed at a small-scale is borne out in the wild. The third type is a mapping from an observed behavior in an analytics funnel to UX test sessions exhibiting similar behavior. These relevant test sessions do not have to all belong to the same test. Similar to the first type of mapping, these relevant sessions allow users to drill into behaviors observed in the wild and better understand the human context for actions. Finally, the fourth type of mapping links an observed behavior in an analytics funnel to a new UX test that can be launched to better understand that specific action. Instead of mapping to pre-existing UX test results, it might be better to launch a

new test to understand if any recent changes to design have changed user perceptions and their modes for interacting with the digital asset.

The canonical representation of an interaction trace is a sequence of **page transitions** that occur during use. In the realm of web applications, a page transition occurs when users navigate from one page to another, which is indicated by a change in the URL. Clickstream analytics data already contains an aggregated count of how many times a page transition has occurred. However, not all UX testing procedures innately record these transitions. Augmenting existing UX testing procedures to systematically keep track of page transitions triggered by test participants will enable mapping them with behavioral analytics data.

To bridge these two data sources together, we first identify the active UI space of a given web application and then project both behavioral analytics and UX test data recorded for the application onto its UI space. We finally sketch two interactive visualizations — UX Maps and UX Path Flows — that demonstrate how this bidirectional mapping supports richer UX analysis workflows.

6.2.1 Crawling the UI Space

We define the UI space of an application as the collection of states that are reachable from a given entry point. For web applications, the entry point is usually the home page. As both data sources (behavioral analytics and UX tests) naturally contain transitions between the pages of web applications, they can be layered on top of their UI spaces.

In order to find the UI spaces of web applications, we built a web crawler that starts from an entry point (i.e. the home page), and discovers all available pages by recursively following the links available on every page. When the crawler visits a page, it records the raw HTML and the screenshot of that page. Using the raw HTML, we can identify the available links the page contains by looking at all the anchor (`<a>`) elements. The HTML anchor element creates a hyperlink to other web pages through its `href` attribute, which contains the URL that the hyperlink points to. This URL does not strictly have to be pointing to other pages; it can also point to files, email addresses, telephone numbers, etc. The crawler only keeps track of URLs that point to other pages within the same web application in order to populate the UI space. As a result, the crawler outputs a collection of available pages within the UI space of the web application, along with their screenshots and raw HTML. The crawler is implemented in Node.js, using the *Puppeteer* [46] and *cheerio* [20] libraries.

Automatic exploration of web applications has been a point of interest in the Data Mining and Extraction field. Crescenzi et al. introduced the concept of "crawling" web pages to extract data [31] and explored clustering them based on their structure [32]. Castillo

talked about policies that define the behavior of a web crawler, detailing selection, re-visit, politeness, and parallelization policies which became integral aspects of modern crawlers of today [18]. More recently, Kumar et al. introduced techniques for mining designs of web pages to automate curation of designs available in-the-wild [85].

We implemented our crawler guided by previous research. Given an entry point, our crawler finds reachable pages in a web application and allows exploration of UI spaces of web applications by recursively following links that appear on each page. We also record design data in the form of page screenshots and DOMs.

Researchers have also explored utilizing existing user interactions for automatic navigation through applications. Examples from GUI testing showcase promising results. Brooks and Memon suggested using interaction data from end-users to automatically explore application states for the purposes of GUI testing automation [12] while Chen et al. proposed augmenting crowd GUI testing methods with interactive event-flow graphs which track and aggregate every tester's interactions [24].

Although, leveraging UX test data to automatically navigate through some portion of the UI state is theoretically possible, we currently do not use any data from prior user interactions during our exploration. This data will only cover the fraction of the UI space that can already be explored with our current methods and leave out paths that are not yet covered by UX tests.

6.2.2 Translating Analytics Data

The page transition data contains the source and the target paths along with the number of users following that path. In the case of behavioral analytics data, these counts shows how many times a page transition has occurred during organic use and allows filtering out important quantitative metrics like conversion and drop-off rates. The page transitions from behavioral analytics will form the initial connections between the available states within the UI space. To demonstrate how the data from conventional clickstream analytics tools can be translated into a consumable format, we use Google Analytics as an example.

Google Analytics provides an API that can be queried in order to access raw usage data. Each query is required to have the **start-date**, **end-date**, and **metrics** fields. The first two fields indicate the time scope of the returned data while the metrics field denotes which of the many available quantitative metrics should be filtered for the response. We use the `ga:pageviews` metric to filter the user traffic information for each page. Google Analytics also provides additional ways to filter out metrics through the optional **dimensions** field. Dimensions allow users to categorize the returned *metrics* values with respect to a specified

property. As we want to understand how many times a page transition has occurred, we use the `ga:pagePath` and `ga:previousPagePath` dimensions. The result of this query is a list of page transitions along with a tally of how many times they have occurred.

6.2.3 Augmenting the UX Test Data

In order to augment the collected UX test data with page transitions, we use a proprietary web extension that records user actions and keeps track of the visited URLs as users interact with a web page. When a user event (mouse click, scroll, etc.) occurs, the extension takes a snapshot of the current page state. This snapshot includes the page URL, the visible Document Object Model (DOM), and the event type. By aggregating this event data, we derive all the page transitions that have occurred during UX assessment.

Having these page transitions from UX tests enables visualizing the current UX test coverage over the UI space of an application. In addition, when combined with behavioral analytics data, it is possible to identify if existing UX tests cover high-traffic pages and flows. All in all, this combination will inspire UX tests based on both behavioral analytics and UX assessment insights.

6.2.4 UX Maps: From Behavioral Analytics to UX Tests

Now that we have a list of available UI states within a web application and the page transitions that have occurred both during organic use and UX tests, we can combine these into a visualization that produces insights which can lead to new UX tests. Network diagrams are widely used to convey information about the transitions between a set of states. The nodes of a network diagram denote the available states while the edges show the flow between these states. A **UX Map** is a network diagram which has available UI states as nodes and page transitions from both behavioral analytics and UX tests as edges. In other words, UX Maps project transitions from both behavioral analytics and UX tests onto the UI space of a web application in the form of a network diagram.

Before we can use the list of UI states as nodes in UX Maps, we cluster similar pages into groups and use these groups as nodes. The motivation behind this decision is to decrease the visual load, since using each page as its own node would lead to a packed network graph. We first group all pages with respect to their URL paths using a prefix tree (Trie). We split each URL path using the slashes (`/`) as delimiters (i.e. the prefix) and form the page groups by "cutting" the prefix tree so that URLs that have the same start up to their second slash (`depth = 2`) would go into the same group (Figure 6.5a). Within each resulting group, we

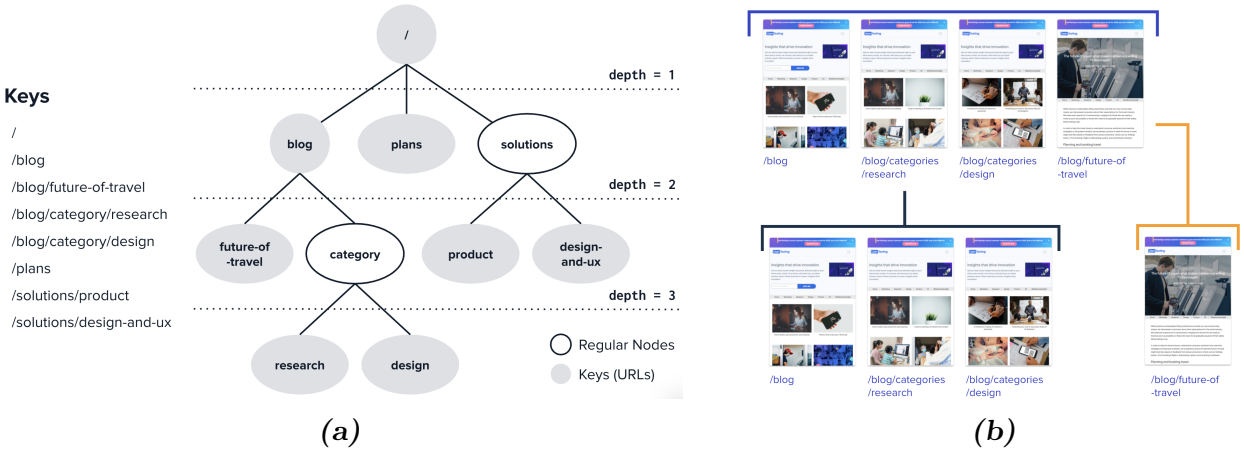

Figure 6.5: *We first cluster pages using a prefix tree by splitting their URL paths from "/" characters. We populate the prefix tree using the split URL paths and encode the path strings into the tree structure. The filled nodes on the prefix tree correspond to the listed keys (6.5a). We form the clusters by "cutting" the tree at* `depth = 2`*; this means pages having the same starting URL paths up to their second "/" will be put into the same cluster. We then calculate in-group UI similarity using the number of different kinds of DOM elements as a metric. We use this metric to further split these initial groups using agglomerative nesting (6.5b).*

calculate in-group UI similarity for each page using the counts of different kinds of DOM elements as a metric. We use the Agglomerative Nesting algorithm (AGNES) [81] to identify page groups that do not belong to their initial page path groups (Figure 6.5b). After splitting these page groups into their separate groups, we use these final list of page clusters as nodes in the UX Map. By default, the UX Map nodes are colored gray.

Translating the page transitions into cluster transitions is straightforward. As each page transition contains information about both the source and the target of the transition, we map them directly to transitions between page clusters. We match the source and the target URLs for each transition with the page cluster containing these URLs and form the edges.

We start by projecting the transition data from behavioral analytics onto the page cluster nodes. These transitions form the initial edges between the nodes and are colored with the same gray. The width of an edge indicates the amount of traffic flowing between the nodes it links together. We then compute the aggregated incoming view counts for each node and adjust their sizes accordingly; larger nodes have higher total incoming traffic. Additionally, we pick the top 10 nodes that have the highest incoming traffic and find the top 10 high flow edges connected to these nodes. While ranking the edge flows, we normalize the flow values using the total incoming traffic of their source node. The resulting nodes and edges are *pages and transitions of interest* and are highlighted with the color orange. We then

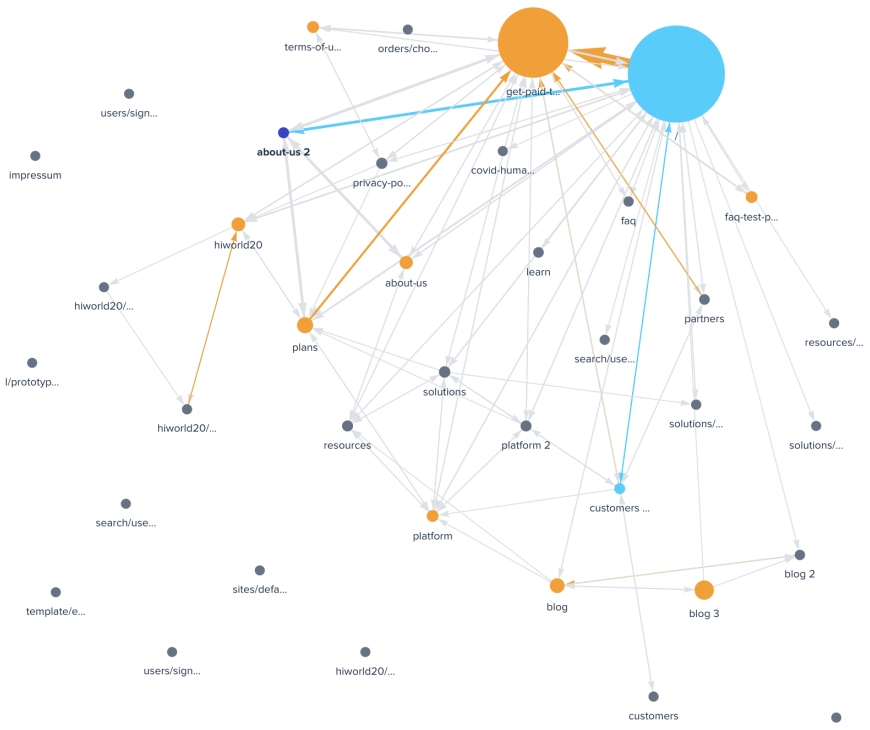

Figure 6.6: *UX Maps after projecting page transitions from behavioral analytics and UX test data. Node sizes and edge widths indicate user traffic. Pages and transitions of interest and are shown in orange. UX test coverage is shown in light blue. Selecting a node, shown in dark blue, reveals details about the page cluster it represents (Figure 6.7).*

layer the transition data from UX tests on top of everything. The edges that correspond to transitions occurring in UX tests are highlighted with blue. The result is a UX Map showing the UI space of the application along with behavioral analytics and UX testing data (Figure 6.6).

Clicking on a node reveals several details about the page cluster it represents (Figure 6.7). First, the analytics data listing the top incoming and outgoing traffic from and to other nodes is displayed. Next, the UX tests, if any, that cover the selected node are shown along with links that point to their associated resources. Lastly, the screenshot and the URL of each page in the represented cluster is presented.

By inspecting the UX Maps visualization, stakeholders can discover opportunities for new UX tests. The uncovered and highly trafficked UI states and paths indicated by the *pages and transitions of interest* are prime candidates for new UX tests. Prioritizing the identification and the resolution of UX issues that potentially exist within these flows would benefit the overall user experience of a web application more. The data from these newly run tests will then be automatically reflected on the UX Maps, increasing coverage and ensuring easier

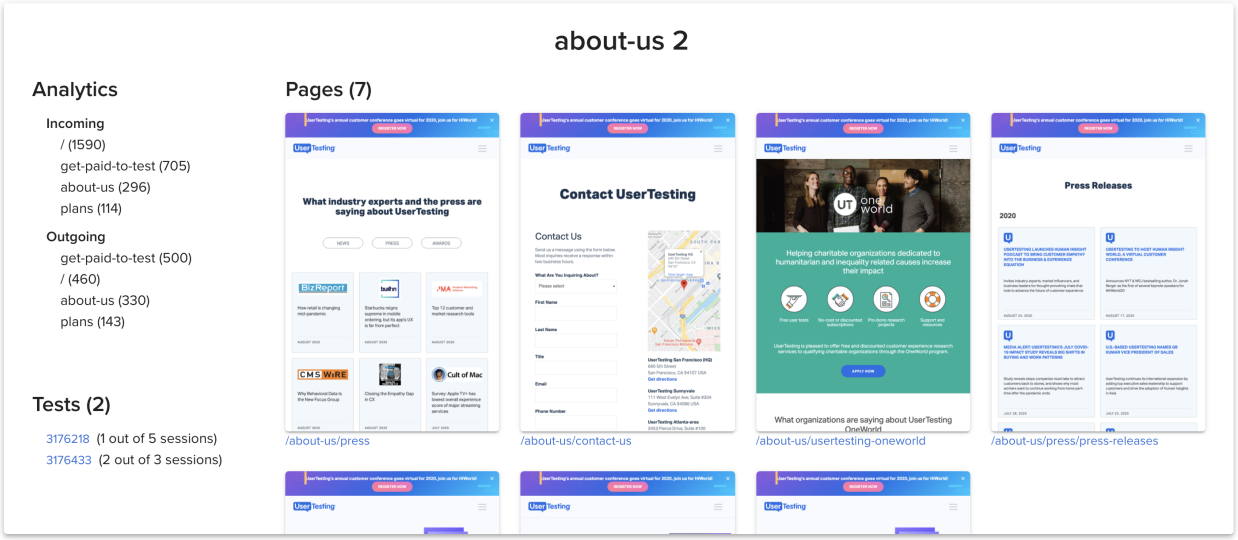

Figure 6.7: *Selecting a node reveals the top incoming and outgoing traffic from and to other nodes, the UX tests along with links pointing to relevant resources, and screenshots and URLs for each page in the cluster.*

collaboration within research teams. UX Maps enhances existing UX analysis workflows by grounding UX assessment decisions with proofs from the analytics world.

6.2.5 UX Path Flows: From UX Tests to Behavioral Analytics

We also want to evaluate and prioritize UX testing insights by utilizing behavioral analytics data. This is possible through the opposite direction of the mapping between the two data sources. UX tests are performed with multiple participants to identify different paths, interactions, and thought processes to achieve a specific task. Although UX tests are crucial in identifying the mental models of users, they cannot provide at-scale confirmation of these mental models. This leaves stakeholders with a set of UX findings that they potentially need to take action on. Combining these findings with behavioral analytics data will empower stakeholders to make recommendations with a firmer understanding of which issues impact most users.

We augment UX test outcomes with analytics data through UX Path Flows. A UX Path Flow displays aggregated behavioral data from the participants of a UX test in the form of a Sankey diagram. The nodes of this diagram represent the pages each participant has visited while the edges represent their transitions between these pages. Taller nodes and thicker edges indicate a larger number of participants visiting the page or performing the transition. UX Path Flows provide identification of common and anomalous paths. By projecting these

paths into the analytics space, it is possible to validate UX insights.

After running a UX test, UX Path Flows allow easy identification of success and failure paths for the specified task. Moreover, when a node is selected, UX Path Flows reveal the pertinent analytics data for the represented page; showing all transitions starting from the selected page to other pages in the application. We utilize page transitions to identify the source and the target pages that correspond to the transitions displayed in the diagram.

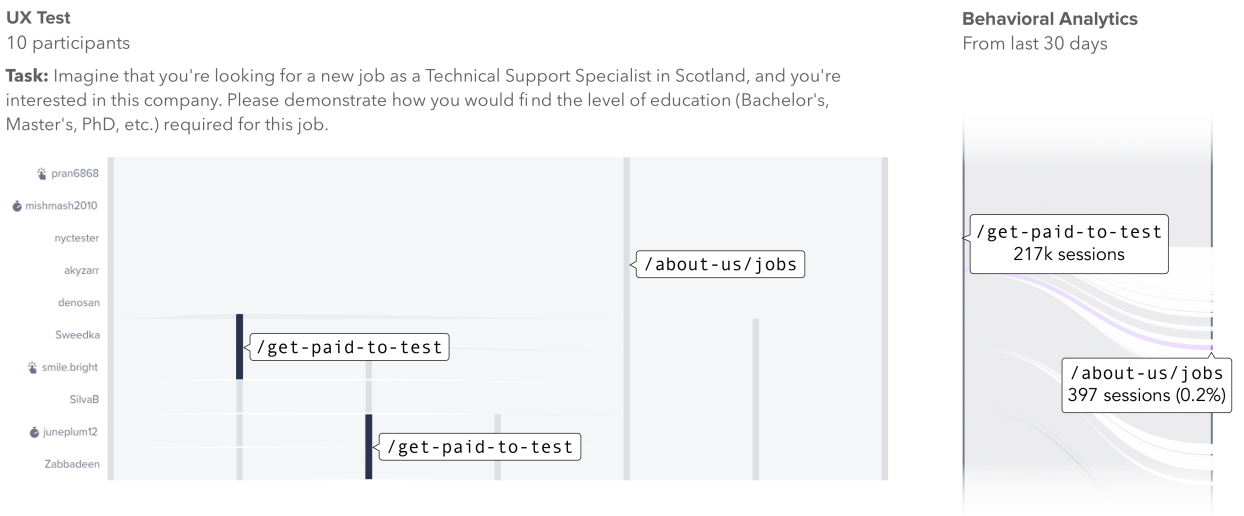

Figure 6.8: *UX Path Flow of a UX test where ten participants were asked navigate a website in order to find the educational requirements for a specific open position starting from the home page. Highlighted nodes indicate a UX issue where participants went to a different page then the intended. Highlighting these nodes reveals the relevant analytics data which conveys this transition rarely occurs in-the-wild (0.2%).*

For example, for a UX test where 10 participants were asked navigate a website in order to find the educational requirements for a specific open position starting from the home page, the UX Path Flow shows a summary of the participant paths through the website (Figure 6.8). The visualization displays similar interactions, alternative paths, and the unique end states. Inspecting the UX Path Flow representation reveals 4 out 10 participants taking an inefficient path (through `/get-paid-to-test` instead of directly `/about-us/jobs`). Highlighting these nodes reveals the relevant analytics data which tells a different story: this transition occurs very rarely in-the-wild.

Since UX Path Flows visualize all user journeys from a UX test, it is common to see multiple paths that indicate UX issues. Having access to analytics data makes it easier to gauge the severity of each identified issue. Identifying the impact of potential fixes to these issues helps define a prioritization which will empower stakeholders with actionable insights leading to a better allocation of efforts to improve user experience.

6.3 EVALUATION

We ran a qualitative, moderated study with five potential users of UX Maps in order to better understand and validate potential use cases. The study was focused on gathering initial impressions, allowing for organic exploration and feedback, and asking targeted questions around UX patterns and potential areas of improvement.

6.3.1 Methodology

Participants were recruited via direct contact in the authors professional network. All participants work within the customer success (three participants) or marketing (two participants) organizations at a technology company with over 500 employees. The company builds UX testing software, and Customer Success Managers (CSMs) are responsible for helping customers engage with/better adopt the product, often making recommendations for what their customers can or should be testing. The marketers are building, promoting, and maintaining the website of this company. All participants brought a breadth of experience with managing different types of customers (enterprise and mid-market) and funnels (current customer and new lead focused).

The 30-minute moderated sessions were conducted remotely via Zoom. While recording their screens and voices, a UX researcher and an engineer on the team gave participants an initial overview of UX Maps. Subsequently, we asked them a series of questions to gauge impressions, understand their needs and the potential value of the tool, and ask for feedback on specific elements of the experience. Sessions concluded with a Likert scale question to measure participants' level of agreement with the value proposition for UX Maps.

The interactive UX Maps visualization that participants were introduced to and asked to review was based on Google Analytics data from the past 30 days for the company where all participants are currently employed. This data was combined with UX assessment data collected for the company's marketing website, provided by the company.

6.3.2 Results

Overall, responses were largely positive, especially from the participants on the marketing team (P2 and P5) who had more context and familiarity with the information architecture and website elements included in the visualization.

> *This is awesome. I've been dying to see a better holistic view of our website, and where traffic is flowing. It's really hard to do that with Google Analytics because*

the most we get are tables and charts, and this visual representation provides a lot of information in one page. (P5)

When asked to provide their early impressions, a couple participants (P1 and P4) had some initial confusion about how the diagram was organized/prioritized, but were able to clarify and move on with ease.

Upon further organic exploration, all participants found value in the ability to see which areas of the site had been a part of qualitative UX testing. For participants who are Customer Success Managers (P1, P3, and P4), the combination of these two types of data would enable them to understand their customers' testing behaviors and identify gaps where they could recommend further research to increase engagement and adoption of UX testing tools. The two participants who are marketers felt that having access to this combined view would help mitigate the burden of having multiple data streams from different tools. It would allow them to better understand at a high level where they could dig deeper into their analytics data, make improvements to their website, monitor traffic, and conduct further UX testing.

They especially appreciated the contrast in how these areas were highlighted visually (with the color blue) in comparison to areas that were heavily trafficked (with the color orange), as it empowered them to ask why certain areas of high traffic hadn't been explored further from a research standpoint.

While probing for feedback around design decisions and system behavior, we explained to participants that the clusters were currently categorized based on URL paths as well as similarities in visual design elements on each page. We asked them whether they would prefer having clusters based solely on URL paths, or leave the categorization as-is. All participants indicated that they preferred the current set-up, because they found value in being able to see whether a particular visual design pattern was increasing or decreasing traffic to a page, even if the pages in question were in a relatively similar area of the site.

When asked where there was room for improvement, participants indicated that they would like to see more readily available context within the tool around how things were labeled and clustered.

I like the intuitiveness there [of the orange clusters], I like how you're showing me where I should test and dedicate my time to look into... My next question is, why is that recommended? Any type of analytics or like engagement metrics in conjunction with the recommendations from a visual perspective would be helpful context. (P1)

They also expressed a need for the ability to set up ideal flows or path funnels within the map, so that they could immediately hone in on areas that were most relevant to them, and

be able to specify parameters in addition to/other than the traffic metric. Finally, three participants (P2, P3, and P5) mentioned that they wanted a way to adjust the time limit on the tool. They explained that they would use this for multiple purposes, including looking back further than 30 days to see where previous testing had been done, and benchmarking traffic/metrics/testing for time periods where there was an event of significance (e.g. a marketing campaign to drive traffic, a conference that would require people to visit the website to purchase tickets, etc.).

When asked, all participants also expressed that being able to combine behavioral analytics data from UX Maps with UX testing path flow data in the form of Sankey diagrams (UX Path Flows) would be valuable.

> *For me to see gaps [in what my customers have tested], especially if there are gaps in what we've recommended, that would be really helpful. And [having access to analytics and Sankey data together] would be extremely helpful to our customers, especially since a lot of our customers have been talking about journey mapping lately, given how much the customer journey has changed lately with consumer behavior changing and everyone being home... It would help for me in my role [as a Customer Success Manager] if there's something that's happening organically that they haven't dove into, understanding why they haven't conducted this research and what the priorities are for them.* (P4)

To conclude the sessions, we asked participants to rate their level of disagreement (1) or agreement (5) with the following statement:

> *"UX Maps demonstrate how bridging behavioral analytics and UX testing can support richer UX analysis workflows. UX Maps project UX test data onto the usage graph of an application, revealing opportunities for new tests: highly trafficked UI states and paths that have not been tested before."*

All participants gave a rating of 4 or 5, and indicated that with some improvements as mentioned above, they would find UX Maps highly valuable in their workflows on a monthly if not weekly basis.

6.4 DISCUSSION

The results from the formative study suggest that people on both sides — those primarily using behavioral analysis platforms and those working with UX testing tools — find it useful

to map between organic traffic data and scripted tests. Future work should explore these bidirectional workflows more deeply.

UX Maps allow users to visualize historical UX test data in the context of usage traffic data, which can reveal opportunities for launching new UX tests. In the future, users could also define analytics-based triggers that could automatically generate and launch new UX tests. For example, a sudden increase in the drop-off rate along a conversion funnel could automatically trigger the launch of a UX test which could be used to diagnose the problem, especially after a design update. Users could define these triggers on either an analytics or UX testing platform, specifying them over conversion flows and other important funnels. If a user defined triggers on an analytics platform, it could leverage a test launching API defined by a UX testing tool to automatically generate tests when trigger conditions were met. To generate the tests, the test launching API could use existing test templates and autofill tokens based on trigger context provided by the analytics platform. Triggers could also be defined within a UX testing tool: it could monitor traffic of a digital asset through an analytic platform's API and similarly launch tests based on trigger context.

In the other direction, UX Path Flows map selected path transitions from UX test data to organic traffic data, allowing users to generate insights contextualized by analytics. In the future, testing platforms themselves could automatically generate these insights. For example, if a test creator determined a "success" state for a task, the system could generate insights that explain whether the distribution of paths for a test were similar or significantly different from those in the wild. This could help a test creator understand if the smaller test sample was representative of behaviors in the wild or if more participants needed to be added to the test.

We believe that these improvements will not only make UX workflows more powerful for user research practitioners, but also will give other stakeholders better access to user experience testing outcomes. The novel workflows we describe ground high-level overviews of qualitative data from UX tests with quantitative framing from analytics metrics, while also providing new tools to more efficiently engage with the research.

These new workflows have the potential to blur the lines of marketing, product, design, and UX research in industry. A marketer who traditionally looks at primarily analytics data might suggest concrete design changes if she is given more human context for why people are behaving the way that they do. Similarly, a designer equipped with analytics data might reprioritize their work to address UX problems that are more closely tied to business outcomes. They also create a way for team leaders to gain a high level understanding of team productivity and how user experience may have had an impact in leading to broader company-level outcomes by linking all research directly with quantitative outcomes.

CHAPTER 7: CONCLUSION

This dissertation introduced the ODIM framework, the first fully-functional interaction mining framework that enables capturing data from any Android app, without any specialized hardware or operating system modifications. ODIM democtratizes interaction mining for practitioners and end-users by enabling everyone to capture in-the-wild interactions. In addition, we showcased the potential of interaction mining applications made possible through easier access to interaction data. Similar to existing work, these applications play a complementary role for existing systems that depend on user interactions.

ODIM enables future datasets that can match the pace of app updates, making it possible to have healthier data for any system that leverages interaction data. Even though most of the machine learning models that have been trained with existing interaction mining datasets still perform well, the availability of up-to-date data through ODIM will potentially benefit both existing and future systems that leverage structural data on user interfaces. Such systems include end-user services like SAVANT (Chapter 4) and the privacy probe (Chapter 5), in which the correctness of programmatic UI representations are vital for identifying and labelling specific UI components.

The rising popularity of foundation models, specifically Large Language Models (LLMs) which leverage large amounts of textual data, also present exciting opportunities for ODIM. With its simplicity and scalability, ODIM is the perfect tool to provide these models with the at-scale interaction data they need. ODIM captures interaction traces that comprise sequences of UI screens with user gestures that result in transitions between these screens along with programmatic representations and screenshots for each UI screen. We also enable users to collect textual descriptions for interaction traces and the redactions performed on each UI screen. Combined, this data is highly suitable for use with LLMs and other foundation models with different modalities. Exciting potential applications include task automation systems powered by LLMs, automatic detection of privacy-sensitive UI regions, and the generation of UX insights through these models.

In the remaining sections, we navigate the intricacies of deploying ODIM in real-world scenarios, addressing critical factors that underpin its effectiveness. Our aim is to offer a comprehensive exploration of ODIM's application and its fundamental principles, offering insights on its potential to augment interaction data collection and integrate into the broader landscape of user-centric research and development. We then present a vision for a future interaction mining ecosystem enabled by ODIM, in which all stakeholders contribute to and benefit from public, and up-to-date interaction mining datasets.

7.1 CONSIDERATIONS FOR REAL-WORLD DEPLOYMENT

We outline five vital aspects to address for the practical application of ODIM: crowdsourcing traces, optimizing performance, dealing with accessibility ethics, preserving user privacy, and expanding to other platforms.

7.1.1 Crowdsourcing Traces

One pivotal facet of deploying ODIM revolves around crowdsourcing interaction traces. The ODIM framework integrates with crowdsourcing platforms, offering practitioners a convenient way to include new participants in data collection. Yet, the challenge of recruiting participants remains. Crowdsourcing platforms, while valuable, have inherent limitations that affect user recruitment. For a vibrant interaction mining ecosystem, it is crucial to ensure a continuous influx of traces. The question of how to incentivize individuals to willingly share their data remains a paramount concern. Perhaps, the answer lies in the hands of future interactive machine learning systems that can interface with ODIM, offering innovative solutions to this challenge.

7.1.2 Optimizing Performance

The performance of ODIM in real-world scenarios is another important consideration. ODIM employs an on-device approach to record user actions in the background through an accessibility service. A pertinent issue arises from the reported performance problems in Android apps that leverage accessibility features. The root cause of these problems is the extensive monitoring of various accessibility events, ranging from regular content change notifications to general announcements [136]. ODIM, however, only monitors a select subset of accessibility events that are directly relevant to capturing user actions, ensuring it does not introduce any accessibility-related performance lags. Nevertheless, the capture of UI screenshots during user touch events through a background thread did present challenges during development. It became evident that continuous screenshot capture and real-time matching with user events led to performance degradation. This issue was mitigated by refining the implementation to capture screens only when required, effectively eliminating prior performance overheads.

7.1.3 Accessibility Ethics

The ethical dimension of accessibility is a complex consideration for ODIM. ODIM leverages the capabilities of Android's accessibility service, a fact that raises questions regarding accessibility ethics. Android's official stance is clear: accessibility services should exclusively serve users with disabilities in utilizing Android devices and apps [39]. Despite this, ODIM utilizes Android functionality advised for enhancing app accessibility to collect interaction data. This data has already proven its worth in addressing and resolving accessibility issues, serving as a valuable resource to fuel future accessibility services and improvements [23, 132, 141, 142, 166, 175, 176]. This underscores the compatibility of interaction mining data with the mission of creating more accessible systems and training models to automate accessibility solutions.

7.1.4 Preserving User Privacy

The preservation of user privacy has been a central tenet throughout the development of ODIM. The framework incorporates two critical features to protect sensitive user data: the accessibility button and the redaction interface. The accessibility button empowers users by allowing them to enable and disable interactions capture at their discretion. The redaction interface provides a means for users to selectively remove UI elements containing privacy-sensitive information. However, both of these privacy safeguards largely depend on user diligence. It is conceivable that users might inadvertently miss some privacy-sensitive information when reviewing their recorded interaction traces. An ideal solution to this challenge would involve the automatic detection and removal of personally identifiable information (PII). Achieving this goal necessitates the development of user-centered privacy models, which, in turn, require the collection of at-scale interaction mining data. In this endeavor, ODIM stands as a valuable enabler.

7.1.5 Other Platforms

While ODIM's on-device interaction mining approach is inherently platform-agnostic, it currently has an exclusive implementation for the Android platform. Although iOS provides accessibility services for developers, the closed-source nature of iOS code raises uncertainties about the feasibility of recording user interactions. Despite this, ODIM's potential extends beyond smartphones. It holds the capacity to generalize to other devices, provided that they operate on Android or similar principles.

7.2 A VISION FOR A SUSTAINABLE INTERACTION MINING ECOSYSTEM

ODIM lays the foundation for a vision of the future: a **sustainable interaction mining ecosystem** where all participants play an active role in contributing to and benefiting from the wealth of interaction traces. In this ecosystem, UX researchers extract valuable UI insights through interaction mining, a critical element in need-finding for machine learning practitioners who seek to create user-centric solutions. The reciprocity of this interaction, where researchers contribute to the refinement of ML models, is amplified by end users who, in turn, improve these systems by providing their own interaction traces to a public interaction trace repository (Figure 7.1).

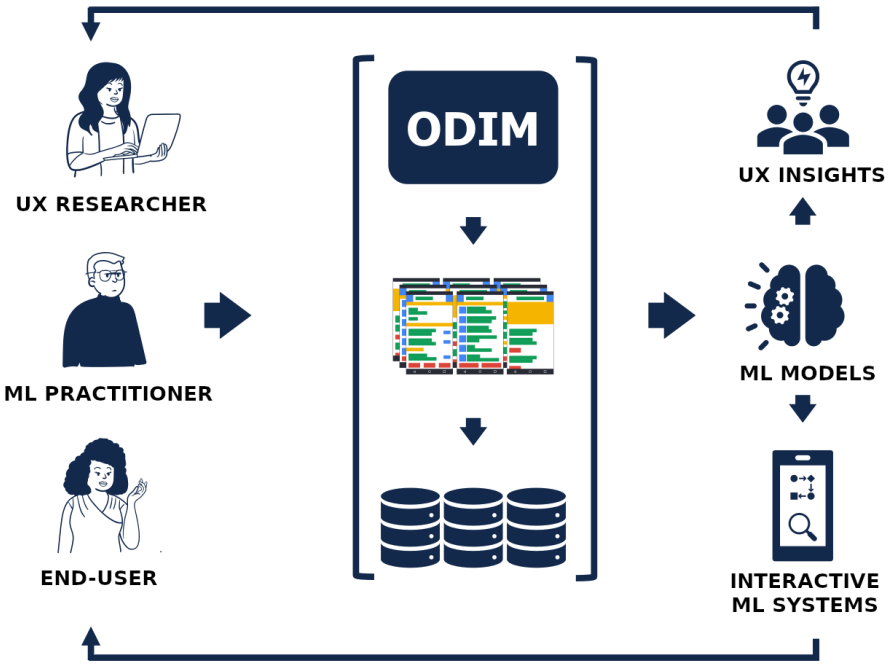

Figure 7.1: ODIM *is at the center of a future sustainable interaction mining ecosystem driven by collective contributions and mutual benefit.*

The outlook for this future is one of optimism, characterized by the following key outcomes:

Public Accessibility of Interactions: Interaction data is openly accessible, and it is a shared resource available to all stakeholders, fostering collaboration, research, and innovation.

Privacy without Compromise: Users can interact with digital interfaces without the constant worry of their privacy being compromised. The systems that collect interaction

data respect and safeguard user privacy at every stage.

Intuitive User Interfaces: User interfaces have evolved to a state where they consistently make sense, offering smooth and intuitive interactions that cater to individual needs and preferences.

Diverse Systems Enhancing Digital UX: A plethora of systems has emerged to enrich and simplify everyday life. These systems are founded on robust interaction mining, responding to the specific needs and expectations of users.

This vision is made tangible through the democratization of interaction mining, where ODIM stands as a catalyst. The framework propels the establishment of an open, sustainable interaction mining ecosystem driven by collective contributions and mutual benefit. ODIM empowers the UX community to exchange knowledge, unify around data-driven best practices, identify and rectify detrimental design patterns, and propel the development of the next generation of ML-backed user interfaces. In a world where digital user experiences are an integral part of daily life, every enhancement in accessibility, personalization, and efficiency carry the potential to generate substantial collective value.

REFERENCES

[1] E. Adar, M. Dontcheva, and G. Laput, "Commandspace: modeling the relationships between tasks, descriptions and features," in *Proc. UIST*. ACM, 2014, pp. 167–176.

[2] Adobe, "Adobe business: Products & services with adobe experience cloud," 2023. [Online]. Available: https://business.adobe.com/

[3] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *Ai Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[4] Amplitude, "Amplitude — the digital optimization system," 2023. [Online]. Available: https://amplitude.com/

[5] D. Arsan, A. Zaidi, A. Sagar, and R. Kumar, "App-based task shortcuts for virtual assistants," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021, pp. 1089–1099.

[6] M. Authenticator, 2022, https://play.google.com/store/apps/details?id=com.azure.authenticator.

[7] T. Azim, O. Riva, and S. Nath, "ulink: Enabling user-defined deep linking to app content," in *Proc. MobiSys*. ACM, 2016, pp. 305–318.

[8] R. Baeza-Yates, D. Jiang, F. Silvestri, and B. Harrison, "Predicting the next app that you are going to use," in *Proc. WSDM*. ACM, 2015, pp. 285–294.

[9] T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2018, pp. 1–6.

[10] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage," in *Proc. MobileHCI*. ACM, 2011, pp. 47–56.

[11] R. Bond, M. Mulvenna, J. Wallace, D. Finlay, K. Boyd, and M. Patterson, "Ux-handle: a cloud-based system to streamline usability testing analytics," in *Proc. HCI*. Association for Computing Machinery, 2017.

[12] P. A. Brooks and A. M. Memon, "Automated gui testing guided by usage profiles," in *Proc. ASE*, 2007, pp. 333–342.

[13] S. Bunian, K. Li, C. Jemmali, C. Harteveld, Y. Fu, and M. S. Seif El-Nasr, "Vins: Visual search for mobile user interface design," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New

York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi-org.proxy2.library.illinois.edu/10.1145/3411764.3445762

[14] P. Buono, D. Caivano, M. F. Costabile, G. Desolda, and R. Lanzilotti, "Towards the detection of ux smells: The support of visualizations," *IEEE Access*, vol. 8, pp. 6901–6914, 2019.

[15] A. Burns, D. Arsan, S. Agrawal, R. Kumar, K. Saenko, and B. A. Plummer, "A dataset for interactive vision-language navigation with unknown command feasibility," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer, 2022, pp. 312–328.

[16] M. Caraway, D. A. Epstein, and S. A. Munson, "Friends don't need receipts: The curious case of social awareness streams in the mobile payment app venmo," *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. CSCW, Dec. 2017. [Online]. Available: https://doi.org/10.1145/3134663

[17] J. P. Carrascal and K. Church, "An in-situ study of mobile app & mobile search interactions," in *Proc. CHI*. ACM, 2015, pp. 2739–2748.

[18] C. Castillo, "Effective web crawling," in *ACM SIGIR Forum*, vol. 39, no. 1. Acm New York, NY, USA, 2005, pp. 55–56.

[19] A. Cavoukian, "Privacy by design," 2009.

[20] cheeriojs, "cheerio — fast, flexible, and lean implementation of core jquery designed specifically for the server," 2021. [Online]. Available: https://cheerio.js.org/

[21] F. Chen, K. Xia, K. Dhabalia, and J. I. Hong, "Messageontap: A suggestive interface to facilitate messaging-related tasks," in *Proc. CHI*. ACM, 2019, p. 575.

[22] J. Chen, C. Chen, Z. Xing, X. Xia, L. Zhu, J. Grundy, and J. Wang, "Wireframe-based ui design search through image autoencoder," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 3, pp. 1–31, 2020.

[23] J. Chen, A. Swearngin, J. Wu, T. Barik, J. Nichols, and X. Zhang, "Towards complete icon labeling in mobile applications," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–14.

[24] Y. Chen, M. Pandey, J. Y. Song, W. S. Lasecki, and S. Oney, "Improving crowd-supported gui testing with structural guidance," in *Proc. CHI*, 2020, pp. 1–13.

[25] Y. Chen, M. Zha, N. Zhang, D. Xu, Q. Zhao, X. Feng, K. Yuan, F. Suya, Y. Tian, K. Chen *et al.*, "Demystifying hidden privacy settings in mobile apps," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 570–586.

[26] E. H.-h. Chi, "Improving web usability through visualization," *IEEE Internet Computing*, vol. 6, no. 2, pp. 64–71, 2002.

[27] K. Church, D. Ferreira, N. Banovic, and K. Lyons, "Understanding the challenges of mobile phone usage data," in *Proc. MobileHCI*. ACM, 2015, pp. 504–514.

[28] G. Cloud, "Dialogflow," 2021. [Online]. Available: https://cloud.google.com/dialogflow

[29] Contentsquare, "Contentsquare — digital experience analytics - dxp analytics," 2021. [Online]. Available: https://contentsquare.com/

[30] B. R. Cowan, N. Pantidi, D. Coyle, K. Morrissey, P. Clarke, S. Al-Shehri, D. Earley, and N. Bandeira, "What can i help you with?: infrequent users' experiences of intelligent personal assistants," in *Proc. MobileHCI*. ACM, 2017, p. 43.

[31] V. Crescenzi, G. Mecca, P. Merialdo *et al.*, "Roadrunner: Towards automatic data extraction from large web sites," in *Proc. VLDB*, vol. 1, 2001, pp. 109–118.

[32] V. Crescenzi, P. Merialdo, and P. Missier, "Clustering web pages based on their structure," *Data & Knowledge Engineering*, vol. 54, no. 3, pp. 279–299, 2005.

[33] B. Deka, Z. Huang, and R. Kumar, "Erica: Interaction mining mobile apps," in *Proc. UIST*. ACM, 2016, pp. 767–776.

[34] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar, "Rico: A mobile app dataset for building data-driven design applications," in *Proc. UIST*. ACM, 2017, pp. 845–854.

[35] B. Deka, Z. Huang, C. Franzen, J. Nichols, Y. Li, and R. Kumar, "Zipt: Zero-integration performance testing of mobile app designs," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, pp. 727–736.

[36] D. F. Delivery, 2022, https://play.google.com/store/apps/details?id=com.dd.doordash.

[37] A. Developers, "Accessibility event," 2023. [Online]. Available: https://developer.android.com/reference/android/view/accessibility/AccessibilityEvent

[38] ——, "Accessibility node info," 2023. [Online]. Available: https://developer.android.com/reference/android/view/accessibility/AccessibilityNodeInfo

[39] ——, "Accessibility service," 2023. [Online]. Available: https://developer.android.com/reference/android/accessibilityservice/AccessibilityService

[40] ——, "Introduction to activities," 2021. [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities

[41] ——, "App actions," 2021. [Online]. Available: https://developers.google.com/actions/appactions

[42] ——, "Handling android app links," 2021. [Online]. Available: https://developer.android.com/training/app-links

[43] ——, "Built-in intents," 2021. [Online]. Available: https://developers.google.com/actions/reference/built-in-intents/

[44] ——, "Intents and intent filters," 2021. [Online]. Available: https://developer.android.com/guide/components/intents-filters

[45] G. Developers, "Play protect," 2021. [Online]. Available: https://developers.google.com/android/play-protect

[46] ——, "Puppeteer — tools for web developers," 2021. [Online]. Available: https://developers.google.com/web/tools/puppeteer

[47] S. Dittrich, F. Hof, and A. Wiethoff, "Interacdiff: Visualizing and interacting with ux-data," in *Proc. MuC*, 2019, pp. 583–587.

[48] T. M. T. Do, J. Blom, and D. Gatica-Perez, "Smartphone usage in the wild: a large-scale analysis of applications and context," in *Proc. ICMI*. ACM, 2011, pp. 353–360.

[49] A. D. Documentation, "Donating shortcuts," 2021. [Online]. Available: https://developer.apple.com/documentation/sirikit/donating_shortcuts

[50] ——, "Siri shortcuts," 2021. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/sirikit/overview/siri-shortcuts/

[51] ——, "Suggesting shortcuts to users," 2021. [Online]. Available: https://developer.apple.com/documentation/sirikit/shortcut_management/suggesting_shortcuts_to_users

[52] J. Dong, Y. Zhao, and T. Peng, "A review of design pattern mining techniques," *International Journal of Software Engineering and Knowledge Engineering*, vol. 19, no. 06, pp. 823–855, 2009.

[53] B. Doosti, T. Dong, B. Deka, and J. Nichols, "A computational method for evaluating ui patterns," *arXiv preprint arXiv:1807.04191*, 2018.

[54] J. A. Fails and D. R. Olsen Jr, "Interactive machine learning," in *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003, pp. 39–45.

[55] E. Fast, B. Chen, J. Mendelsohn, J. Bassen, and M. S. Bernstein, "Iris: A conversational agent for complex tasks," in *Proc. CHI*. ACM, 2018, p. 473.

[56] X. Ferre, E. Villalba, H. Julio, and H. Zhu, "Extending mobile app analytics for usability test logging," in *Proc. INTERACT*. Springer, 2017, pp. 114–131.

[57] R. Fiebrink, "Machine learning as meta-instrument: Human-machine partnerships shaping expressive instrumental creation," *Musical Instruments in the 21st Century: Identities, Configurations, Practices*, pp. 137–151, 2017.

[58] Flurry, "Flurry — mobile app analytics platform for android & ios," 2023. [Online]. Available: https://www.flurry.com/

[59] A. Fourney, R. Mann, and M. Terry, "Query-feature graphs: bridging user vocabulary and system functionality," in *Proc. UIST*. ACM, 2011, pp. 207–216.

[60] FullStory, "Build a more perfect digital experience — fullstory," 2021. [Online]. Available: https://www.fullstory.com/

[61] Gmail, 2022, https://play.google.com/store/apps/details?id=com.google.android.gm.

[62] Google, "Analytics tools & solutions for your business - google analytics," 2023. [Online]. Available: https://marketingplatform.google.com/about/analytics/

[63] ——, "Gson," 2023. [Online]. Available: https://github.com/google/gson

[64] D. Gurari, Q. Li, C. Lin, Y. Zhao, A. Guo, A. Stangl, and J. P. Bigham, "Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 939–948.

[65] S. Hao, B. Liu, S. Nath, W. G. Halfond, and R. Govindan, "Puma: programmable ui-automation for large-scale dynamic analysis of mobile apps," in *Proc. MobiSys*. ACM, 2014, pp. 204–217.

[66] W. U. Hassan, S. Hussain, and A. Bates, "Analysis of privacy protections in fitness tracking social networks -or- you can run, but can you hide?" in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 497–512. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/hassan

[67] Heap, "Heap - illuminate the full user journey," 2023. [Online]. Available: https://heap.io/

[68] G. A. Help, "Find info about what's on your screen," 2019. [Online]. Available: https://support.google.com/assistant/answer/7393909

[69] E. Herder and H. Weinreich, "Interactive web usage mining with the navigation visualizer," in *CHI Extended Abstracts*, 2005, pp. 1451–1454.

[70] J. I. Hong, J. Heer, S. Waterson, and J. A. Landay, "Webquilt: A proxy-based approach to remote web usability testing," *ACM Transactions on Information Systems*, vol. 19, no. 3, pp. 263–285, 2001.

[71] J. I. Hong, Y. Agarwal, M. Fredrikson, M. Czapik, S. Hanna, S. Sahoo, J. Chun, W.-W. Chung, A. Iyer, A. Liu *et al.*, "The design of the user interfaces for privacy enhancements for android," *arXiv preprint arXiv:2104.12032*, 2021.

[72] Hotjar, "Hotjar: Website heatmaps & behavior analytics tools," 2023. [Online]. Available: https://www.hotjar.com/

[73] Z. Hu, Y. Ma, Q. Mei, and J. Tang, "Roaming across the castle tunnels: An empirical study of inter-app navigation behaviors of android users," *arXiv preprint arXiv:1706.08274*, 2017.

[74] F. Huang, J. F. Canny, and J. Nichols, "Swire: Sketch-based user interface retrieval," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–10.

[75] IFTTT, "Every thing works better together - ifttt," 2021. [Online]. Available: https://ifttt.com/

[76] S. Inc, "Snapchat," 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.snapchat.android

[77] U. Inc., "Usertesting: The human insight platform," 2023. [Online]. Available: https://www.usertesting.com/

[78] Instagram, "Instagram," 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.instagram.android

[79] J. Jeong, N. Kim, and H. P. In, "Gui information-based interaction logging and visualization for asynchronous usability testing," *Expert Systems with Applications*, p. 113289, 2020.

[80] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer, "Climbing the app wall: enabling mobile app discovery through context-aware recommendations," in *Proc. CIKM*. ACM, 2012, pp. 2527–2530.

[81] L. Kaufman and P. J. Rousseeuw, "Agglomerative nesting (program agnes)," *Finding Groups in Data: An Introduction to Cluster Analysis*, pp. 199–252, 2008.

[82] H. Kaur, M. Gordon, Y. Yang, J. P. Bigham, J. Teevan, E. Kamar, and W. S. Lasecki, "Crowdmask: Using crowds to preserve privacy in crowd-powered systems via progressive filtering," in *Fifth AAAI Conference on Human Computation and Crowdsourcing*, 2017.

[83] D. Kim, S. Park, J. Ko, S. Y. Ko, and S.-J. Lee, "X-droid: A quick and easy android prototyping framework with a single-app illusion," in *Proc. UIST*, 2019, pp. 95–108.

[84] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer, "Bricolage: example-based retargeting for web design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 2197–2206.

[85] R. Kumar, A. Satyanarayan, C. Torres, M. Lim, S. Ahmad, S. R. Klemmer, and J. O. Talton, "Webzeitgeist: design mining the web," in *Proc. CHI*, 2013, pp. 3083–3092.

[86] N. Labroche, M.-J. Lesot, and L. Yaffi, "A new web usage mining and visualization tool," in *Proc. ICTAI*, vol. 1. IEEE, 2007, pp. 321–328.

[87] V. V. Laptev, P. A. Orlov, and O. V. Dragunova, "Visualization of dynamic data structures with flow charts in web analytics," *St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems*, no. 4, 2017.

[88] S. Lederer, J. I. Hong, A. K. Dey, and J. A. Landay, "Personal privacy through understanding and action: five pitfalls for designers," *Personal and ubiquitous computing*, vol. 8, no. 6, pp. 440–454, 2004.

[89] H.-Y. Lee, L. Jiang, I. Essa, P. B. Le, H. Gong, M.-H. Yang, and W. Yang, "Neural design network: Graphic layout generation with constraints," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 491–506.

[90] U. Lee, J. Lee, M. Ko, C. Lee, Y. Kim, S. Yang, K. Yatani, G. Gweon, K.-M. Chung, and J. Song, "Hooked on smartphones: an exploratory study on smartphone overuse among college students," in *Proc. CHI*. ACM, 2014, pp. 2327–2336.

[91] L. A. Leiva, A. Hota, and A. Oulasvirta, "Enrico: A dataset for topic modeling of mobile ui designs," in *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, 2020, pp. 1–4.

[92] F. Lettner, C. Grossauer, and C. Holzmann, "Mobile interaction analysis: Towards a novel concept for interaction sequence mining," in *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*, ser. MobileHCI '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 359–368. [Online]. Available: https://doi-org.proxy2.library.illinois.edu/10.1145/2628363.2628384

[93] H. Li, X. Lu, X. Liu, T. Xie, K. Bian, F. X. Lin, Q. Mei, and F. Feng, "Characterizing smartphone usage patterns from millions of android users," in *Proc. IMC*. ACM, 2015, pp. 459–472.

[94] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Generating graphic layouts with wireframe discriminators," *arXiv preprint arXiv:1901.06767*, 2019.

[95] T. Li, G. Li, J. Zheng, P. Wang, and Y. Li, "Mug: Interactive multimodal grounding on user interfaces," *arXiv preprint arXiv:2209.15099*, 2022.

[96] T. Li, Y. Agarwal, and J. I. Hong, "Coconut: An ide plugin for developing privacy-friendly apps," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, pp. 1–35, 12 2018.

[97] T. J.-J. Li and O. Riva, "Kite: Building conversational bots from mobile apps," in *Proc. MobiSys*. ACM, 2018, pp. 96–109.

[98] T. J.-J. Li, A. Azaria, and B. A. Myers, "Sugilite: creating multimodal smartphone automation by demonstration," in *Proc. CHI*. ACM, 2017, pp. 6038–6049.

[99] T. J.-J. Li, M. Radensky, J. Jia, K. Singarajah, T. M. Mitchell, and B. A. Myers, "Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations," in *Proc. UIST*, 2019, pp. 577–589.

[100] T. J.-J. Li, L. Popowski, T. Mitchell, and B. A. Myers, "Screen2vec: Semantic embedding of gui screens and gui components," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi-org.proxy2.library.illinois.edu/10.1145/3411764.3445049

[101] Y. Li, J. He, X. Zhou, Y. Zhang, and J. Baldridge, "Mapping natural language instructions to mobile ui action sequences," *arXiv preprint arXiv:2005.03776*, 2020.

[102] Y. Li, G. Li, L. He, J. Zheng, H. Li, and Z. Guan, "Widget captioning: Generating natural language description for mobile user interface elements," *arXiv preprint arXiv:2010.04295*, 2020.

[103] Y. Li, Y. Guo, and X. Chen, "Peruim: Understanding mobile application privacy with permission-ui mapping," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 682–693.

[104] Y. Li, Z. Yang, Y. Guo, and X. Chen, "Humanoid: A deep learning-based approach to automated black-box android app testing," in *Proc. ASE*. IEEE, 2019, pp. 1070–1073.

[105] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing," in *Proceedings of the 2012 ACM conference on ubiquitous computing*, 2012, pp. 501–510.

[106] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling {Users'} mobile app privacy preferences: Restoring usability in a sea of permission settings," in *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, 2014, pp. 199–212.

[107] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, and H. Xiong, "Personalized mobile app recommendation: Reconciling app functionality and user privacy preference," in *Proc. WSDM*. ACM, 2015, pp. 315–324.

[108] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning design semantics for mobile apps," in *Proc. UIST*. ACM, 2018, pp. 569–579.

[109] W. LLC, "Whatsapp messenger," 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.whatsapp

[110] T. P. Ltd., "Tiktok," 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.zhiliaoapp.musically

[111] Y. Ma, X. Liu, M. Yu, Y. Liu, Q. Mei, and F. Feng, "Mash droid: An approach to mobile-oriented dynamic services discovery and composition by in-app search," in *Proc. ICWS*. IEEE, 2015, pp. 725–730.

[112] Y. Ma, Z. Hu, Y. Liu, T. Xie, and X. Liu, "Aladdin: Automating release of deep-link apis on android," in *Proc. WWW*. International World Wide Web Conferences Steering Committee, 2018, pp. 1469–1478.

[113] K. Martin and K. Shilton, "Putting mobile application privacy in context: An empirical study of user privacy expectations for mobile devices," *The Information Society*, vol. 32, no. 3, pp. 200–216, 2016.

[114] Maze, "Rapid, remote testing for agile teams," 2023. [Online]. Available: https://maze.co/

[115] Z. C. Meetings, 2022, https://play.google.com/store/apps/details?id=us.zoom.videomeetings.

[116] I. Meta Platforms, "Messenger," 2023. [Online]. Available: https://play.google.com/store/apps/details?id=com.facebook.orca

[117] Q. Metric, "Continuous product design — quantum metric," 2021. [Online]. Available: https://www.quantummetric.com/

[118] N. Micallef, E. Adi, and G. Misra, "Investigating login features in smartphone apps," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018, pp. 842–851.

[119] Microsoft, "2019 voice report: Consumer adoption of voice technology and digital assistants," 2019. [Online]. Available: https://about.ads.microsoft.com/en-us/insights/2019-voice-report

[120] Mixpanel, "Mixpanel: Product analytics for mobile, web, & more," 2023. [Online]. Available: https://mixpanel.com/

[121] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2018.

[122] D. K. Mulligan and J. King, "Bridging the gap between privacy and design," *U. Pa. J. Const. L.*, vol. 14, p. 989, 2011.

[123] S. Music and Podcasts, 2022, https://play.google.com/store/apps/details?id=com.spotify.music.

[124] Y. Nan, M. Yang, Z. Yang, S. Zhou, G. Gu, and X. Wang, "Uipicker: User-input privacy identification in mobile applications," in *Proc. {USENIX} Security Symposium*, 2015, pp. 993–1008.

[125] L. J. . B. News, 2022, https://play.google.com/store/apps/details?id=com.linkedin.android.

[126] E. Novak and C. Marchini, "Android app update timing: A measurement study," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 551–556.

[127] J. Novett, "Microsoft beats google to the punch: Bing for android update does what now on tap will do," 2019. [Online]. Available: https://venturebeat.com/2015/08/20/microsoft-beats-google-to-the-punch-bing-for-android-update-does-what-now-on-tap-will-do/

[128] T. S. of Mobile 2020, 2020, https://www.appannie.com/en/go/state-of-mobile-2020/.

[129] OpenAI, "Chatgpt," 2023. [Online]. Available: https://chat.openai.com/

[130] T. Orekondy, B. Schiele, and M. Fritz, "Towards a visual privacy advisor: Understanding and predicting privacy risks in images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3686–3695.

[131] T. Orekondy, M. Fritz, and B. Schiele, "Connecting pixels to privacy and utility: Automatic redaction of private information in images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8466–8475.

[132] S. Pareddy, A. Guo, and J. P. Bigham, "X-ray: Screenshot accessibility via embedded metadata," in *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 2019, pp. 389–395.

[133] Pendo, "Analytics — product and user journey analytics by user and account," 2023. [Online]. Available: https://www.pendo.io/product/analytics/

[134] R. Pimentel, "Chatito," 2021. [Online]. Available: https://github.com/rodrigopivi/Chatito

[135] Qualtrics, "Qualtrics xm // the leading experience management software," 2023. [Online]. Available: https://www.qualtrics.com/

[136] M. Rahman, ""working as intended" - an exploration into android's accessibility lag," 2016. [Online]. Available: https://www.xda-developers.com/working-as-intended-an-exploration-into-androids-accessibility-lag/

[137] E. M. Redmiles, S. Kross, and M. L. Mazurek, "Where is the digital divide? a survey of security, privacy, and socioeconomics," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 931–936.

[138] E. M. Redmiles, J. Bodford, and L. Blackwell, ""i just want to feel safe": A diary study of safety perceptions on social media," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, no. 01, 2019, pp. 405–416.

[139] G. D. P. Regulation, 2018, https://gdpr-info.eu/art-4-gdpr/.

[140] X. Rong, A. Fourney, R. N. Brewer, M. R. Morris, and P. N. Bennett, "Managing uncertainty in time expressions for virtual assistants," in *Proc. CHI*. ACM, 2017, pp. 568–579.

[141] A. S. Ross, X. Zhang, J. Fogarty, and J. O. Wobbrock, "Examining image-based button labeling for accessibility in android apps through large-scale analysis," in *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, 2018, pp. 119–130.

[142] ——, "An epidemiology-inspired large-scale analysis of android app accessibility," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 13, no. 1, pp. 1–36, 2020.

[143] I. S. Rubinstein and N. Good, "Privacy by design: A counterfactual analysis of google and facebook privacy incidents," *Berkeley Tech. LJ*, vol. 28, p. 1333, 2013.

[144] A. R. Sereshkeh, G. Leung, K. Perumal, C. Phillips, M. Zhang, A. Fazly, and I. Mohomed, "Vasta: a vision and language-assisted smartphone task automation system," in *Proc. IUI*, 2020, pp. 22–32.

[145] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *Proc. Ubicomp*. ACM, 2012, pp. 173–182.

[146] Sprig, "In-context user research — sprig," 2021. [Online]. Available: https://sprig.com/

[147] A. Support, "Use siri shortcuts," 2021. [Online]. Available: https://support.apple.com/en-us/HT209055

[148] A. Swearngin and Y. Li, "Modeling mobile interface tappability using crowdsourcing and deep learning," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–11.

[149] A. Swearngin, M. Dontcheva, W. Li, J. Brandt, M. Dixon, and A. J. Ko, "Rewire: Interface design assistance from examples," in *Proceedings of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–12.

[150] C. Tan, Q. Liu, E. Chen, and H. Xiong, "Prediction for mobile application usage patterns," in *Nokia MDC Workshop*, vol. 12, 2012.

[151] F. O. . P. Tracker, 2022, https://play.google.com/store/apps/details?id=org.iggymedia.periodtracker.

[152] M. C. P. Tracker, 2022, https://play.google.com/store/apps/details?id=com.lbrc.PeriodCalendar.

[153] L. Tran, D. Kong, H. Jin, and J. Liu, "Privacy-cnh: A framework to detect photo privacy with convolutional neural network using hierarchical features," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[154] Twitter, 2022, https://play.google.com/store/apps/details?id=com.twitter.android.

[155] Upwork, "Upwork — the world's work marketplace," 2023. [Online]. Available: https://www.upwork.com/

[156] UserZoom, "Userzoom — user research & ux testing solution," 2023. [Online]. Available: https://www.userzoom.com/

[157] Venmo, 2022, https://play.google.com/store/apps/details?id=com.venmo.

[158] D. Waisberg and A. Kaushik, "Web analytics 2.0: empowering customer centricity," *The Original Search Engine Marketing Journal*, vol. 2, no. 1, pp. 5–11, 2009.

[159] B. Wang, G. Li, X. Zhou, Z. Chen, T. Grossman, and Y. Li, "Screen2words: Automatic mobile ui summarization with multimodal learning," 2021. [Online]. Available: https://arxiv.org/abs/2108.03353

[160] D. Wang, J. Andres, J. D. Weisz, E. Oduor, and C. Dugan, "Autods: Towards human-centered automation of data science," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[161] S. Waterson, J. A. Landay, and T. Matthews, "In the lab and out in the wild: remote web usability testing for mobile devices," in *CHI Extended Abstracts*, 2002, pp. 796–797.

[162] S. J. Waterson, J. I. Hong, T. Sohn, J. A. Landay, J. Heer, and T. Matthews, "What did they do? understanding clickstreams with the webquilt visualization system," in *Proc. AVI*, 2002, pp. 94–102.

[163] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, "The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 1077–1093.

[164] R. Y. Wong and D. K. Mulligan, "Bringing design to the privacy table: Broadening," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 262.

[165] Workflow, "Workflow - powerful automation made simple," 2021. [Online]. Available: https://workflow.is/

[166] J. Wu, X. Zhang, J. Nichols, and J. P. Bigham, "Screen parsing: Towards reverse engineering of ui models from screenshots," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021, pp. 470–483.

[167] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, and J. P. Bigham, "Webui: A dataset for enhancing visual ui understanding with web semantics," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–14.

[168] Z. Wu, Y. Jiang, Y. Liu, and X. Ma, "Predicting and diagnosing user engagement with mobile ui animation via a data-driven approach," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–13.

[169] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *Proc. IMC*. ACM, 2011, pp. 329–344.

[170] B. Yan and G. Chen, "Appjoy: personalized mobile application discovery," in *Proc. MobiSys*. ACM, 2011, pp. 113–126.

[171] Q. Yang, A. Scuito, J. Zimmerman, J. Forlizzi, and A. Steinfeld, "Investigating how experienced ux designers effectively work with machine learning," in *Proc. DIS*, 2018, pp. 585–596.

[172] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan, "Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing," *IEEE transactions on information forensics and security*, vol. 13, no. 5, pp. 1317–1332, 2018.

[173] S. Zerr, S. Siersdorfer, J. Hare, and E. Demidova, "Privacy-aware image classification and search," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 35–44.

[174] T. Zhang, Y. Liu, J. Gao, L. P. Gao, and J. Cheng, "Deep learning-based mobile application isomorphic gui identification for automated robotic testing," *Ieee Software*, vol. 37, no. 4, pp. 67–74, 2020.

[175] X. Zhang, A. S. Ross, and J. Fogarty, "Robust annotation of mobile application interfaces in methods for accessibility repair and enhancement," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 609–621.

[176] X. Zhang, L. de Greef, A. Swearngin, S. White, K. Murray, L. Yu, Q. Shan, J. Nichols, J. Wu, C. Fleizach, A. Everitt, and J. P. Bigham, "Screen recognition: Creating accessibility metadata for mobile applications from pixels," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi-org.proxy2.library.illinois.edu/10.1145/3411764.3445186

[177] S. Zhao, J. Ramos, J. Tao, Z. Jiang, S. Li, Z. Wu, G. Pan, and A. K. Dey, "Discovering different kinds of smartphone users through their application usage behaviors," in *Proc. UbiComp*. ACM, 2016, pp. 498–509.

[178] Y. Zhong, T. Raman, C. Burkhardt, F. Biadsy, and J. P. Bigham, "Justspeak: enabling universal voice control on android," in *Proc. W4A*, 2014, pp. 1–4.

[179] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Mobile app recommendations with security and privacy awareness," in *Proc. KDD*. ACM, 2014, pp. 951–960.