Politecnico di Torino

Ingegneria Informatica

A.a. 2021/2022

Sessione di laurea Ottobre 2022

Using Feedback to Promote Meaningful App-Switching Suggestions

Relatori:

Luigi De Russis

Alberto Monge Roffarello

Candidato:

Matteo Moschelli

Acknowledgements

This thesis is the conclusion of a long and enriching journey, filled with satisfactions and joys. It seems only right to dedicate a few words to the people who have contributed, with their tireless support, to its realization.

First of all, a sincere thanks to my supervisors, Luigi De Russis (Politecnico di Torino) and Alberto Monge Roffarello (Politecnico di Torino), who followed me during all the phases of this work, for giving me the opportunity and trust to express myself to the fullest in writing the thesis.

I thank all the Friends that were at my side during these years, both those who shared their studies with me and those from my daily life. It was really important to know that I could count on someone in times of need.

I want to address a special thanks to Barbara, who proved an invaluable help during hard times, and a constant source of support and discussion in the last three years.

An affectionate thought goes to my grandmothers, who care about me above all else, and my late grandfathers, whose joy for the goals I reached I am sure would be unstoppable.

Finally, I greatly thank my parents, for always believing in me and supporting me through all my years of study and education.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In modern times, *smartphones* assumed a role of growing relevance and presence in our lives, providing not only a communication tool, but also various functionalities that are easily accessible. For example, users could read their emails and immediately insert and event on Calendar, while also chatting with a friend on Telegram.

Recently, a large literature production focused on analyzing smartphone usage and its consequences under different aspects. For example, the results reported in [1] highlighted connections between application categories (as defined by the Android Market), considering apps that are used during the same *phone usage session* (i.e., between a screen unlock and a screen lock), which can also be categorized based on location and time. Other researches, instead, focus on the feelings and experiences of users when interacting with the smartphone, and how to characterize addictive mobile apps and behaviors [2, 3].

Among all these study directions, one emerging and interesting field of research regards the so-called *app switching behavior*, i.e., "transitioning from one app to another in the same usage session to consume content" [4]. To this end, several works have shown the relevance and validity of this behavior, for example pointing out how the last used app is a strong indicator for discovering the next apps that the user is going to use [5], or analyzing various patterns that can characterize an application chain, with several revisitations of the same content inside a single session [6].

All these studies, however, proposed strategies and algorithms able to produce only high-level features that could characterize phone sessions by gathering data from different users and extracting off-line (i.e., not in real time) global usage patterns. In order to solve this problem, a new methodology for automatically extracting and characterizing app switching behaviors and design novel interactions to support these switches in modern smartphones was proposed, with the introduction of RecApps, a recommendation system capable of analyzing the user's smartphone usage, and then extract association rules representing habitual switching behaviors, using them as links to other apps [4].

Despite the good results in predicting the next app to use, based on contextual information shown by the methods and models proposed in previous works, they tend to not take into consideration information about the user, possibly disregarding their *digital wellbeing*, i.e., “a state where individual comfort is preserved despite an environment characterized by the overabundance of digital communication” [7].

Furthermore, given the ease with which users reflect about their usage habits, and also their will to take effective actions to monitor and control such behaviors, more care should be taken when designing the suggestion process of a recommendation system, in order to promote only those transitions that the user finds useful and meaningful, and limit the frequency of those that would bring unhealthy and addictive behaviors.

Ignoring this fundamental aspect may lead to the development of negative and unhealthy usage behaviors in users, and even consolidate them when reinforced and reiterated through a system that facilitates such app switches, since they would encourage the user to continuously and even mindlessly switch from one app to another.

1.1 Goal

The goal of this thesis is to design and implement a new approach for a *recommendation system*, which is able to improve users’ digital wellbeing while also proposing useful suggestions. The literature analysis and design phases were aimed at finding the most useful features of digital wellbeing apps, and adapt them to the new recommendation system. The main focus was on devising a strategy for integrating some information given by each user inside the recommendations extraction process, in order to better adapt the suggestions given to each user to their feelings and experiences.

The starting point for this new system was the *RecApps* mobile application, which consisted in a data analysis methodology based on association rules extraction, and a floating widget to show the resulting suggestions based on contextual information.

Given the objective nature of the existing collected information, an approach based on *proactive feedback* given by the user, and attached to each phone session, was considered, in order to guide the transition extraction procedure and include also some contribution of the user in the entire analysis.

The resulting improved suggestions should reflect users’ feedback and their perception of meaningful interactions and phone sessions, and thus leading to switching behaviors that do not reinforce addictive or unhealthy behaviors, but instead promote useful transitions, based on the contextual information that make the user mark a phone session as positive.

After the design and implementation phases, an in-the-wild user study was conducted to assess whether the adopted strategy and the resulting application were effective in supporting meaningful interactions and switching behaviors. Results were obtained through the use of objective metrics on the collected data, and also the analysis of the final interviews to the participants.

The analysis of the results, obtained through the use of objective metrics on the collected data, and also the analysis of the final interviews to the participants, allowed to come to the conclusion that considering users' feedback in the recommendation process is a useful strategy, since it allows them to control the suggestions they receive, and conduct more meaningful interactions with their smartphone.

1.2 Thesis Organization

This thesis is organized in the following chapters:

- Chapter 2 introduces the state of the art for what concerns app-switching systems, the initial version of RecApps and its internal mechanisms, and and overview about app to support digital wellbeing;
- Chapter 3 describes the design phase of the new version of RecApps, articulated through the analysis of previous literature about digital wellbeing apps and their useful features, the definition of the proposed solution of including proactive user feedback in the association rule mining procedure, and the description of the updated algorithm to implement into the application;
- Chapter 4 provides a report of how the chosen strategies were actually implemented inside the new version of RecApps;
- Chapter 5 describes the user study that was conducted after the development of the application was concluded, its characteristics and the kind of information that needed to be extracted;
- Chapter 6 provides the analysis of the results of the user study;
- Chapter 7 presents a final discussion about this work's results, its found limitations and potential changes and improvements.

Chapter 2

Background

This chapter provides an introduction about the main concepts addressed in this thesis: *app switching* techniques, which are the first goal of the newly implemented system, the behavior of the original *RecApps* application, that served as a starting point for such work, and *digital wellbeing*, which is the key argument that inspired the most important design decisions.

2.1 App Switching, Strategies and Tools

App switching behaviors (i.e., "transitioning from one app to another in the same usage session to consume content" [4]) provide an interesting tool for understanding dependencies between applications [8], by analyzing the apps that are used in a single *phone usage session* (i.e., between the screen unlock and the next lock).

One interesting aspect to consider is how these correlations can be extracted from recent usage data of the user, and analyzed to build a *recommendation system* to facilitate the switch between apps [5, 9, 10]. Another strong use for this kind of information is to preload an application in order to save time and resources when that app needs to be actually launched, like the model proposed in [11].

When dealing with the topic of app switching, there are several solutions and studies that provide ways to describe interactions and correlations among applications used in a single phone usage session (i.e., between the screen unlock and the next lock), like the algorithm implemented in [8]. Another interesting aspect to consider is how these correlations can be extracted from recent usage data of the user, and used to build a recommendation system to facilitate the switch between apps [11, 5, 9, 10].

2.1.1 Interactions Between Mobile Applications

Several recent works explored the topic of *correlations* between used apps in a single phone session, under different aspects. One interesting example is [8], which explored the differences in mobile usage sessions when they include a search engine application or not. With the use of two applications designed for the study, *AppLogger* to track usage events (e.g., app launches, turning the screen on and off, unlocking the phone, and accessing the home screen), and *MSearch* to embed a search engine and collect data about search queries and search interactions, interesting correlations were found between app usage and search interactions.

The results of this work have shown that sessions characterized by a mobile search tend to include more applications than those without a search interaction, and also that the applications used after the search interactions mostly involved actions related to the search (e.g., sharing content, links, or screenshots). Furthermore, search interactions seemed to derive directly from the content of the applications used before in the same session.

Another study that investigated how different apps are used together during the same usage session is [1]. With the installation of a background service as part of an existing application, a large amount of usage and contextual information was gathered from thousands of users, and information about apps was further characterized with the addition of a category (i.e., the same that was found for each application in the Android Market).

Then, statistics about *chains* of app usage (i.e. sequences of apps that are used during a period of time of at least 30 seconds, during which the smartphone did not go in standby mode) were extracted and analyzed. This allowed to observe some interesting patterns in the probability of transitioning from one category of applications to another, with *"Communication"* being the most frequent category users switch to, while coming from a different one. Some other notable unique connections include going from a *"Lifestyle"* application to a *"Shopping"* one.

Finally, another interesting finding is that, in the context of the same *phone session*, users tend to open again some apps that were previously opened, highlighting the usefulness of better supporting this kind of behavior with built-in mechanisms inside the smartphone.

2.1.2 App Prediction in Switching Behaviors

A *prediction* and *recommendation* system [11, 5, 9, 10] can help the users in reaching a desired app more quickly (they could be put in a more visible place, because the system recognizes that it is among the next apps that the user will use) or making it open faster (if the system is able to put them in a prelaunch state).

The problem that should be solved is an app prediction problem, whose general description could be stated as follows: "*Given a list of installed apps {a_{u1}, a_{u2},..., a_{u_n}} by a user u on his/her phone and the user's spatiotemporal context C, the problem of app usage prediction is to find an app a_{u_i} that has the largest probability of being used under C. Specifically, we aim to solve the problem:* $\max_{a_{ui}} P(a_{ui}|C, u), \forall i, 1 \le i \le n;$" [10]

A system that is able to answer to this kind of problem should also consider several aspects, in order to formulate predictions and take actions [5], summarized by these factors:

- the kind of *contextual information* that is relevant to characterize applications involved in the prediction procedure;
- the methodology used to extract said information and represent it as the context for a *phone session* (i.e., a series of applications);
- how different context sources can be combined together to *predict* app usage in an effective way.

The most common features that are used to describe application usage (and also group them into sessions, between a screen unlock and the next lock) are *location* and *time*, while features specific to the different proposed systems include *sensor signals* (e.g., accelerometer, gyroscope, light sensors) [11], *latest used app* (building on the idea that there should be a major correlation between two sequential used apps) and *user profile configuration* (which, in the authors' opinion, could give hints about some needs or mood of the user, or a special customization of the phone) [5], or *charge cable* and *audio cable* [10].

All the methods and models proposed in the analyzed works have shown good results in predicting an app based on contextual information, but they tend to not take into consideration information about the user, possibly disregarding their digital wellbeing. Moreover, the majority of the previous works exploited representational contexts, which are defined before the user interacts, thus not taking into consideration the current context and phone usage session.

2.2 The Original RecApps Algorithm

Despite a large production in HCI literature covering smartphone usage, there are no certain guidelines about how to *support common and habitual switching behaviors* on smartphones. The first step in this direction should address the problem of being able to detect and reproduce the typical app switching behaviors performed by each user.

While previous works mainly highlighted only high-level trends (for example, regarding the general category of the used apps and their connections), the work presented in [4] introduced RecApps (from *Rec*ommending *Apps*), an interactive floating widget with the goal of proactively supporting the user *transitions* from one app to another. Transitions are supported by showing *suggestions* about which apps the user could probably use next, each time a new app is opened on the smartphone.

The suggestions for the next app are computed periodically, in the form of *association rules* that match an *antecedent* app and *contextual information* (e.g., location, physical activity, time) to other *consequent* apps, in order to show those same apps when the context is met again.

Furthermore, the resulting recommendations are directly based on the *usage habits* of each user, instead of mixing together data from different users and analyzing off-line global usage patterns.

2.2.1 Data Collection and Phone Sessions

The first main step of the RecApps algorithm is the extraction of smartphone usage data, in terms of *phone-related information*, such as screen and app events, and other *contextual information*, like time or activity and location events. A *phone session* is temporally delimited by a pair of consecutive lock-unlock events, to identify the start and the end of each session, and these temporal boundaries are then used to isolate all the events related to the same phone session.

Table 2.1 contains the information that RecApps collects and considers when building a phone session.

The collected information is then transformed in a *transactional format*, i.e., a vector representation in which each entry associates the value 1 to the presence of a specific item (e.g., an app) in the corresponding transactional session, and the value 0 to the absence of an item in that session. Figure 2.1 shows a possible representation of a phone session in a transactional format.

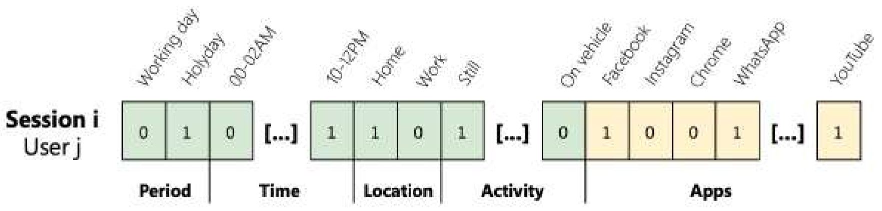


Figure 2.1: An example of a phone session represented in a transactional format.

Table 2.1: The different types of information that RecApps used to model the phone usage sessions.

Information	**Description**
Activity Events	Start/stop a given activity, i.e., *still*, *walking*, *running*, *cycling*, and *on vehicle*. Each activity event includes a timestamp, an activity, and the type of the event, i.e., start or stop.
Location Events	Enter/exit a given location area. The application forces users to define at least their *home* and *work* locations. Each location event includes a timestamp, a location, and the type of the event, i.e., enter or exit.
Screen Events	Lock/unlock the smartphone screen. Each screen event includes a timestamp and the type of the event, i.e., lock or unlock.
App Events	Open/close a given mobile app. Each app event includes a timestamp and the type of the event, i.e., open or close.

Note: From "Understanding and Streamlining App Switching Experiences in Mobile Interaction", by Alberto Monge Roffarello and Luigi De Russis, 2022, *International Journal of Human-Computer Studies* [4]

2.2.2 Smartphone Habits Extraction

The obtained dataset of phone sessions is then used as input for the *Apriori algorithm* [12] to mine and extract *association rules*, representing smartphone habits.

The Apriori algorithm employs a level-wise approach for generating association rules (each level represents the number of items in the consequent of the rule). Initially, only rules with one item in the consequent, and high enough confidence, are extracted, and are then used to extract new candidate rules for the next level.

For example, if $\{acd\} \rightarrow \{b\}$ and $\{abd\} \rightarrow \{c\}$ are high confidence rules for the first level, then the candidate rule $\{ad\} \rightarrow \{bc\}$ can be generated by merging the consequents of both rules.

For each level, the candidate rules are then pruned based on their *confidence*, avoiding to generate new candidate rules if said metric is too low.

As an example, if the confidence for $\{bcd\} \rightarrow \{a\}$ is low, then all the rules containing a in their consequent, including $\{cd\} \rightarrow \{ab\}$, $\{bd\} \rightarrow \{ac\}$, $\{bc\} \rightarrow \{ad\}$, and $\{d\} \rightarrow \{abc\}$ can be discarded [13].

The obtained results are then filtered to find promising association rules, by using common metrics for association rules evaluation (e.g., *support, confidence, lift*) in order to prune uncorrelated combinations and negatively correlated combinations. Furthermore, all habits that do not include an app in the antecedent are excluded, since they do not represent a proper switching behavior.

An example of smartphone habits that can be extracted through the RecApps methodology are shown in Figure 2.2, which present the way in which RecApps links the information about app usage and contextual information to the usage of the next apps.

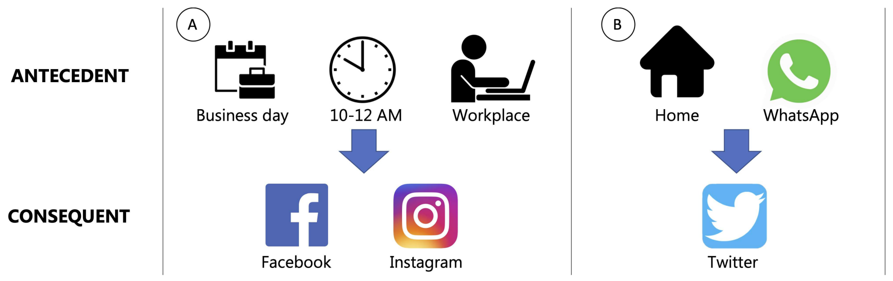


Figure 2.2: Two habits that can be found through the explained RecApps methodology. The habit (B) is a habitual app switching behavior between WhatsApp and Twitter that happens when the user is at home, while the habit (A) does not represent a switching behavior, since it does not include an app in the antecedent.

2.2.3 RecApps Implementation and Evaluation

The first developed version of RecApps was implemented as an Android app, employing the explained data analytic to find recurring switching behaviors in the user's smartphone usage sessions, and present them in the form of suggestions inside a *floating widget*.

The app continuously collects *usage info* in background and, when available, it associates contextual information to the created phone sessions. The information about physical activities (i.e., *still*, *walking*, *running*, *cycling*, and *on vehicle*) can be detected through the Google Activity Recognition APIs [14], while locations of interest are defined by the user at the startup of the application or in a later moment.

RecApps periodically employs the collected phone usage data to recalculate *association rules* that model *habitual app switching behaviors* of the user. The result of this computation is a set of *recommendations* that are then proactively shown to the user in a floating widget, when the *antecedent* app of a transition is opened and the other *contextual information* are verified to be the same as in the candidate transition. Each suggested app serves as a *shortcut* to open the corresponding application.

In [4], the authors explain the results of a three week in-the-wild user study, to evaluate the developed application and the data analytic method it implements in a real-world setting. During the first week, RecApps ran in the background, silently collecting data about phone usage sessions of the users. After obtaining enough data, the recommendations started to be computed, and for the remaining two weeks the widget started showing up, containing the most relevant suggestions, based on the context of each phone session. During the recommendations phase, the association rules were periodically recomputed, in order to take into consideration the most recent information that RecApps was collecting.

The metrics evaluated with the test, and the results of the final interviews with the participants of the experiment, had shown that RecApps was able to support a quick reproduction of habitual switching behaviors, while also producing rules that reflected less frequent habits.

2.3 Digital Wellbeing and technology overuse

The ubiquitous presence of smartphones in our daily routine has enabled an abundance of new opportunities for users, ranging from help in navigation and getting directions to real-time share of content and events. However, the pervasive use of smartphones in nearly every moment of our life but their increasing use represents also an increasing source of distraction, with an excessive use leading to problems for mental health and social interactions [15], or even social anxiety regarding overuse and dependence from such devices [16].

For this reason, HCI research is starting to focus on the topic of *intentional "non-use" of technology*, and in investigating and potentially avoiding *smartphone addiction*. Many mobile applications can provide tools for changing users' behavior with smartphones, and even Google and Apple have announced tools for promoting a more responsible use of smartphones. This brought Google to summarize its commitment with the term *"Digital Wellbeing"*:

"We're committed to giving everyone the tools they need to develop their own sense of digital wellbeing. So that life, not the technology in it, stays front and center." [17] [15]

Despite the available tools and their effects, many smartphone users recognize that they engage in compulsive and habitual behaviors that they later on find frustrating [16], making them wish to limit some aspects of their smartphone use [3]. For this reason, the new system presented in this work focused on the idea of promoting meaningful interactions for the user, in order to not bring them in an endless cycle of potentially useless app switches, eventually reinforcing addictive behaviors.

2.3.1 Digital Self-Control Tools and Habit Formation

In order to support and promote people's digital wellbeing, the last few years brought to life various examples of the so-called *Digital Self-Control Tools (DSCTs)*, with the goal of supporting users' self-control over their technology use, by allowing them to monitor and track their device usage, and also to apply some preventive actions (e.g., timers or lock-out mechanisms when certain time thresholds are reached). DSCTs are intended to help users improve their behavior with technology, and can serve as digital interventions for behavior change [18].

People often analyze themselves their excessive smartphone use and consider it problematic, and for this reason they are willing to try and adopt different strategies to mitigate these behaviors. One accessible solution is found in "digital wellbeing assistants" apps that can be easily downloaded and installed on the smartphone.

They offer several features, ranging from *tracking the general usage of the phone* to *monitoring the usage of other apps*. More in detail, the available features include *phone summaries* (i.e., a recap screen with statistics about the usage time of the smartphone), *app summaries* (i.e., a recap screen with statistics about the usage time of other applications), *home-screen widgets* and *e-mail summaries* to present information.

In addition to the basic features for tracking and observing usage data, *digital wellbeing apps* often offer *interventions* mechanisms to reduce addictive usage of phone and applications. Some of these tools include *app timers* or *app blockers*, in order to be notified when a certain threshold of use time is reached, or to directly block the usage of a given app under the same conditions.

Other relevant features that some apps include involve supporting users through *motivational quotes*, and rewarding them if they engage and succeed in some "digital wellebing challenge". These characteristics are seen as an effective strategy to support motivation of the user, by also adding metaphors and gamification principles to the promotion of healthier behaviors with the smartphone.

Furthermore, a few apps are capable of automatically extract information from user data regarding the most problematic apps or behavior, and instantiate interventions or even *redesign the phone UI*, to randomize the location of some apps in order to prevent opening and use out of habit of those apps.

Another aspect of the analysis conducted in [15] refers to the thoughts and comments of the interested users regarding digital wellbeing applications. From a general perspective, users like using digital wellbeing apps and find them useful tools, being more satisfied with using them the more they are accurate in tracking usage information such as screen time, unlocks, or app usage time.

Users also point out that these apps were found useful in many different use cases, ranging from studying and working, to parental control or sleep. On the topic of specific features of digital wellbeing apps, *restrictive actions* prove useful to control and monitor some unhealthy behaviors, which can also be identified through the use of *statistics*.

Another interesting aspect that was analyzed is how users consider their digital wellbeing apps as *self-monitoring tools*, that they can use to discover and comprehend how they use their smartphones. In this way, they can also improve their awareness of potential addictive behaviors, and thus they can employ these same tools to break unhealthy habits and learn how to use their smartphone in a conscious and responsible way.

While many users enjoy using digital wellbeing apps and appreciate their characteristics, there are other reasons that make users dislike these kinds of apps. One recurring example is that some of the interventions are not restrictive enough, making them permissive and ignorable, and bypassable in some ways. This brings users to propose other strategies to make the limitations imposed by digital wellbeing apps more difficult to be avoided.

Despite many users of digital wellbeing apps find these products a useful tool for tracking and monitoring potential unhealthy behaviors regarding their smartphone usage, they do not promote the *formation of new habits* [15].

The work presented in [19], however, follows the idea that applications and technology-based interventions represent a potential tool to provide real *habit support*, and also presents some design guidelines for interventions based on contextual cues and implementation intentions, to effectively provide habit formation.

Habits can be described as *automatic responses to contextual cues* (e.g., current location, routine events, or preceding actions). This association between the cues and the task form automatically through *repetition*, and can be used as *trigger* for events that should drive the behavior that will become the formed habit. This process can be directed in a more effective way by forming *implementation intentions*, which are *action plans* in the format *"When situation X arises, I will perform response Y"*. They help in connecting the forming habit with an existing routine and transform it into a *event-based task* (easier to remember than time-based tasks). The whole procedure builds on the idea that each repetition of the task, with an explicit relationship with its cues, will reinforce the association, leading to a more efficient action initiation and increased automaticity of the behavior.

Another decisive aspect for habit formation is *positive reinforcement*, which aims at increasing the feeling of *satisfaction* with each successful repetition of the habit action, eventually reinforcing the need to repeat said action in the future.

Given these founding elements in habit formation, the authors of [19] analyzed the features of several *habit formation apps*, under the aspects of *functionality* (i.e., which strategies they implement to promote habit formation), *habit formation support* (i.e., which elements of habit formation are explicitly implemented by the apps) and *behavior change techniques* (i.e., the mechanisms that each app employed to make users reinforce their new habits).

The functionality review that the author performed showed that the considered apps focus mainly on implementing features that support *self-tracking*, without any explicit design effort to support habit formation. Even if self-monitoring is crucial in the beginning of the habit formation process, it does not help users in forming associations between the task and the environment, or in improving the automaticity of performing a certain task.

Given these limitations of the existing habit formation apps, the authors proposed some guidelines for future design of similar apps:

- *Support trigger event*, in order to set a goal (the forming habit) and a trigger condition (e.g., "I will do X after eating breakfast"), also monitoring the completion of the objective;
- *Use reminders to reinforce implementation intentions*, by setting alerts and notifications before the trigger event, since they can help in form the association between the goal and the trigger, possibly encouraging users to remember it themselves;
- *Avoid features that teach users to rely on technology*, since having the users rely on a technological solution can interfere with the habit formation process, making it more difficult to form associations between contextual cues and the task.

2.3.2 Compulsive Phone Use and Addicting Apps

In the most recent years, the theme of *excessive phone use* has become a prevalent problem, and this often has brought a negative impact on social interaction and mental health [2]. Technology-related addictions have been classified as *"behavioral additions"*, since technological devices such as smartphones reinforce features that may lead to addictive tendencies.

Recent research has rarely focused on individual apps when addressing the problem of smartphone addiction. Moreover, problematic smartphone use was usually detected through the aid of self-reported questionnaires, and only recently some computational strategies started to be employed (e.g., *classification* by modeling device-level usage information, or *predictions* through app usage patterns analysis).

Some work instead focused on understanding addictive behaviors to individual apps, and model their features [2]. Results shown that the categories of apps that are considered the most addictive are *social* and *communication* apps, with *web browsers* as the next category in order. Interestingly, the addiction to a web browser app reflects the addiction to the kind of *content* it accesses (e.g., social websites).

The study also investigated the aspect of *withdrawal* from the use of applications, and it demonstrated how it is easier to uninstall a social app, but more difficult to do the same with communication apps, since users are more attached to the latter than to the former.

Finally, the same work also proposed four types of *usage features* of each app, that may help in identifying problematic addictive behaviors to said app. The choice followed the assumptions that users' addiction to an application grows with several aspects:

- the more *time* they use the app;
- the more *frequently* they open the app *compulsively*, instead of only reacting to notifications;
- the more heavily they are triggered by *external notifications* to use the app;
- the more *regularly* they use the app every day;
- the more *places* where they use the app.

As many users engage in *compulsive smartphone usage and checking* and find it frustrating, it may be interesting to investigate which are the *triggers* and situations that lead to compulsive use and also cause these behaviors to end [16].

Users independently tend to reflect on their compulsive smartphone usage and act by sometimes deleting apps that drive their compulsive checking without any necessary meaning.

Participants of the study conducted in [16] (which involved a *think-aloud usage session* for each user, where the most problematic apps were highlighted by them) explained that habitual phone checking fills almost every moment of *downtime*, often engaging in these behaviors without awareness. The cited work tried to define a set of *common triggers* that pull users in compulsive checking and another set of factors that lead them to stop the compulsive checking.

The most common triggering events for a compulsive phone checking behavior include:

- *Downtime*, defined as a moment of free time without any other evident source of stimulation or demand for active engagement for the user;
- *Tedious Tasks*, which make users check their phone compulsively or continuously due to the repetitive or tedious nature of the task they should be doing;
- *Social Awkwardness*, such as situations in which participants may find they have nothing to say, of feel embarassed about being idle, that bring them to check in with low-demand apps;

- *Anticipation*, an urge to check an app expecting some social or informational reward, with the same behavior repeating with short time intervals;
- *Nothing*, since often the compulsive checking of the phone seemed to not require a trigger to start, making users check the phone without any specific reason or condition.

On the topic of triggers that tend to end an instance of compulsive checking behavior, the study discovered what was called the *"30-Minute Ick Factor"*. This term summarized a recurring sense of disgust and regret after spending some time checking the phone without any particular goal or reason. Many participants cited durations close to *30 minutes* as cadence for moment of self-reflection that made them realize the mismatch between the excessive time investment and the relatively low sense of meaning, that made them break the usage session.

When combining the scope of *app switching* with the ones of *addictive apps* and *compulsive behaviors*, it is clear that a system which is able to support app transitions while also paying attention to the *digital wellbeing* of the users should try to make them recognize which are the apps that cause the most addictive behaviors, in order to not reinforce such negative patterns.

2.3.3 Meaningfulness in Smartphone Use

A large portion of past research production has stated that many people with they could limit some aspects of their interaction use. They often tend to judge their usage of the smartphone as *meaningless*, a choice that could be based on the intentions that drive them to use the phone, and the kind of use they make of it [3].

The *Uses and Gratifications* (U&G) perspective allows to analyze media usage and consumption as active choice of the users, which can be driven by the need of obtaining certain rewards or to reach specific goals. This allows to examine the motivations that draw people to use particular technological tools and medias.

From a general point of view, the typical gratifications that users seek from media can fall under two motivations: *instrumental* motivation, i.e., the one in which users engage with technology to obtain a specific result or goal, and *habitual* motivation, i.e., when users engage with technology out of habit to pass the time.

Following a similar approach, based on associations between certain U&Gs for different apps to the perceived degree of meaningfulness, the authors of [3] described some *common features* that help describing how users tend to label their experience with smartphones as meaningful or meaningless.

Among the most frequent characteristics of meaningful interactions, users report that using the smartphone for *productivity purposes* (i.e., "getting things done or self-improvement") makes it easier to associate it with a sense of meaning. Another instance of meaningful and positive associations involved *active communication*, often regardless of the "official" app category that was declared on the stores.

On the contrary, regarding meaningless interactions and usage, almost all participants reported *social media* as a meaningless category of applications to use. Some users, however, reflected on the fact that using social media in a more active way could bring the judgment to a meaningful result (e.g., using a social app to share some interesting content that was previously obtained). Similarly, also the *entertainment* category was sometimes seen as meaningless, since it provided a way to entertain mindlessly.

The same approach of evaluating the *meaningfulness* of a category of applications may be employed by the new recommendation system, by assigning a U&G to each potential transition, in order to promote those that represent a meaningful interaction for the user.

Chapter 3

Design and Method

This chapter describes all the design phases and choices that characterized the design of a new RecApps mobile application, with the goal of renewing and improving the data collection strategy and also to make it producing suggestions that could benefit the user's digital wellbeing.

After an initial analysis of the literature regarding digital wellbeing and app transitions strategies, a possible set of design directions to start from for the redesign of RecApps was defined. In the end, the choice for the main changes and additions to focus on, as reported in Table 3.6, was for designing and implementing a system to account for *user feedback* regarding usage sessions, in order to consider it when extracting *association rules* and *suggestions* for the user's app switches.

The last section of the chapter describes the adopted methodology and *algorithm* which exploits the feedback information, integrated inside the extracted recommendations, in order to produce the final set of apps that will be displayed inside the *floating widget* when an app is opened and a matching context is detected.

3.1 Literature Analysis and Design Directions

The first consideration comes directly from the analysis of the final interviews for the user study of RecApps [4], where participants reported that, sometimes, recommendations inspired some reflections regarding habitual behaviors that they wish they could avoid. To support this, it may prove useful to let the user customize the suggestions, so that some of them are always active, and others are prevented from being shown.

By taking inspiration from the aspects of *digital wellbeing apps* that make users like these tools [15], one feature that could be easily adapted to RecApps is that DSCTs allow users to control unhealthy behaviors. By working on the kind of suggestions that the app produces and shows to the users, it may make their app switching habits more *meaningful*, avoiding to get lost for long time periods using useless apps.

Although it may be observed that RecApps is not able to implement restrictive solutions by itself, this aspect does not represent a complete issue. Even if a DSCT that puts effective restrictions on the smartphone usage of the users is appreciated [15], it is also true that an exaggeration of these controls may lead to a too restrictive tool, and this increases the risk of users abandoning the use of the app itself [20]. This fact supports the idea that implementing a mechanism that controls and improves the quality of the proposed transitions in order to promote meaningful usage patterns is a good design direction, since it would provide some limitations without being too restrictive.

By analyzing the statement that "smartphones, in particular, have been found to be a source of distraction, and their excessive use can be a problem for mental health and social interaction" [15], one of the possible strategies that was considered involved identifying nearby *Bluetooth devices* (i.e., making them represent other smartphones and, therefore, other people), and stop showing suggestions in these situations. Similarly, the idea to limit suggestion to only apps with the same Uses & Gratifications [3] with respect to the current context (e.g., only *"productivity"* apps when at work), in order to reduce distractions and interference with daily activities and tasks that extend over time.

However, an arguable downside of these strategies is that they risk to heavily limit the amount of *suggestions* that are proposed to users, mainly due to the presumably constant presence of other smartphones near the one on which RecApps is running, even if there is no interaction between the two owners that may be hindered in receiving app suggestions.

Following the U&G perspective described in [3] (i.e., a description and classification of people's motivations for engaging with technology), one useful action may be to let the user set (at the startup of RecApps, and possibly periodically after the first time) which are the U&G they associate with the apps installed on the device, in order to consider this information along with the collected one when extracting the association rules.

From the same article, it may be useful to extract the concept of *"meaningfulness"* and exploit it to filter some of the suggestions, in order to have the users switch only towards apps they find useful. One example could be when dealing with a potential transition that has a social app in the antecedent or, especially, in the consequent. These apps are often seen as examples of *meaningless smartphone use*, but avoiding to show them at all may go against the opportunities of active communication that social network apps sometimes represent.

One of the most interesting aspects analyzed in [16], regarding the triggers that end compulsive smartphone use, is the *"30-Minute Ick Factor"*, which reflects a typical time duration after which users start to reflect about the waste of time that their last usage session represented. Based on this principle, one of the most promising strategies that was considered involved providing users with a sort of *“30-Minute Ick Factor Button”*, through which they can inform the system that their last phone session is not meaningful, and so it is to be excluded from the data collection and the resulting analysis.

On the contrary, it may also happen that the user finds meaningful some pattern that the system automatically labeled as negative, and as such would not be suggested in order to not reinforce it. To solve this potential problem, the user can be provided with a mechanism to set some exceptions to the decisions of the system, in order to not loose completely some transitions that the system may associate with compulsive behaviors (and as such would avoid to suggest). This follows the concept that, in the end, the *meaningfulness* of smartphone use is a subjective evaluation.

The last aspect that could be useful to analyze is the topic of *addictive apps* [2]. Through the data collected by RecApps, it may be possible to identify a priori which are the apps that provoke *addictive behaviors* by the user, and then the shortcuts to said apps may be disabled by default, and users may be able to manually enable them.

Moreover, following users’ impressions that withdrawing from using an application is easier than controlling the time spent on it, simply avoiding certain suggestions may go along the same principle. Despite not actively reducing the time the users spend on certain applications, this strategy should be able to not increase it.

Finally, the features used to describe *addictive apps*, and the assumptions from which they derive, all represent information that can be retrieved from the one that RecApps extracts from each phone session.

To sum up the main strategies that were considered as possible design directions to extend and improve the existing RecApps application:

- users could be able to *influence* the possible suggestions that they may be shown, by making some of them permanent and disabling some others;
- suggestions could be *disabled* in certain situations or *limited* to those matching a particular context;
- information regarding the *typical use* associated with each application should be specified by users, in order to be considered in the rule extraction process;
- suggestions should reflect the kind of usage that users find *meaningful*, in order to guide them towards apps that they find *useful*;
- users may be able to explicitly mark a phone session as *meaningless*, in order to exclude it from the data analysis procedure;

- users could set some *exceptions* to potential compulsive behaviors, provided they find them useful and meaningful;
- potential *addictive apps* could be identified through an analysis of the information collected by RecApps.

Most of these design ideas include a direct interaction by the users, or some other actions that involves them. This contrasts the old version of RecApps' data analysis and recommendation system, since it only exploits information that can be directly extracted by the smartphone alone.

3.2 User Feedback to Guide Transitions Extraction

The original behavior of RecApps did not account at all for the intervention of users inside the algorithm and in the operations that generated the *recommendations*, and the analyzed literature gave a large amount of design ideas that involved a direct contribution by the user.

For these two reasons, the final choice consisted in a union of some of the previous ones, in the form of a system able to collect *user feedback* and consider it during the *rule* extraction process and *suggestions* presentation step. The choice was made so that each user's contribution could help in better modeling their *habits*, and adapt the rule generation procedure in a more efficient way to each one. By considering these *feedback elements* inside the Apriori algorithm for rule generation, suggestions will not take into consideration only objective phone usage data, but also *emotions and feelings* of the users, with the goal of reinforcing those transitions that came from *positive sessions*.

To properly assess the role of user feedback inside the recommendation process, some aspects needed to be defined and analyzed:

- what *information* makes up the feedback?
- how to *ask for user feedback* and with which *frequency*?
- how to *integrate* the received feedback and the already available information to generate *association rules*?
- how to exploit the obtained result to improve *recommendations*?

3.2.1 Feedback Description and Retrieval

When designing the general characterization for the user feedback to include in the new recommendation system, one possibility that was explored was to relate each *feedback instance* to an entire phone usage session that has just ended. This was useful to associate feedback elements to the same “information aggregation unit” (i.e., a *phone session*) that was already collected and manipulated in the original RecApps application. Another possible motivation to support this implementation is that the user can figure out the whole picture when expressing its opinion.

The first fundamental aspect to consider and retrieve is a general assessment about the *meaningfulness* of the phone session. The definition of “meaningfulness” may not be expanded when asking the user a rating, since this may leave to the users the task of interpret the term in a personal way, and based on their life and experience.

The first design on how to retrieve this information about meaningfulness involved a *5-points Likert scale* (i.e., from *Not at all meaningful* to *Very meaningful*), based on the question “How much do you feel like you have spent your time on something meaningful?”. An alternative formulation, and the final choice, consisted in a *binary judgement* (e.g., good/bad), in order to be more immediate and easily characterize the content of a session, and the opinion of the user about it. Another motivation that led to prefer the second solution is that too many possible choices for the meaningfulness feedback may have caused the associations between apps and/or contextual information and the feedback element to be too weak, and thus produce less accurate transitions and suggestions.

Table 3.1 presents a brief overview on the considered possibilities to represent *meaningfulness* information, together with their analyzed advantages and disadvantages.

Table 3.1: A summary for the explored alternatives to extract information about Meaningfulness.

Proposed strategy	Advantages	Disadvantages
5-points Likert scale	More precise evaluation	Too many choices may cause the association rules to become too weak and less accurate
Binary evaluation	More immediate, easier to understand the opinion of the user about the session	More confusing about the link between the evaluation and the meaningfulness

A second interesting source of information about the user’s feedback on a phone session can be the U&G they associate the most to the entire phone session. According to [3], it is possible to define U&G for the *motivation* of the interaction with the smartphone (e.g., *instrumental* or *habitual*), and about the type of *usage* (e.g., *productivity*, *information*, *entertainment* and so on).

As a last aspect, also inspecting the kind of *emotions* that each user associates to the phone session may be a good way to better characterize each phone session, and improve the data analysis. The first set of possible emotions from which users could choose were inspired by P. Ekman's basic emotions (*anger*, *disgust*, *sadness*, *joy*, *fear*, and *surprise*) [21] and some of the secondary ones (*amusement*, *excitement*, *relief*, *satisfaction*, and *shame*). Given the large amount of choices that it would have been necessary to include to provide users with an equal set of positive and negative emotions, the emotions defined by R. Plutchik (*joy*, *sadness*, *trust*, *disgust*, *anger*, *fear*, *surprise*, *anticipation*) [22].

Despite the usefulness of having a limited set of emotions to choose from, in order to simplify any further analysis of the results, neither of the two alternatives seemed to be able to effectively describe the possible emotions that users could perceive while using their smartphones. Therefore, the final choice of possible emotions to include in the feedback retrieval procedure included *sociality* (connection with other people, social reciprocity, building relationships), *satisfaction* (both immmediate and temporary, and enduring, from things that may come useful in the future), *regret* (the same concepts of the "30-Minute Ick Factor") and *boredom* (described as a trigger behavior for compulsive smartphone usage) [16]. Some of these alternatives found some confirmations also in another previous work, that associated smartphone usage features and their patterns with the possible emotions that users could feel [23].

The overall considerations about each possible strategy that was examined in order to define the set of choices about the emotions felt during the phone sessions, along with advantages and disadvantages for each option, is presented in table 3.2.

The final composition of *feedback features* and their possible values that was chosen includes:

- *EMOTION*, with values *sociality*, *satisfaction*, *regret*, *boredom*, and the possibility to enter a custom value, and re-use it for future sessions;
- *USAGE*, with values *productivity*, *information*, *entertainment*, *communication*, and *social media*;
- *OVERALL*, with values *positive* and *negative*.

Table 3.2 reports a brief summary of the considered options for describing the content of the user feedback regarding a recent phone session, along with their analyzed advantages and disadvantages.

The way of retrieving all the chosen information that has been considered from the beginning involved the compilation of a short *survey*, divided in three sections (i.e., one for each type of information to retrieve). Along with the *interface* to show the questions and the possible options, also some information regarding the past

Table 3.2: A summary for the explored alternatives to extract information about Emotions.

Proposed strategy	Advantages	Disadvantages
Ekman's basic emotions	An large set of options to describe emotions	Too many options required to balance "positive" and "negative" emotions
Plutchik's emotions	A balanced set of emotions, easier to analyze and form correlations	Inaccurate to describe feelings of users about smartphone usage
Choices from literature	Taken from studies that analyzed people's reactions and experiences when using their smartphones	Limited set of choices, custom option needed

phone session (e.g., the used apps) should be provided, in order to allow the user to keep some context regarding the session for which the feedback is retrieved.

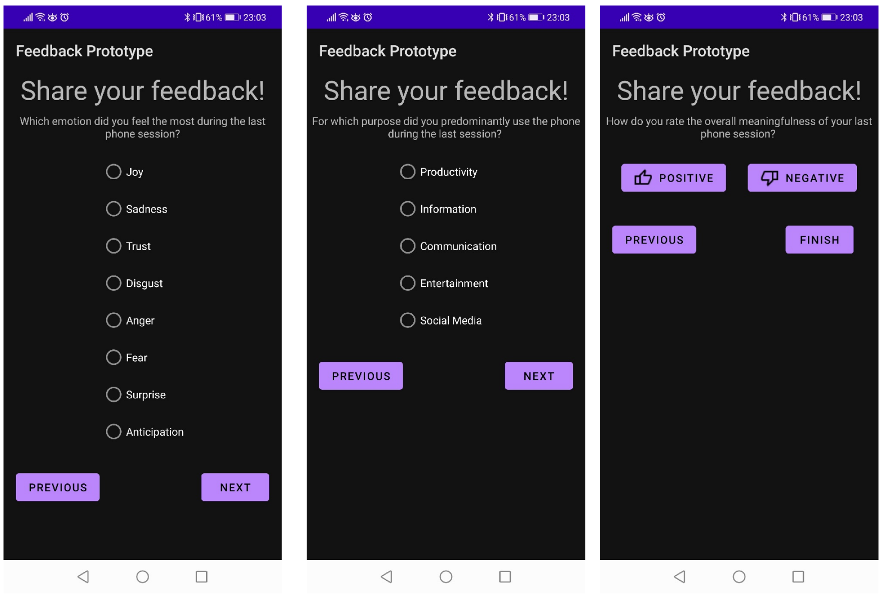
Apart from an initial screen to present the feedback procedure and give the user the possibility to skip the feedback gathering for the last session, two alternative designs were considered for the possible organization of the interface and presentation of the options.

The first possible layout involved a separate screen for each feedback element to retrieve. This disposition allows the user to separately answer each question, and visualize the full set of possible answers each time, but of course the time spent each time the form is shown is higher (the result is shown in Figure 3.1).

The other disposition placed all the form inputs inside a single screen, making it more compact and therefore quicker to compile, but in order to properly arrange the elements in the available space, the options needed to be collapsed inside other form elements, and so in this way the users could not see all the available answers (the result is shown in Figure 3.2).

Table 3.3 gathers the two possible alternative designs that were considered for the *feedback form interface*, which should be presented to users in order to retrieve all the feedback information.

In the end, the version of the feedback form design that was chosen as a starting point for the actual implementation was the first one, for the reason that having more dedicated screen space for each kind of feedback elements would have allowed to enrich the presentation of the possible options (e.g., with some *icons* to associate to each choice).



Which emotion did you feel the most during the last phone session?

For which purpose did you predominantly use the phone during the last session?

How do you rate the overall meaningfulness of your last phone session?

(a) Emotion **(b)** Usage **(c)** Overall

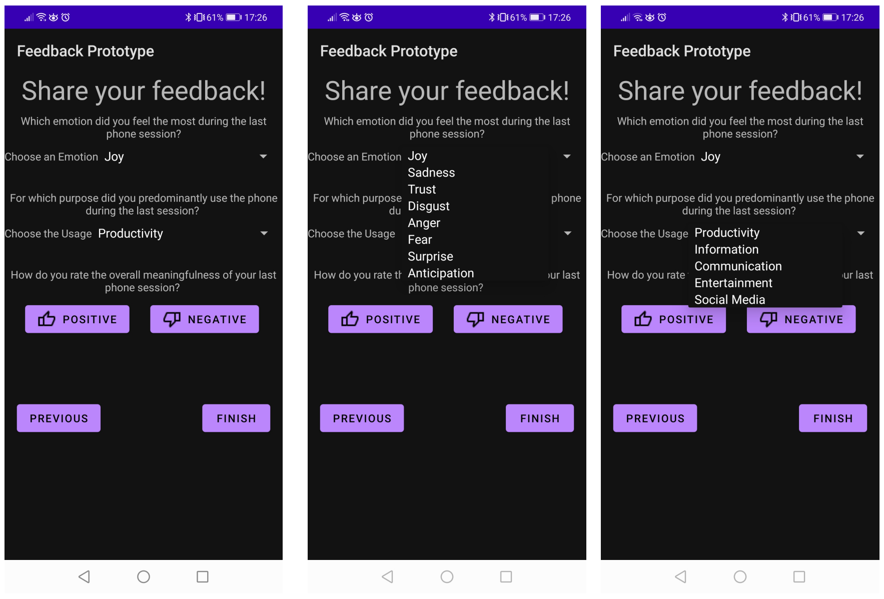
Figure 3.1: The first alternative version of the feedback form, in which each feedback element is retrieved in a different screen.

3.2.2 Feedback Form Proposition

After deciding the layout for the *feedback form*, another fundamental aspect for the user feedback system addressed the *frequency* and the way in which the form was shown to the user. Although asking a feedback to the user at the end of each

Table 3.3: A summary for the explored alternatives to arrange the feedback form interface.

Proposed strategy	Advantages	Disadvantages
Separate tabs	Each section has more available space to present all the options, even adding icons to describe each choice	It takes longer to compile the form, or eventually to skip until the end to avoid giving feedback
Single tab	More compact and quicker to compile	The user do not immediately see all the options for the emotion and usage sections


Feedback Prototype
Share your feedback!
Which emotion did you feel the most during the last phone session?
Choose an Emotion
Joy
Sadness
Trust
Disgust
Anger
Fear
Surprise
Anticipation
For which purpose did you predominantly use the phone during the last session?
Choose the Usage
Productivity
Information
Communication
Entertainment
Social Media
How do you rate the overall meaningfulness of your last phone session?
POSITIVE
NEGATIVE
PREVIOUS
FINISH

(a) General appearance **(b)** Emotion **(c)** Usage

Figure 3.2: The second alternative version of the feedback form, in which all the form input elements are placed inside the same screen.

phone session could enrich each session with additional information, it would result in an annoying behavior from the user's point of view.

One partial solution could be to introduce a *probability* of showing the feedback form, leaving the user with the possibility of changing said probability (i.e., increasing or reducing the frequency of feedback collection).

Moreover, it could be useful to identify some sessions that represent *candidates* to receive user feedback, in order to restrict the feedback collection procedure to only these phone sessions. Such sessions may be found in two ways:

- consider *usual or representative sessions*, i.e., with a *duration* close to the *average duration* of some sessions (this average could be computed considering the day of week and the time slot);
- find the phone sessions that match some of the already computed *association rules*, and collect feedback about those sessions, in order to assign feedback to what the system has already produced (this would require to change the starting time of the transitions computation, moving it earlier during the first week of data collection).

The first approach would imply to wait at least a week before starting the feedback collection (since an entire week is needed to compute an average duration for each day and time slot), making the first transitions ignore any possible feedback from the user. The second approach, instead, allows to start the feedback extraction earlier, making also the first recommendations benefit from the feedback and the resulting rules.

Addressing the topic of how to make users receive the feedback form for compilation, several alternatives were considered:

- the users could *autonomously* reach the interface for feedback collection and compile it, however the risk is that users are not explicitly encouraged and prompted to provide feedback, and it may happen that the feedback is given for an ongoing session, loosing part of its potential future context;
- the form may appear in a *blocking way* as soon as the phone is unlocked (i.e., at the start of the next phone session), but this may result in an annoyance for the user, in case they need to repeatedly deny the feedback form;
- a *notification* may inform that the user is able to compile the feedback form for the last terminated phone session, and the user can use it to access the form itself.

Table 3.4 shows the overview of the considered strategies to employ in order to determine *how often* the form should be presented to the users, and with which *mechanisms*.

The chosen configuration included the *blocking strategy* for showing the form, and the *probability* strategy to limit the number of times the form is shown. Moreover, to compensate for a potentially low number of sessions with feedback, a procedure to *propagate the feedback* to past sessions with the same context (i.e., apps and contextual information) was implemented.

3.2.3 Feedback Integration, the New Algorithm

After the users finishes the compilation of the *feedback form*, the information gathered in this way need to be included among the attributes that characterize a phone session, together with the *used apps* and other *contextual details* (i.e., time, period, activity, and location), as shown in Figure 3.3.

To achieve this result, the collected data will be transformed in the same *transactional format* that RecApps uses to express the available information as input for the *Apriori algorithm*, i.e., an array of `NamedItem` objects, each one consisting of a label to represent its value and another one to distinguish its type (e.g., `TIME` or `FEEDBACK_EMOTION`).

Next up is the *transition* generation procedure, which needs to be revised to consider new criteria for filtering the extracted *association rules*, since it now takes into account also the *feedback elements*. As a general setting, elements which

Table 3.4: A summary for the explored alternatives to arrange the feedback form frequency and the strategy with which it is presented to the user.

Feedback form frequency		
Proposed strategy	Advantages	Disadvantages
At the end of each session	All the sessions are enriched with feedback information	The task of giving feedback to each phone session become tedious for the user
Probability of showing after a session	Being less frequent, the user will not grow weary of being asked to give feedback	If the user sets a low probability, very few session will receive feedback, which needs to be propagated in order to be more frequent
Feedback form proposition		
Proposed strategy	Advantages	Disadvantages
Manual insertion by user	Does not require any kind of proposition by the system	It is easier to ignore the new feedback aspect of the system
Blocking proposition in the next session	The user is always presented with the request to compile the feedback form	The user needs to explicitly refuse to complete the form
Notification for inserting feedback	It is not intrusive, so it could be always asked, does not require explicit actions to deny the feedback insertion	Each notification is related to one session, so each time a new notification is shown, the previous one is lost

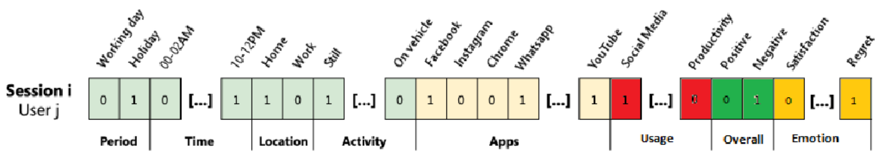


Figure 3.3: An example of the new phone session, integrated with feedback information from the user, represented in a transactional format.

describe the *meaningfulness* of the session/transition will appear in the *consequent* of the rule, while the other two kinds of information (i.e., about the *emotion* and the *usage*) will enrich the rule's *antecedent.*

This means that rules containing a feedback element of type `FEEDBACK_OVERALL` in their antecedent and those that include an element of type `FEEDBACK_EMOTION` or `FEEDBACK_USAGE` in the consequent need to be excluded.

Another interesting feature that was designed to further exploit the information retrieved through user feedback was the computation of some special *"feedback rules"*, i.e., association rules that considered only feedback information extracted from phone sessions. This allowed to take into account the most frequent *correlations* that each user made between elements of type `FEEDBACK_EMOTION` or `FEEDBACK_USAGE` with the general judgement, of type `FEEDBACK_OVERALL`.

The result of this phase will produce improved *transitions*, that will include information regarding the feedback that the user associated to previous *phone sessions*, and that was considered to be frequently associated to other elements like applications and contextual information (an example of the renewed transitions generation procedure is given in Figure 3.4).

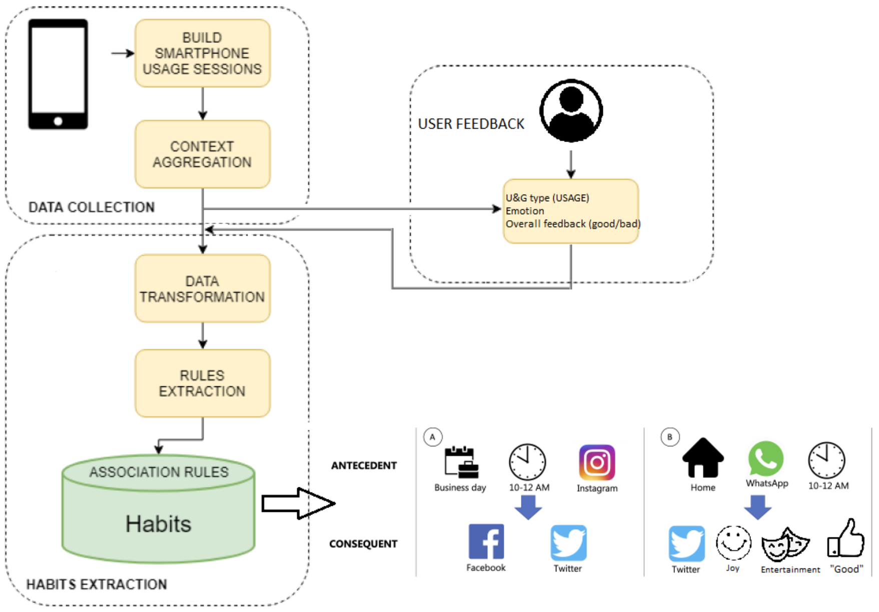


Figure 3.4: The complete data analysis procedure, with the integration of user feedback and the new extracted transitions.

After defining the details regarding how the new feedback information had to be integrated with the pre-existing information, a new strategy for showing suggestions to users, based on an algorithm that took into account also the *feedback*, had to be designed. To this end, two alternative solutions were considered and analyzed, based on two different usages for user feedback.

The first hypothesis was based on the "traditional" rule extraction phase, and exploited the eventual feedback elements included inside the transitions to implement a *sorting criterion* for evaluating the *relevance* of each proposed transition. The sorting operation and decision about which apps to include as suggestions inside the widget are performed after the set of potential transitions (i.e., the transitions that reflect the app context and the other contextual information) is computed, in order to restrict the decisions only on those transitions that could be effectively used and shown.

The *"relevance score"* starts at 0 by default, and is computed based on these criteria:

- the value of the `FEEDBACK_OVERALL` element determines a contribution of $+1$ (if *Positive*) or -1 (if *Negative*);
- the value of the `FEEDBACK_EMOTION` element determines, based on the consequent of the eventual feedback rule in which it is involved, a contribution of $+1$ (if the consequent is *Positive*) or -1 (if it is *Negative*);
- the value of the `FEEDBACK_USAGE` element determines, based on the consequent of the eventual feedback rule in which it is involved, a contribution of $+1$ (if the consequent is *Positive*) or -1 (if it is *Negative*).

After the *score* is assigned to each potential transition, those transitions with a negative score are automatically excluded from being shown on the widget, while the others are sorted, based on their score, and passed to the widget in the same order. This allows to eventually limit the number of icons to show inside the widget, and thus propose only the most relevant transitions for the user, under a certain context.

As an example, suppose that, after the execution of the rule extraction procedure, the following transitions:

$$\begin{gathered}
T_1 : \{WhatsApp, Home, 10-12, Entertainment\} \rightarrow \{YouTube, Positive\},\\
T_2 : \{WhatsApp, Home, SocialMedia, Satisfaction\} \rightarrow \{Instagram, Positive\},\\
\vdots\\
T_N : \{WhatsApp, 10-12, SocialMedia, Regret\} \rightarrow \{Facebook, Negative\}
\end{gathered}$$

and the following feedback rules:

$$\begin{gathered}
FR_1 : \{Entertainment\} \rightarrow \{Positive\},\\
\vdots\\
FR_{M-1} : \{Regret\} \rightarrow \{Negative\},\\
FR_M : \{SocialMedia\} \rightarrow \{Negative\}
\end{gathered}$$

are computed.

When the user opens the WhatsApp app on his smartphone, while at home at 11:00 (which is considered to be inside the time slot $10-12$), a set of "active" transitions is extracted, and those transitions undergo the ranking procedure through the computation of the relevance score.

Transition T_1 receives a +1 for the *Positive* element in the consequent, and a +1 for the association FR_1 between *Entertainment* and *Positive.* Transition T_2 receives a +1 for the *Positive* element in the consequent, and a −1 for the association FR_M between *Social Media* and *Negative.* Finally, transition T_N receives a −1 for the *Negative* element in the consequent, a −1 for the association FR_{M-1} between *Regret* and *Negative*, and a −1 for the association FR_M between *Social Media* and *Negative.*

Summing up, the possible apps that could be included in the widget, and their relevance scores are:

$$A_1 : YouTube \rightarrow +2,$$
$$A_2 : Instagram \rightarrow 0,$$
$$A_3 : Facebook \rightarrow -3$$

which means that Facebook will be immediately removed from the possible apps to show in the widget, while YouTube and Instagram will be passed to the widget instance and placed inside it (eventually, if a maximum size of 1 is specified, only YouTube will be shown, since it has a greater relevance score).

The second possible alternative exploits the feedback elements to *prune* the sessions from which to extract association rules, in order to preserve only those that were *meaningful* to the user. The general procedure includes two steps:

- computing a *relevance score* for each phone session, by weighing the eventual values of each category of feedback element, obtaining:

  $$S = W_E * R_{E_i} + W_{U_i} * R_U + W_O * O_i,$$

  where W_E, W_U and W_O are the weights associated with each *feedback element*, R_{E_i} is a contribution based on the eventual consequent of a *feedback rule* involving emotion i (+1 if *Positive* or −1 if *Negative*), R_{U_i} is a contribution based on the eventual consequent of a *feedback rule* involving usage i (+1 if *Positive* or −1 if *Negative*), and O_i is a contribution based on the value of the `FEEDBACK_OVERALL` element i (+1 if *Positive* or −1 if *Negative*);

- applying a *selection criterion* (e.g., a score *threshold* of preserving the *first N sessions*) to form a set of phone sessions that serve as the input for the *Apriori algorithm.*

This approach suffers from the fact that, given that phone sessions with similar apps and context, but not all with some feedback associated to them, will receive different scores, it is possible that some of these sessions will be included in the final set for the rule extraction phase, and some others will not. Therefore, this may cause some possible habits to not be frequent enough (i.e., with high enough confidence) to be included in the set of association rules that will then generate the transitions.

For this reason, the solution to *propagate the feedback* given by the user to one phone session to all other *similar past sessions* (i.e., with the same *used apps* and the same *context*) was introduced.

Another important aspect that this strategy would imply refers to the choice of the *weights* W_E, W_U and W_O, to be multiplied by each feedback contribution, in order to produce the final score S.

When making a comparison between the two different approaches, the second one seems to make a lesser use of the user feedback information, since in that case it act more as a filtering mechanism than a way to improve and guide the recommendations. Moreover, reducing the number of available phone sessions for the Apriori algorithm may reduce as well the variety of associations and transitions that are extracted (e.g., a discarded phone session could have reinforced one or more habits that involve some of its used apps).

The first algorithm, instead, focuses more on exploiting all the available information from the retrieved phone sessions, and only after use this new information to perform more accurate suggestions. More in general, this first solution focuses on promoting transitions that reflect meaningful interactions for the user, while the second one builds upon the phone sessions that the users find more useful.

The definitive alternatives that were analyzed to choose the version of the *"feedback algorithm"* to use in order to include the feedback information in the transition extraction phase are exposed in Table 3.5.

Table 3.5: A summary for the explored alternatives to integrate feedback information inside the computed transitions and recommendations.

Proposed strategy	Advantages	Disadvantages
Ranking the transitions	Focuses on the relevance of the suggested transitions	Does not directly influence the existing algorithm to generate transitions
Pruning the phone sessions	Only the most meaningful phone sessions are considered in the Apriori algorithm	Risk to exclude too many phone sessions from the rule extraction phase

In the end, the final decision was to implement the first alternative for the feedback algorithm, in order to focus more on the meaningfulness of the proposed switches between apps, rather than only on using the best phone sessions according to the user to produce transitions. Furthermore, this choice allows to preserve the entirety of the phone sessions, and exploit more data to detect frequent patterns, whether they reflect good habits for the user or not.

Table 3.6 shows a summary of all the different alternatives that were considered for all the aspects of the user feedback implementation in the new recommendation system, also providing a report of all the confirmed choices.

Table 3.6: A summary of all the design alternatives for various aspects of the new feedback algorithm. In **bold** all the options that were considered in the implementation phase.

Meaningfulness assessment		
Proposed strategy	Advantages	Disadvantages
5-points Likert scale	More precise evaluation	Too many choices may cause the association rules to become too weak and less accurate
Binary evaluation	More immediate, easier to understand the opinion of the user about the session	More confusing about the link between the evaluation and the meaningfulness
Emotions		
Proposed strategy	Advantages	Disadvantages
Ekman's basic emotions	An large set of options to describe emotions	Too many options required to balance "positive" and "negative" emotions
Plutchik's emotions	A balanced set of emotions, easier to analyze and form correlations	Inaccurate to describe feelings of users about smartphone usage

Choices from literature	Taken from studies that analyzed people's reactions and experiences when using their smartphones	Limited set of choices, custom option needed
	Feedback form interface	
Proposed strategy	Advantages	Disadvantages
Separate tabs	Each section has more available space to present all the options, even adding icons to describe each choice	It takes longer to compile the form, or eventually to skip until the end to avoid giving feedback
Single tab	More compact and quicker to compile	The user do not immediately see all the options for the emotion and usage sections
	Feedback form frequency	
Proposed strategy	Advantages	Disadvantages
At the end of each session	All the sessions are enriched with feedback information	The task of giving feedback to each phone session become tedious for the user
Probability of showing after a session	Being less frequent, the user will not grow weary of being asked to give feedback	If the user sets a low probability, very few session will receive feedback, which needs to be propagated in order to be more frequent
	Feedback form proposition	
Proposed strategy	Advantages	Disadvantages
Manual insertion by user	Does not require any kind of proposition by the system	It is easier to ignore the new feedback aspect of the system

Blocking proposition in the next session	The user is always presented with the request to compile the feedback form	The user needs to explicitly refuse to complete the form
Notification for inserting feedback	It is not intrusive, so it could be always asked, does not require explicit actions to deny the feedback insertion	Each notification is related to one session, so each time a new notification is shown, the previous one is lost

	Feedback integration strategy	
Proposed strategy	Advantages	Disadvantages
Ranking the transitions	Focuses on the relevance of the suggested transitions	Does not directly influence the existing algorithm to generate transitions
Pruning the phone sessions	Only the most meaningful phone sessions are considered in the Apriori algorithm	Risk to exclude too many phone sessions from the rule extraction phase

Chapter 4

Implementation

This chapter summarizes the results of the implementation phase for the new recommendation system, which focused on adapting the previously designed feedback strategy to the existing RecApps mobile application.

After an initial introduction of the main original components of RecApps, the implementation steps needed for the *feedback* collection and the *algorithm* to include the extracted feedback in the recommendation procedure are explained. Finally, the last section presents some further additions that were introduced in RecApps to better integrate the newly designed system in the application.

4.1 The Original RecApps Structure

RecApps is implemented as an Android mobile application, written in Java. It provides a simple user interface to interact with the main aspects of the underlying system (e.g., the recent *recommendations* or the last visited *locations*), modeled through the use of a `MainActivity` class that hosts several other Fragments.

4.1.1 Classes and Database Entities

The main information that is retrieved and managed inside RecApps is represented through Java classes and their corresponding Entities in the internal database.

The `PhoneSession` class brings together information regarding all the contextual information and app events that were detected between the start and end timestamps, including:

- an array of `AppEvent` objects, representing the *used apps* inside the session;
- an array of `NamedItem` objects, a class that represents the collected information in a proper format for applying the *Apriori algorithm*;

- two timestamps for the start and end of the *phone session* (corresponding to the two last detected `PhoneEvent` objects);
- a series of strings representing *contextual information* (i.e., location, activity, time, and period).

The extracted *association rules* and their representation as *app transitions* are contained into objects of the `AppTransition` class. This class is characterized by the same set of potential *contextual information* that is contained in a *phone session* (i.e., to represent the context under which a transition is considered to be active), an *app context* (i.e., the application that should trigger the recommendation of the transition) and the *rule consequent*, in the form of a `HashMap` object to associate each potential app to a flag that determines which app was used, if any.

All the presented classes and their instances are collected and stored inside the internal *database* of the application, implemented through the Room Persistence Library [24]. All the classes that represent persisted data are labeled as Entities, in order to be mapped to a table in the database (each instance of an Entity class is treated as a row in the corresponding table).

The `AppDatabase` class offers a static singleton instance, which is used to retrieve, store and manipulate the maintained data entities. The access to the content of a single table is made possible through the use of a *DAO (Data Access Object)*, an interface whose methods are mapped to *queries* in the underlying database table. The `AppDatabase` instance implements the methods that are invoked by the application, which in turn invoke the proper method of the corresponding DAO instance.

4.1.2 Listeners and services

A Listener is an instance of the `BroadcastReceiver` class, whose role is to “wait” for external messages, whether they are produced by the operative system or by other applications.

The kind of messages that RecApps declares the intention to receive are related to GPS, WiFi and screen events.

The `ScreenListener` class is the component that detects, in the `onReceive()` method, a *screen event* (i.e., a screen lock or unlock). When such an event is notified, based on the type of action (i.e., `ACTION_SCREEN_ON` or `ACTION_SCREEN_OFF`) a `PhoneEvent` of the appropriate type is created and stored inside the local database . In the case of a screen lock event, a `PhoneSession` object is instantiated and its attributes are assigned based on all the app and contextual events that were detected during the usage period.

A Service is an application component that performs operations in the background, without providing any user interface. Inside RecApps, there are three types of operations that are performed through services:

- current location and physical activity recognition;
- periodic upload of the collected data on Cloud Firestore and computation of association rules;
- accessibility events detection and transition management.

The core service and component of the application is represented by the `AppMonitor` service. This class is in charge of registering the listeners when it gets connected, refreshing the current *geofences* (i.e., a feature of the Android location APIs to monitor user location and proximity [25]), and intercept the opening of a new application and check if there are *transitions* that can be shown inside the floating widget. This last operation is performed in the `onAccessibilityEvent()` method, which is triggered after a new application reaches the foreground. Then, all the transitions that have the last opened app as the *antecedent* are retrieved, and a search for the active ones is made. These transitions are passed to the *widget* instance in order to show their icons as links to perform an app switch.

4.1.3 Utility Features and Methods

The `StatsManager` static class implements the `buildPhoneSession()` method, which is used to retrieve all the collected information regarding *apps* and *contextual data* that was detected between the start and end timestamps.

Another interesting class, that provides several utility methods, is the `Utils` class. Some examples include the `getFirstCalculationTimestamp()` method, to retrieve the timestamp of the first execution of the *Apriori algorithm* for extracting association rules (i.e., one week after the installation of RecApps), and the `getTransitionCalculationDelay()` method, which allows to retrieve the delay for the first schedule of the `AppTransitionWorker` job.

4.2 The System for Feedback Collection

When dealing with the topic of implementing the chosen strategy for collecting *user feedback*, great care was taken in order to integrate it as much as possible with the existing classes and procedures that were already used inside RecApps, in order reduce and optimize the amount of changes that needed to be done.

First, some *attributes* needed to be added to the `PhoneSession` class to represent the newly acquired information, along with some *flags* to give further details about the status of the *session*, and to the `ContextType` enumeration, in order to include other kinds of *contextual information* (i.e., the type of *feedback element*) inside the content of the session.

The next challenge that was faced addressed the problem of designing an effective *interface* for showing to the user the *feedback form*, in order to give an appropriate amount of information regarding the past session, and present the possible options in a clear and interesting way.

After the interface was completed, the application logic to determine if the form needed to be shown to the user, and to propagate feedback information between similar phone sessions was added to the operations performed when the phone is unlocked, and a new phone session is about to start.

4.2.1 Changes to the Entity Classes

The `PhoneSession` class is the one that received the most changes and integrations during the development of the new *recommendation system*, since it had to contain various information about the *user feedback*, along with other utility flags and attributes.

The values of the *feedback elements* that are assigned by the user after the feedback form is shown are stored inside three `String` variables. Moreover, the `day` attribute is used to compute the information about the day in which the *phone session* was stored (i.e., from 0 to 21, the duration of the user study).

Finally, three boolean flags were introduced:

- `completed` to determine a session that already received feedback (or for which the decision to not ask the feedback was taken);
- `userFeedback` to recover the sessions for which the user gave a feedback (used in the user study analysis);
- `changed` to reflect potential changes to the local instance also on the remote one stored in Firestore.

The `ContextType` enumeration received some updates as well, in the form of the new values for distinguishing the different *feedback elements* in the array of `NamedItem` elements inside each *phone session.* Specifically, the values `FEEDBACK_EMOTION`, `FEEDBACK_USAGE`, and `FEEDBACK_OVERALL` were added.

The `AppTransition` class was changed to reflect the new additions to the content of the *phone session.* More in detail, the same attributes used in the `PhoneSession` class to represent *feedback elements* that are considered frequent inside the rule are added, along with the `relevanceScore` attribute, to represent the ranking score that is used when determining which active transitions are kept to show inside the widget.

Moreover, also the *constructor* of the class was updated, in order to store the new information regarding *feedback* inside the attributes and the *consequent*, and the `compareTo()` method was added in order to implement the `Comparable` interface and provide a mechanism to sort the transitions based on the *relevance score*.

Finally, an implementation decision was taken in order to consider as valid transitions also those that do not have an app inside their antecedent (e.g., $\{Home, 10-12, Entertainment\} \rightarrow \{YouTube, Positive\}$), since they would represent a starting recommendation for when the user opens the home screen of the smartphone.

To support this change, the `getTransitionsWithoutAppContext()` method was added to the `TransitionDao` interface, in order to provide as output a `List` containing only the transitions that do not start with another app.

4.2.2 The Form for Inserting Feedback

When developing an implementation for the feedback form that was previously designed, the focus was on creating an interface that could appear over the usual screen content, so that the user could receive the form regardless of the application that was currently shown on the screen.

This consideration led to use a "Dialog Activity" approach, i.e., to show the entire content of an Activity inside the space of a Dialog, which is usually smaller than the entire screen size. This allowed to not cover the entire screen space while showing the form, but to achieve a good amount of available space to show the interface.

The `FeedbackActivity` class describes the user interface in which the *feedback form* is shown. It keeps track of the choices of the user regarding *feedback elements*, the referenced `PhoneSession` object, the available *custom emotions* (i.e., emotions that were inserted by the user in previous answers to the feedback form) and the *used apps*.

Its `onCreate()` method is used to complete the layout of the interface (i.e., giving it the right background and rounded borders), load the reference to the last *phone session* and retrieve the previously inserted *custom emotions*.

The retrieval of the custom emotions has to be performed inside an `AsyncTask` execution, because the database cannot be accessed from the main thread of the application, due to potential long wait times.

The values of the feedback elements related to the last phone session, the content of the session itself, and the other maintained values can be retrieved and updated through some *getter* and *setter methods*, e.g., `setUsage()` and `getCustomEmotions()`.

Inside the activity layout, a single `NavHostFragment` element is placed. This serves as a *navigation controller*, to transition between different fragments that show the different parts of the *feedback form*.

The navigation flow is implemented thanks to the Jetpack Navigation Library [26], which allows to define *navigation* in a declarative way, as an XML resource file describing a graph of *screens*. A transition between two fragments is expressed through a navigation arc, which is labeled in order to be referenced in code.

Five `Fragment` instances are used to implement the layout of the feedback form:

- `FragmentIntro` shows a recap of the apps used in the last *phone session*, and the possibility to start or skip the compilation of the feedback form;
- `FragmentEmotion` allows to choose an *emotion* value for the feedback, and also to insert a *custom emotion* (both new or chosen among the ones that were previously inserted);
- `FragmentUsage` allows to choose an *usage* value for the feedback;
- `FragmentOverall` allows to choose an *overall* judgement for the feedback regarding the last phone session, ending the feedback collection procedure;
- `FragmentFinal` is the last shown fragment, and shows a brief animation while the *transitions* are recomputed based on the last given feedback.

Forward navigation is made possible by either selecting one of the available options within one of the screens, or with the *"SKIP"* button, which allows to proceed without applying a value for one of the feedback elements in the phone session. Back navigation, instead, is always performed with the *"BACK"* button, which returns to the previous screen, while also resetting the selection, if there was one.

The `FragmentIntro` class implements the graphic interface for the first fragment of the navigation flow. It is used to provide an introduction to the *feedback* collection to the user, and to allow the user to observe all the apps that were used inside the considered *phone session*.

This, together with the usage time of each app, will give the user the possibility to reflect about the last session, and to give a more precise feedback regarding its content.

To recover the information about the used apps and their duration, the implementation of the `generateUsedApps()` method, was considered. It generates a `HashMap` object that associates each app with the a string that represents its duration. This object is updated by each pair of corresponding `AppEvent` objects contained in the input session (i.e., one for when the app is opened and the other for when it is closed).

After each app usage time is computed, the map entries are formatted in order to show the duration with both minutes and seconds.

After the duration of each used app is computed, the resulting map is passed as input to an `AppUsageAdapter` instance, a subclass of a `RecyclerView`, in order to show them in a scrollable list. Each element to show is bound to a `ViewHolder`, a graphical element that contains the details about a single data item. Figure 4.2a shows an example of the presented information about a phone session in the introductory section of the feedback form.

The next fragment that is shown to the user is the `FragmentEmotion` (Figure 4.2b), in which the options for the `FEEDBACK_EMOTION` feedback element are presented.

In this screen, the user has the possibility to insert an emotion that is not present among the predefined ones. This can be done with the *"Other"* icon, which in turn opens a `CustomEmotionDialogFragment` instance, as shown in Figure 4.1.

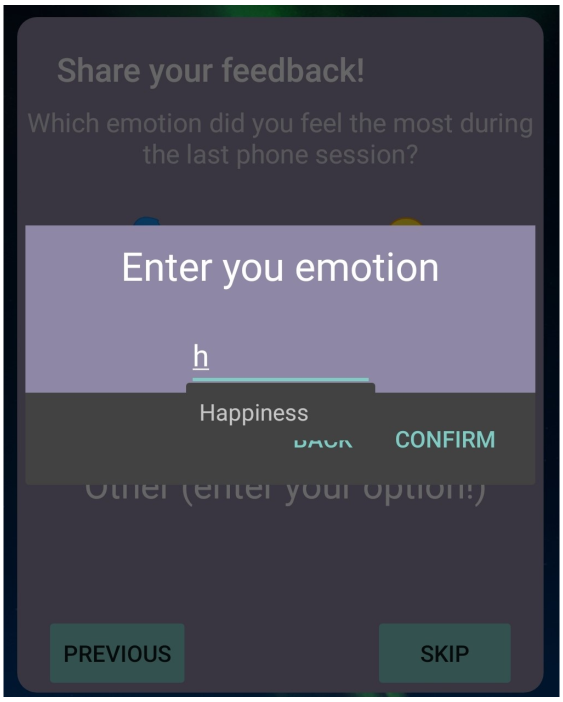
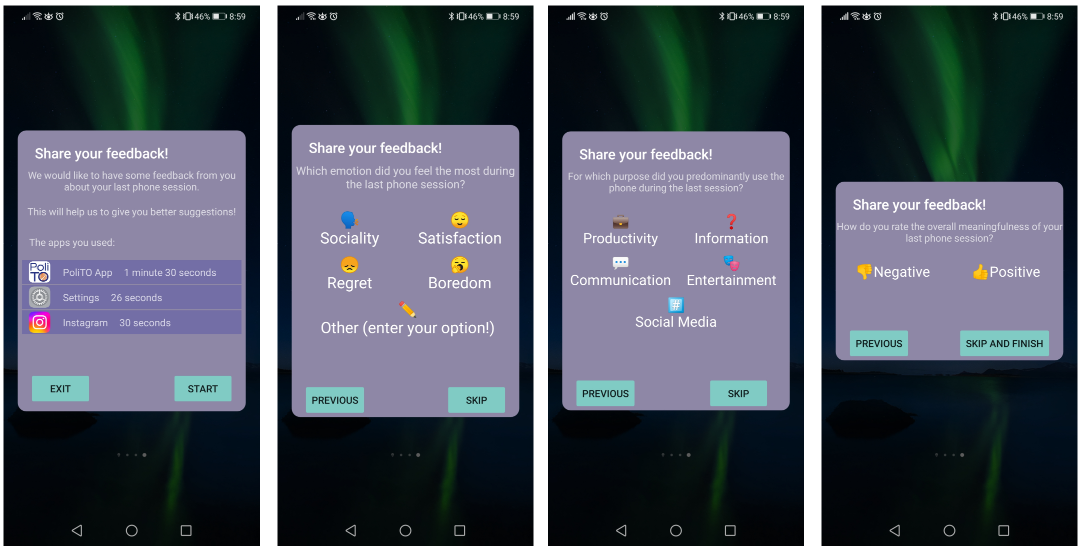

Share your feedback!
Which emotion did you feel the most during the last phone session?
Enter you emotion
h
Happiness
BACK
CONFIRM
Other (enter your option!)
PREVIOUS
SKIP

Figure 4.1: The content of the CustomEmotionDialogFragment, showing a proposition for a custom emotion.

The interface shows an `AutoCompleteTextView` field, in which the user can type in a value for the custom emotion. The object receives an `ArrayAdapter` of strings, containing previous values of custom emotions that the user already inserted, so that they can be proposed after the first character of the new emotion is inserted (in order to avoid misspelling or case errors, and have always the same format for the same emotion).

The next two fragments are used to retrieve information about the main phone usage during the session (`FragmentUsage`, Figure 4.2c) and the overall opinion regarding the entire phone session (`FragmentOverall`, Figure 4.2d). After the last screen is reached, and either an option is chosen or the selection is skipped, the feedback collection for the considered phone session is completed, and the retrieved information is prepared in order to be saved inside the `PhoneSession` instance. Figure 4.2 shows the appearance of all the fragments that compose the feedback form interface.


Share your feedback!

We would like to have some feedback from you about your last phone session.

This will help us to give you better suggestions!

The apps you used:

PoliTO App	1 minute 30 seconds
Settings	26 seconds
Instagram	30 seconds

Share your feedback!

Which emotion did you feel the most during the last phone session?

Share your feedback!

For which purpose did you predominantly use the phone during the last session?

Share your feedback!

How do you rate the overall meaningfulness of your last phone session?

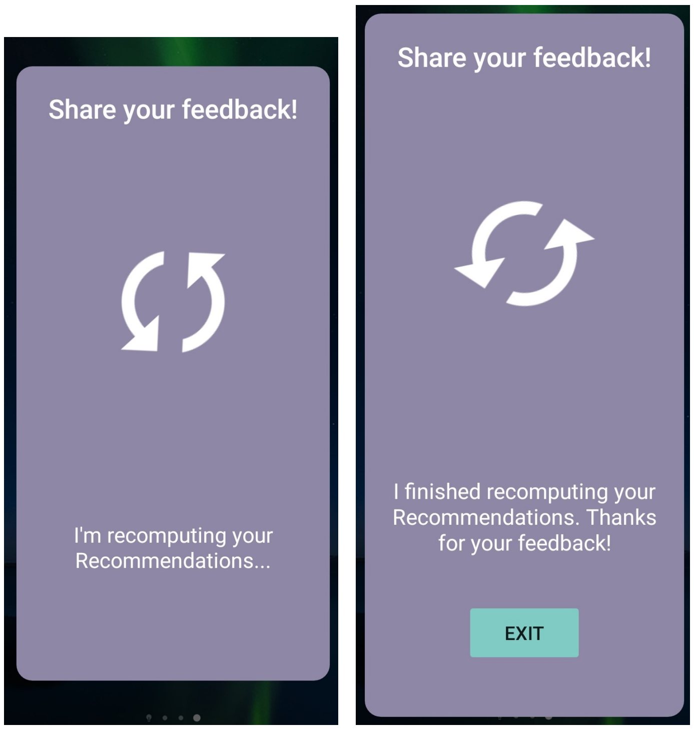
(a) FragmentIntro **(b)** FragmentEmotion **(c)** FragmentUsage **(d)** FragmentOverall

Figure 4.2: The complete layout of the four main fragments of the feedback form.

After the feedback collection procedure reaches the final step, the computation proceeds with the invocation of the `finalizeResults()` method of the `FeedbackActivity` class. Its operations have the effect of inserting the feedback information inside the `PhoneSession` object that was received by the Activity (i.e., both the attributes for the feedback values and the `NamedItems` to represent the new information as input for the *Apriori algorithm*).

Next, if a *custom emotion* was inserted, its value is inserted into the local database, in order to be suggested in future instances of the form. After this operation, the *phone session* is updated inside the database, and the inserted *feedback* is propagated to similar phone sessions that did not already have feedback values.

After these operations are executed, a new screen is presented in the graphical interface, the `FragmentFinal` class. It has the purpose of giving the user some visible feedback about a recalculation of the *transitions* and the *feedback rules*. An animation and a message alerting the user that the recalculation is taking place are shown, and after the operation is competed, another message reporting the success of the procedure is presented. Figure 4.3 shows the two states of the screen.



Share your feedback!

I'm recomputing your Recommendations...

Share your feedback!

I finished recomputing your Recommendations. Thanks for your feedback!

(a) Recalculation in progress **(b)** Recalculation completed

Figure 4.3: A visual feedback informing the user that the recommendations are being recalculated.

If the user does not start the feedback form compilation, the system reacts by invoking the `exitWithoutFeedback()` method in the `FeedbackActivity`. Its purpose is to assign a provisional `null` value to all the feedback elements of the session, and then search for a candidate session from which to propagate the given feedback. This last operation is performed thanks to the `propagateFeedbackToSession()` method.

Another important feature was implemented in order to give users a clear visual indication of the effect of their negative feedback. This means that, whenever an overall negative feedback is given to a phone session, the apps that were used inside that session are removed from the *recommendations* shown in the widget, if present. The `removeNegativeAppsFromWidget()` method is invoked after the phone session is updated, and only if the user inserted the *Negative* value for the `FEEDBACK_OVERALL` element.

This behavior, together with the visual feedback for the recalculation of the transitions, has the purpose to make the user aware that the collected information have a concrete impact on the computed rules and the suggested transitions.

4.2.3 Feedback Propagation to Other Sessions

As discussed in the Design chapter, asking the user for feedback only a limited amount of times (i.e., based on a *probability*) would leave a small number of phone sessions with feedback information, thus leading to rules involving this additional information that are not frequent, and for this reason excluded.

These considerations supported the need to include a mechanism to *propagate* the feedback that users give to a single phone session, in order to make it more frequent among the collected data and include it into the resulting transitions.

Two similar mechanisms for feedback propagation were implemented, one for “backward” propagation (i.e., from the considered session to past sessions) and the other for “forward” propagation (i.e., from a past session to the considered session, if left without any feedback).

After the user has inserted values for the feedback elements related to the last phone session, those values are propagated to similar past sessions, with the invocation of the `propagatePhoneSessionContext()` method.

The method retrieves from the internal database all the *phone sessions* without any value for the feedback elements, and for each of them determines if the *context* is similar with the one of the last phone session. If such comparison is successful, the same values of the feedback elements of the last session are assigned to the similar past sessions, which are then updated in the database.

The `propagatePhoneSessionContext()` takes advantage of two utility methods, `compareContext()` and `compareApps()`, which are used together to determine if the last phone session and one of the previous ones share the same contextual information and the same used apps.

The `compareContext()` method performs a comparison between the two context elements passed as arguments in String format. Such comparison is considered to be satisfied if and only if the two values are the same, or if they both are `null`.

The `compareApps()` method, instead, needs to determine if the used apps in the two considered sessions are the same. The first operation is to retrieve an `ArrayList` of strings containing the names of the used apps for both sessions. Then, the sizes of the two arrays are compared, in order to immediately return `false` if the number of used apps is different between the two sessions.

If the number of used apps is the same, each app used in the first session is then searched among those of the other session, returning `false` in case one of these searches does not find a match. If, after the loop ends, the method has not terminated yet, it means that the two sessions have the same number of used apps, and those apps are the same.

The `propagateFeedbackToSession()` method is intended for the “forward” propagation, and works similarly to the previous method. The key difference is that the list of candidate sessions is directly extracted from the ones that match the exact same contextual information as the original session. This selection is performed thanks to the `getPhoneSessionsMatching()` method of the `PhoneSessionDao` interface.

After the sessions with similar context are found, each of them has its used apps compared with the original session, and the feedback values of the first matching session are propagated. This is done in order to propagate the feedback of the most recent similar session.

4.2.4 The Logic for Showing the Form

After the interface for the feedback form and the strategies to propagate the collected feedback were defined, the next implementation topic addressed the *logic* to consider to decide whether to show the form or not.

Accordingly to the design choices, the adopted methodology involves a *probability* to use in order to limit the *frequency* with which the form is shown.

The implementation influenced the `onReceive()` method of the `ScreenListener` class, more specifically the part in which the `ACTION_SCREEN_ON` event is received, since it corresponds to the start of a new phone session, and thus the feedback form can be shown for the last terminated session. Its implementation is presented in Listing 4.1

After the last *phone session* inserted in the database is retrieved, and the *probability* to show the form is computed based on the *settings* managed by the user, the checks to determine if the form can be presented are performed. First, the considered session must not already have some *feedback* elements (i.e., to avoid considering multiple times the same session, if the last one was not inserted), then, the conditions for showing the form are evaluated, along with the extraction of a random number to test against the feedback probability.

If all the checks are passed, a new `Intent` for the `FeedbackActivity` is created, and the last phone session is passed as an extra argument. If, instead, the feedback form is not shown for the selected phone session, and that sessions did not already received any feedback, a search for candidate sessions from which to propagate the feedback is performed, through the `propagateFeedbackToSession()` method.

```java
public void onReceive(final Context context, Intent intent) {
    if (intent.getAction().equals(Intent.ACTION_SCREEN_ON)) {
        /* ... */
        PhoneSession last = AppDatabase.getLastPhoneSession(context);
        if (last != null) {
            double feedbackProb = 5 * PreferenceManager.getDefaultSharedPreferences(context).getInt("feedback_freq", Constants.DEFAULT_FEEDBACK_FREQUENCY) / (double) 100;
            if (PreferenceManager.getDefaultSharedPreferences(context).getBoolean("feedback_always", false))
                feedbackProb = Constants.TEST_FEEDBACK_PROBABILITY_ALWAYS;
            if (!last.isCompleted()) {
                if (FeedbackAlgorithm.canIShowFeedbackForm(context)) {
                    if (Math.random() <= feedbackProb) {
                        Intent feedbackIntent = new Intent(context.getApplicationContext(), FeedbackActivity.class);
                        feedbackIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_NEW_TASK);
                        feedbackIntent.putExtra("session", last);
                        context.startActivity(feedbackIntent);
                    } else {
                        last.setCompleted(true);
                        FeedbackAlgorithm.propagateFeedbackToSession(last, context);
                    }
                } else {
                    last.setCompleted(true);
                    FeedbackAlgorithm.propagateFeedbackToSession(last, context);
                }
            }
        }
        /* ... */
}
```

Listing 4.1: the additional content of the onReceive() method of the ScreenListener class.

4.3 Integrating Feedback into the Algorithm

To properly exploit the collected feedback information through the dedicated interface shown to the user at the start of a new phone session, some changes needed to be applied to the rule extraction procedure, specifically in the `StatsManager` class.

The additional implementations addressed how the rules are filtered, based on the new available information that can be contained inside them, the newly introduced "feedback rules" to compute correlations between feedback elements, and the algorithm to compute the *relevance* of the available *transitions*, and determine the most meaningful content to fill the widget with.

4.3.1 Adding feedback to association rules

The `buildTransitions()` method is in charge of producing the *association rules*, by means of the *Apriori algorithm*, from the collected `PhoneSession` objects. A fundamental change was implemented in order to exclude from the entire procedure those sessions that are not *completed* yet, i.e., for which a decision on whether or not to ask for *feedback* has not been taken.

This result can be obtained by checking the `isCompleted()` method of the `PhoneSession` class, which is a *getter* method for the `completed` property.

While the `getRules()` method, used to generate the *association rules* from the `NamedItem` object inside each session, remained unchanged, additional checks needed to be implemented in the `filterRules()` method, since the presence of new kinds of information inside the rules could lead to badly formatted rules that made no sense.

To detect and exclude wrong rules, other boolean flags were added inside the method, and tested at the end of its execution to check if the rule could be added to the rule set. Specifically, the new conditions included:

- rules with an element of type `FEEDBACK_OVERALL` inside the body (indicated by the flag `overallInBody`) are excluded, since that kind of information is the consequence of all the other contextual elements;

- rules that have no apps inside the head (flag `noAppsInHead`) are excluded, a plausible situation now that new elements other than those of type `APP` can appear inside a rule's head;

- rules with an element of type `FEEDBACK_EMOTION` or `FEEDBACK_USAGE` inside the head (flag `contextHead`) are excluded, since those kinds of information serve as context for the elements in the rule's consequent;

- rules with at least an element of type `FEEDBACK_EMOTION` or `FEEDBACK_USAGE` inside the body (flag `contextInBody`) are allowed, since now the rules with no apps in the context are considered to be valid.

After all the tests are executed and the value of each flag is assigned, the final condition checks if all the flags are `false`, with the exception of `contextInBody`, since it represents a situation that should be accepted. If the final check is satisfied, the rule is added to the final set, that is finally returned.

Another additional implementation introduced the `FeedbackRule` class, a specialized *association rule* that describes correlations between feedback elements in phone sessions. Specifically, the *antecedent* can contain a single element of type `FEEDBACK_EMOTION` or `FEEDBACK_USAGE`, while the *consequent* is represented through an element of type `FEEDBACK_OVERALL`.

The procedure for generating these new rules is almost the same as for the traditional association rules, but the *confidence* threshold was lowered to a smaller value, in order to compensate the smaller amount of available feedback information.

Inside the `buildFeedbackRules()` method, only the *phone sessions* with at least a feedback element are kept and passed to the *Apriori algorithm*. Then the resulting set of sessions is used to extract the feedback rules that will generate the `AppTransition` objects to store in the internal database.

Similarly to the traditional rules, the new *feedback rules* need to be filtered, once extracted, in order to exclude potential results that have no meaning, such as rules with a `FEEDBACK_OVERALL` element in the antecedent or with the other two kinds of elements in the consequent, since the purpose of feedback rules is to evaluate the relationship between a specific feedback element and the "general" evaluation given by the user.

Both the extracted transitions and the feedback rules are saved in the local database with the dedicated methods, which are the `refreshTransitions()` and `refreshFeedbackRules()` methods.

4.3.2 Using feedback to promote meaningful Suggestions

The second fundamental aspect to consider in order to implement an effective integration mechanism for the collected feedback information inside the recommendation procedure is to define how the *feedback* influences the process of showing certain app switching *suggestions* inside the floating widget.

As decided in the design phase, the chosen methodology to implement this behavior was to rank all the possible *transitions* according to their "relevance", i.e., how much they reflect behaviors and habits that are positive and meaningful for the user.

This strategy exploits some of the methods implemented inside the newly added `FeedbackAlgorithm` static class, which owns also the other methods that are used in the previous phases of the feedback collection procedure.

The root of this entire logic is in the `onAccessibilityEvent()` method of the `AppMonitor` service, in which the opening of a new app is detected, and the potential *transitions* that should be activated given that app and the current *context* are passed to the *widget* in order to show their icons.

First of all, the procedure should be made able to continue even if the opened app is the launcher application of the smartphone. This is done by testing the result of the `amItheLauncher()` method of the `AppUsageStatistics` static class.

Next, the set of available `FeedbackRule` objects is retrieved, along with the appropriate set of candidate `AppTransition`, based on whether the detected app was the launcher or not, which are also filtered to leave only the active ones (i.e., for which the contextual information matches the current ones).

After this loading phase, the "feedback algorithm" is actually executed, with the invocation of the `computeBestTransitions()` method.

The `computeRelevanceScore()` method, reported in Listing 4.2, is the one in charge of determining the *score* to assign to each transition, which will determine its position in the final ranking to take only some of them and show them in the widget. Its code considers each *feedback* element of the transition, one at a time, and determines its contribution to the score. The `overall` value directly brings a $+1$ or -1 to the score, while the `emotion` and `usage` values are searched inside the body of the feedback rules, and if a match is found, then the value of the consequent (retrieved through the `isPositive()` method) determines the contribution.

Since different transitions with different values for their relevance score can instead suggest the same app, thus potentially excluding some potential recommendations from the final list, it is important to remove any duplicate transition from the list prior to making this selection.

This operation is performed inside the `removeDuplicates()` method, which loops over all the transitions, and for each of them searches for other transitions with the same app in the consequent. Each found transition is removed from the candidate list, since it has a lower relevance score than the original one.

4.4 Further changes to the app

In addition to the main design and implementation for improving the *meaningfulness* of the recommendations in the new system, other utility features were considered and implemented inside RecApps.

4.4.1 A screen for recent sessions

This screen was designed by taking into consideration the possibility that the user would not immediately answer to the feedback form, thus depriving the system of

```java
private static Integer computeRelevanceScore(AppTransition
    transition, ArrayList<FeedbackRule> feedbackRules, Context
    context){
     Integer score = 0;
     if(transition.getEmotion() != null){
          String emotion = transition.getEmotion();
          for(FeedbackRule rule : feedbackRules) {
               if (emotion.equals(rule.getBody().getName())) {
                    if (rule.isPositive(context))
                         score++;
                    else
                         score--;
               }
          }
     }
     if(transition.getUsage() != null){
          String usage = transition.getUsage();
          for(FeedbackRule rule : feedbackRules) {
               if (usage.equals(rule.getBody().getName())) {
                    if (rule.isPositive(context))
                         score++;
                    else
                         score--;
               }
          }
     }
     for(String el : transition.getConsequent().keySet()){
          if(el.equals(context.getString(R.string.
    feedback_overall_positive))
               score++;
          else if (el.equals(context.getString(R.string.
    feedback_overall_negative))
               score--;
     }
     return score;
}
```

Listing 4.2: the computeRelevanceScore() method.

important additional information that can improve the future transitions.

This is also reinforced by the fact that the feedback form does not have a 100% probability of being proposed to users, and so loosing information not only because of the limited frequency with which it is collected, but also due to the unwillingness of the user to provide the feedback would have proven too harmful for the procedure.

For this reason, a new screen providing information regarding the most recent *phone sessions* was implemented and added to RecApps, in order to give users a tool for reviewing past sessions and eventually give feedback to some of them.

The `SessionsFragment` class is an instance of `Fragment` that shows a list of the 10 most recent retrieved `PhoneSession` objects. The interface exploits a `SessionsListAdapter`, an instance of the `RecyclerView` class, in order to show the details of each session, e.g., the used apps, the contextual information and the eventual feedback that was assigned to the session. Figure 4.4a shows the appearance and content of the `SessionsFragment` layout.

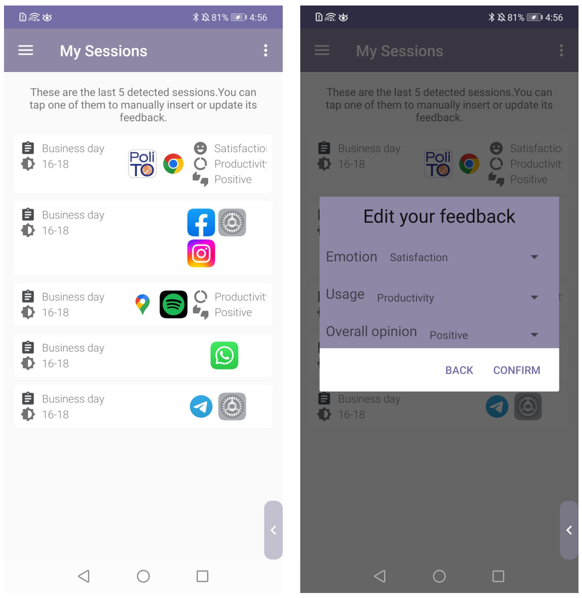
The behavior of allowing the user to edit or add the feedback for a past phone session is obtained by showing an instance of the `UpdateFeedbackDialogFragment`. Its content shows three inputs that initially contain the original value for each feedback element (or the *"No value"* placeholder), which can be changed in order to update the feedback.

The new updated values for the feedback elements of a session will be propagated "backwards" in all the previous phone sessions that did not already receive the feedback, and only if those sessions match the context and used apps of the modified one.

Figure 4.4b shows the appearance of the `UpdateFeedbackDialogFragment` that appears when the user selects one of the past phone sessions to change or assign the values regarding its feedback.

The logic for reacting to the confirmation of the new values for the feedback elements of a phone session is contained inside the `onDialogPositiveClick()` method, implemented by the `SessionsListAdapter` that displays the details of each phone session. The first operations to perform are the substitutions of the old feedback values with the new ones. Together with this, also the `NamedItem` related to the old feedback value needs to be removed, and eventually replaced with a new instance derived from the new value.

After the local update, the phone session needs to be updated also in the database, and the new inserted values are propagated to other sessions. Next, the most recent phone sessions are retrieved again, and a procedure aimed at updating the visual content of the list is executed. This algorithm is based on the `DiffUtils` callback, which computes the differences between two lists of objects, and propagates the appropriate changes to the visualized elements in order to represent the second set of values.



These are the last 5 detected sessions.You can tap one of them to manually insert or update its feedback.

Edit your feedback

Emotion Satisfaction

Usage Productivity

Overall opinion Positive

(a) SessionsFragment **(b)** UpdateFeedback DialogFragment

Figure 4.4: The interface of the SessionsFragment (a) and the DialogFragment used to update the feedback for a phone session (b).

4.4.2 The settings screen

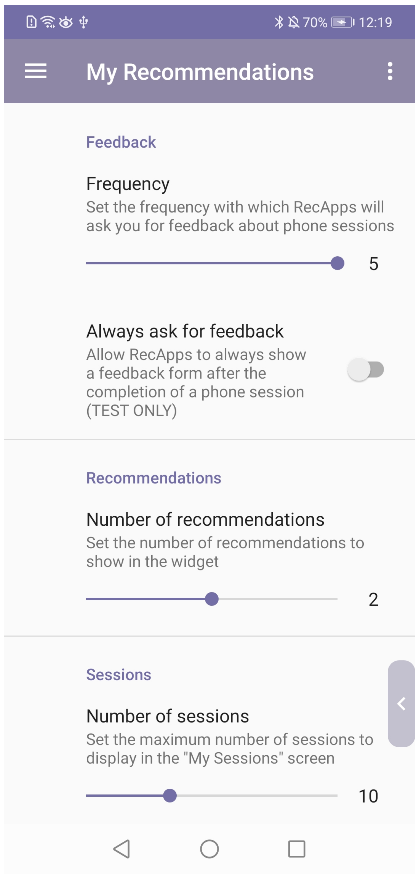
The last major addition that was put inside the original RecApps application in order to implement the new and improved recommendation system was a *settings* page, through the `SettingsFragment` class (Figure 4.5).

The need for such a tool comes from the fact that some of the other implementations require some form of control by the user, in order to customize their behavior and adapt it to the user's needs.

For example, the user may want to reduce the *probability* with which the feedback form, since they find it to intrusive for the normal phone usage. Another case that supports the introduction of a settings page can be found in the original RecApps study, which reported that the average number of applications that could be found inside the widget was 3, and when the number of icons was higher, this caused issues with usability of the entire smartphone screen [4].

A plausible solution to the problem is to give users the possibility to limit the number of *recommendations* shown inside the widget, in order to avoid accidental use and to fully exploit the functionalities of the feedback algorithm, which is able to select the best N transitions to fill the widget.

The RecApps preference screen was implemented with the `root_preferences`



12:19

My Recommendations

Feedback

Frequency
Set the frequency with which RecApps will ask you for feedback about phone sessions
5

Always ask for feedback
Allow RecApps to always show a feedback form after the completion of a phone session (TEST ONLY)

Recommendations

Number of recommendations
Set the number of recommendations to show in the widget
2

Sessions

Number of sessions
Set the maximum number of sessions to display in the "My Sessions" screen
10

Figure 4.5: The newly added settings screen of RecApps.

XML file that describes the various controls and their properties, which are then displayed inside the `SettingsFragment` class.

The chosen options that the user is able to control include:

- the probability of showing the feedback form, in a scale from 1 to 5 (each unit corresponds to 5%, so the range of probabilities is 5% – 25%;
- the number of recommendations to show inside the widget, form a minimum of 1 to a maximum of 3, based on the conclusions emerged in the previous RecApps study;
- the number of sessions to show in the `SessionsFragment`

To retrieve and use the value of one of the preferences, the `PreferenceManager` class is used, in order to get the value of one of the properties given its key, that is defined in the XML configuration file.

Chapter 5

User Study

This chapter presents the structure ad goal of the *user study* that was conducted after the developing of the new recommendation system was completed.

The goal of the study was to evaluate how the newly designed *feedback algorithm* and its method and classes would influence the *transitions* generation procedure, and help in promoting more meaningful *suggestions*, potentially improving the quality of the time spent using the smartphone.

Since the new system for recommendations was developed by taking the original RecApps mobile application as a starting point, the study to assess its effectiveness was structured in the same way as the one that was presented in [4].

Indeed, most of the initial preparation (e.g., number of participants, study structure and data collection) followed the same strategies of the original study.

However, the main focus was not on whether users would interact with the system, but if the introduction of the *feedback* collection procedure helped in producing *suggestions* that reflected the participants' idea of meaningful usage, thus contributing to their *digital wellbeing.*

5.1 Participants

The *participants* for the study were recruited exclusively by contacting personal acquaintances, for a total of 16 people (8 male and 8 female).

Participants were on average 28.06 years old ($SD = 11.52$), and their profession was almost equally distributed between university students (9) and workers (7).

5.2 Method

The study was run between July and August 2022, for a total duration of three weeks. The RecApps apk was distributed to participants through communication apps, along with instructions on how to proceed with the installation and give all the needed permissions, and then installed on their smartphone.

The study structure was the same as the original one, with the first week (*baseline* phase) dedicated only to the background collection of phone sessions and feedback, and the last two weeks (*RecApps* phase) in which *suggestions* were shown inside the widget, and periodically recalculated based on more recent data and *feedback* information.

5.2.1 Data Collection and Metrics

During the entire duration of the study, usage data of the participants was collected in an anonymous form and stored inside Cloud Firestore [27], a document-based, NoSQL database owned and supported by Google. Each day, periodically, the app uploaded all the local usage data on the Firestore instance associated with the RecApps project, inside the collections corresponding to the correct user.

The collected data was organized for each participant, and included information about each *phone session* (e.g, timestamps, used apps, contextual information, added feedback), and the *transitions* that were proposed as icons in the *widget* (e.g., the *contextual information* of the transition, the *consequent apps*, the associated *feedback* information, a flag to determine if the transition was used).

After the three weeks of data collection, the available information was analyzed to extract values for some objective *metrics* regarding the phone sessions and used transitions, which included:

- *Duration*, i.e., of the average duration of smartphone sessions performed by users;
- *Unique Apps*, i.e., the number of distinct apps that were opened and used during a single phone session;
- *App Openings*, i.e., the average number of times an app was opened by a user during a phone session;
- *Used Widgets (%)*, i.e., the percentage of occurrences of an app that was opened by using the corresponding icon when shown inside the widget.

In addition to the presented metrics, further analysis was conducted in order to analyze the effectiveness of the feedback system, specifically:

- *Suggestions impact on feedback*, i.e., comparing the percentage of negative or positive feedback between week 1 and weeks 2-3;
- *Suggestions impact on switching behaviors*, i.e., determining if using the widget during a positive session prevented from replicating a similar negative session;

- *Feedback impact on future suggestions*, i.e., determining if, after a positive (negative) feedback, the suggested apps that were involved in the corresponding session were proposed more (less) frequently.

5.2.2 Qualitative Feedback from Users

After the three weeks of the study were concluded, a final *survey* was proposed to the participants, in order to gather *qualitative feedback* about their experience with the app and the proposed *recommendations*. The first questions were inspired by the ones that were included in the original study, while others were added in order to obtain information about the feedback collection.

The complete list of open-ended questions is the following:

- How was your overall experience with RecApps? Did you like it? Did you find any advantages or disadvantages?
- Did you use the recommendations that were proposed in order to open apps or to switch between two of them?
- Has RecApps changed the way in which you use your smartphone?
- What do you think about the feedback collection of RecApps? Did you find necessary to lower its frequency? Do you think that the proposed questions and options best described your opinion regarding the phone session?
- It seemed to you that the received recommendations reflected the feedback that you were giving?
- Do you have any suggestions to improve RecApps?

Chapter 6

Results

The following chapter covers the results obtained from the *user study* conducted to evaluate the effectiveness of the new *recommendation system* based on using *feedback* in order to improve the *meaningfulness* of the proposed *transitions*.

First, the objective *metrics* derived directly from the analysis of the collected data of the participants is discussed, and then a summary of the *qualitative feedback* provided by participants in the final interview after the study is reported.

6.1 Results given from Objective Metrics

In this section, the analysis of the objective *metrics* extracted from the collected user data is divided into two parts:

- the evaluation of the same metrics that were used in the first RecApps study [4], in order to maintain continuity and verify how the newly designed system performs compared with the original version;
- the evaluation of the new metrics related to the *feedback* elements and their effects on *recommendations* and *phone sessions*.

6.1.1 Evaluation of the Original Metrics

On average, each participant used the *widget* to open a suggested application (both from the home screen and from another app) 69.69 times (SD = 56.41, *median* = 53.00) and, on a daily basis, the average amount of used widgets was of 6.88 (SD = 7.75, *median* = 5.00). Furthermore, the overall *usage percentage* of the widget, considering all the users, resulted in a value of 3.18%.

As shown in Figure 6.1, the percentage of daily usage of the widget seemed to slowly decrease over time, as the study progressed. Since it does not reach values too far from the average, this trend should not be caused by malfunctions of the algorithm, but rather because of a sense of frustration due to the *widget* covering important portions of the screen, as reported in many answers in the final interview:

> *"The only negative aspect that I noticed is that the widget with the recommendations is always displayed on the screen, it would be nice to be able to hide it entirely in some occasions."* (P3)
>
> *"A quite obvious disadvantage was the combined presence of the RecApps' widget and another overlay interface that is displayed when using mobile games, since they occupied two different portions of the screen. Sometimes, I needed to act in positions of the screen where the widget was placed, and so its presence disturbed the experience a bit."* (P9)

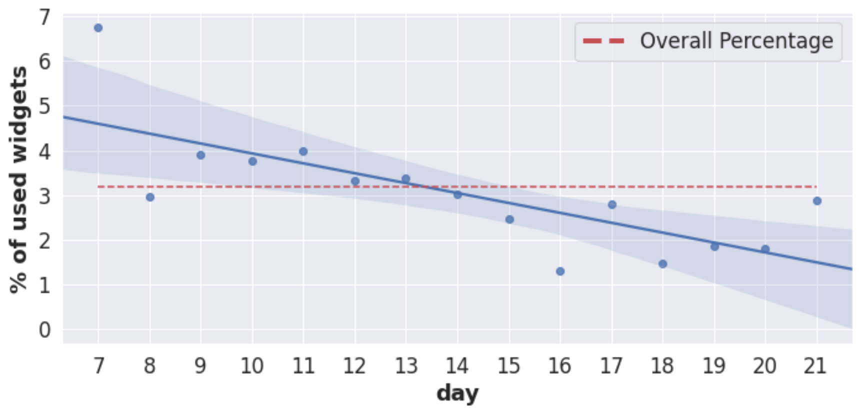


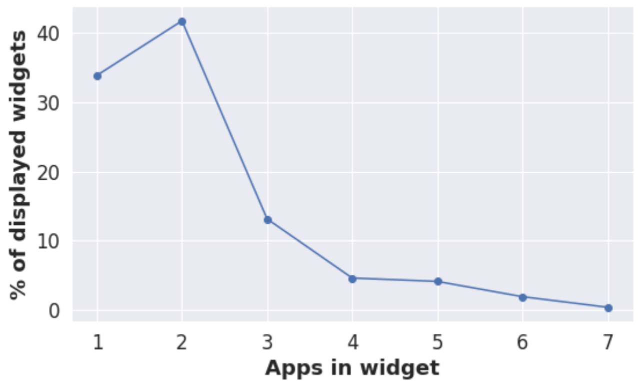
Figure 6.1: The daily percentage of used widgets, showing a slightly decreasing trend for this metric throughout the study.

Focusing on the number of displayed icons inside the widget, the majority of the widgets was shown with 1 or 2 icons inside it ($\sim 30\%$ and $\sim 40\%$ respectively), as reported in Figure 6.2a.

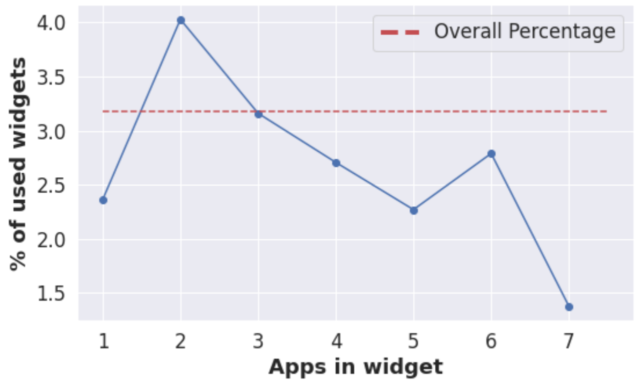
Unfortunately, despite the use of the shared preferences inside the application, in order to let the user set the maximum number of *recommendations* to show inside the widget (or fixing it to a default value of 2), some widgets of size greater than 3 were displayed during the two weeks of recommendations. These occurrences, however, can be seen as outlier cases, since their total percentage is very low.

The *number of icons* displayed inside the widget was found to influence the *probability* of using the widget when it had that amount of apps shown inside. Figure 6.2b shows that the widget was used the most when it contained 2 or 3 apps, with the percentage quickly decreasing when the number of apps was higher. This behavior is once again confirmed by the final interviews of the users, which highlighted how the widget could sometimes appear in uncomfortable positions inside the screen.

> *"One single issue that I experienced is that the buttons with the icons sometimes appears in inconvenient positions."* (P4)
>
> *"Having a widget with many apps always visible on the screen sometimes obstructed the viewing of other content, like when using YouTube or Netflix."* (P8)



(a) Distribution of widget size



(b) Usage of the widget given its size

Figure 6.2: The analyzed information about the widgets' size: the overall distribution of the various sizes (a) and the percentage of use of each widget in relation with its size (b).

Another part of the analysis that was conducted as a parallel with the original study addressed if and how the introduction of the *widget* and the *recommendations* shown therein affected the smartphone usage by the participants.

The median *duration* of a phone session slightly changed from 4.01 minutes during the first week of the study (the *baseline* phase) to 4.83 minutes during the following two weeks (the *RecApps* phase). However, the result of a Wilcoxon Signed-Ranks test [28] (i.e., a nonparametric test that compares two correlated data groups, with the goal of rejecting the null hypothesis that there is no difference between the two groups) did not reveal notable differences between the two sets of values. The summary of the analysis about sessions' duration is reported in Table 6.1.

Table 6.1: The effect of RecApps and its widget on the average duration of the collected phone sessions, along with the p value of the Wilcoxon Signed-Ranks test.

	Baseline		**RecApps**		
	Median	**Range**	**Median**	**Range**	**p**
Duration [min]	4.01	[1.97 - 66.77]	4.83	[2.36 - 44.62]	0.231

Looking at the changes in the average number of *unique apps* in a single session (i.e., the set of distinct applications that a user opened during a phone session) and the average number of *app openings* inside a session (i.e., the number of apps that were shown in the foreground during a smartphone usage session), the former slightly increased from 1.61 (SD = 0.18) to 1.70 (SD = 0.13), and a similar trend was also followed by the latter, moving from 6.31 (SD = 1.58) to 6.54 (SD = 3.73). Again, the resulting p values of the Wicloxon Signed-Ranks tests performed on the two sets of statistics did not point out any relevant correlation.

Table 6.2 brings together the results regarding the statistics of unique app openings and total app openings inside a phone session.

Finally, the influence of RecApps on the *frequency* of the most common app switches was taken into account. The three most common *patterns* that were considered resulted to be:

- WhatsApp to Instagram and vice versa (shown in a total of 2,677 widgets);
- Facebook to WhastApp and vice versa (shown in a total of 2,246 widgets);
- Telegram to WhatsApp and vice versa (shown in a total of 1,339 widgets).

Out of the three analyzed transitions, only the one involving switching between Facebook and WhastApp resulted to be more frequent on average during the

Table 6.2: The effect of RecApps and its widget on the average numbers of unique app openings and overall app openings, along with the p values of the Wilcoxon Signed-Ranks tests.

	Baseline		**RecApps**		
	M	**SD**	**M**	**SD**	**p**
Unique app openings [#]	1.61	0.18	1.70	0.13	0.632
Overall app openings [#]	6.31	1.58	6.54	3.73	0.159

RecApps phase (M = 2.43, SD = 2.50 *vs.* M = 4.17, SD = 6.49). However, no statistical relevance was found in this increase ($p > 0.05$).

To better understand the plausible cause of these trends, also the feedback given to the phone sessions including each pair of apps was considered. The results highlighted that, for example, in the case of the "WhatsApp-Instagram" transition, there was not a clear majority of positive feedback, probably not high enough to cause an evident increase of propositions of that transition.

A more definite difference was instead found for the "Facebook-WhatsApp" transition, thus potentially explaining why the number of transitions from one app to the other increased over time.

As of the last considered transition, "Telegram-WhatsApp", the high difference between the positive and negative feedback was due to the scarce amount of feedback information related to the sessions that included said transition.

The results obtained from the analysis of the most common switching behaviors and the effect of RecApps over those transitions are reported in Table 6.3.

Table 6.3: The effect of RecApps and its widget on the most common app switching behaviors of the participants, , along with the p values of the Wilcoxon Signed-Ranks tests.

	Baseline		**RecApps**		**Feedback**		
	M	**SD**	**M**	**SD**	**POS**	**NEG**	**p**
{WhatsApp, Instagram} [#]	5.59	7.90	4.48	7.64	63.43%	34.57%	0.588
{Facebook, WhatsApp} [#]	2.43	2.50	4.17	6.49	80.00%	20.00%	1.000
{Telegram, WhatsApp} [#]	4.46	4.59	2.31	1.61	98.04%	1.96%	0.203

The last interesting analyzed aspect refers to the "overlooked transitions", i.e., those transitions that were not directly activated by users through the *widget*, but whose *consequent* app was manually opened at the same time that the app was shown in the widget.

This means that, by considering both the amount of used transitions and "overlooked" ones, a global overview of the system's ability to represent and recognize users' behaviors and usage patterns can be obtained.

Overall, 11.01% of transitions were "overlooked" by users and, on average, each of them ignored 241.19 (SD = 325.52) transitions, while still manually opening the suggested app. Combining this result with the overall frequency of use leads to a total of 14.19% of the suggestions that were either used or "overlooked", giving an overall estimate of the *accuracy* of the obtained system.

6.1.2 Evaluation of Metrics Related to Feedback

Overall, participants answered to the form for *feedback* collection 8.51% of the times. Such a limited interest in providing feedback information for the recent *phone sessions* could be caused by the fact that the methodology with which the form was shown to users was (almost) totally statistical (i.e., based on a *probability*). In fact, there was no dependency with the current *context* or with the content of the *phone session* that enriched the decision process, and this may have led users to skip the form more easily, as reported in some of the final interviews:

> *"I sometimes skipped the feedback form entirely, by lack of will or by accident, since I was already tapping on the screen to do other actions and tapped on the 'Exit' button instead."* (P2)
>
> *"The feedback collection was undoubtedly important, but I did not always provided the feedback when asked for, especially if I was at work."* (P4)
>
> *"Sometimes the feedback requests appeared in moments in which I was in a hurry, and could not answer."* (P5)

Analyzing the impact of RecApps on the overall feedback given by users, no evident effects emerged, as shown in Table 6.4. On average, both the number of positive and negative feedbacks given by users slightly decreased (32.06 *vs.* 31.44 and 5.69 *vs.* 4.94). The fact that the negative feedbacks did not decrease may be explained by the will of the users to "guide" more accurately the recommendation process, giving negative feedback to avoid certain apps among the suggested ones inside the widget.

Focusing on whether the presence of the widget managed to avoid the repetition of switching behaviors that were considered negative by users, no results were obtained, due to data scarcity. Unfortunately, the collected amount of data about

Table 6.4: The effect of RecApps and its widget on the overall feedback given by users to the collected phone sessions, along with the p value of the Wilcoxon Signed-Ranks test.

	Baseline		**RecApps**		
	M	**SD**	**M**	**SD**	**p**
Positive sessions [#]	32.06	46.51	31.44	32.00	0.821
Negative sessions [#]	5.69	12.39	4.94	10.08	0.553

sessions and transitions was not enough to provide results, given the restrictive conditions that were needed to extract the results.

In fact, the procedure needed to find all the sessions with positive feedback and in which a widget was used (only a total of 126), and for each of them extract the transitions that were used and the apps that were opened before the widget was activated. Then, for all the negative sessions that were completed before the first one, only the ones whose apps all appeared in the original session contributed to the total count.

This complex procedure led to a total of 0 avoided negative switches, but probably this result was caused due to the small amount of available data compared to the precision needed by the search.

Finally, the effectiveness of the feedback mechanism for replicating positive sessions and avoiding negative sessions was evaluated. The goal of this analysis was to determine if, after a positive feedback on a phone session, the apps involved in that phone session were used more often, potentially due to the recommendations taking into account the positive feedback, and also testing the reverse conditions for negative feedbacks.

The results confirm the expected trend, with apps included in a positive session being used on average 854.5 (SD = 2142.82) times before the feedback and 1272.5 (SD = 2104.74) times after the same feedback, and on the other hand the apps included in a negative session being used on average 1555.0 (SD = 1771.44) times before the feedback and 1370.0 (SD = 1925.77) times after the feedback. Moreover, a Wilcoxon Signed-Ranks test revealed significant differences ($p < 0.05$) in number of apps used before and after a positive feedback, when considering the effects of RecApps.

Table 6.5 shows summarizes the results of this analysis, showing how a positive or negative feedback effectively influenced the number of sessions that contained the same apps.

The analysis of this last metric provided results that support the idea that

Table 6.5: The effect of RecApps and its feedback mechanism on the amount of sessions that included the same apps of the evaluated session, along with the p value of the Wilcoxon Signed-Ranks test.

	Before		**After**		
	Median	**SD**	**Median**	**SD**	**p**
Positive feedback [# of uses]	854.5	2142.82	1272.5	2104.74	0.018
Negative feedback [# of uses]	1555.0	1771.44	1370.0	1925.77	0.704

including feedback in the recommendations of the system helped in supporting the interactions that the participants found most useful.

6.2 Results Given from Qualitative Feedback

Out of the 16 participants to the user study, only two of them reported a neutral or mostly negative experience with the use of RecApps. This was, as stated by them, mostly due to technical problems that were found during the test, or for having the app installed on an old smartphone.

> *"Personally, I did not find RecApps excessively useful, since I tend to use a few apps that I already have close together on the screen. Overall, it was useful to ease switching between different apps, but I also experienced some malfunctions, like when recalculating the recommendations or having large delays when moving the widget on the side of the screen."* (P1)
>
> *"I had a mixed experienced with RecApps. I liked the app itself, but some negative aspects made me feel a mostly negative experience. My device is quite old and has very little available RAM, so having RecApps always active cause several crashes and freezes for the entire phone."* (P8)

Aside from the issues that worsened the perceived experience, the rest of the participants found RecApps a very useful tool to move quickly between certain applications, with some of them even stating that they would not mind having the app always installed on their smartphone (U2, U7 and U9).

Regarding the precision of the *recommendations* and the capability of the system to "learn" the participants' habits and reproduce them based on contextual information, all the participants were satisfied.

> *"I tended to use the recommendations almost each time they were showing*

> *an app that I would have used. The more I was using certain apps, the more easily those apps were included among the suggested ones."* (P3)
>
> *"After a week or so I started to receive suggestions, and after a few days they became even more adherent to my habits."* (P5)
>
> *"I managed to 'get the app used' to my habits, so that it could recommend them to me. In fact, I often managed to use RecApps to switch from one communication app to another, to the mailbox and, sometimes, a social app."* (P9)

Some users even found the experience with RecApps useful to reflect on their *habits* when using the smartphone (P4, P5, P6), or at least confirmed some patterns and behaviors that the user was already aware of (P2).

The general perception about the added feedback mechanisms was that it proved able to improve the recommendations, which in turn were able to match the provided feedback.

6.2.1 Feedback Frequency

Most of the participants found that the default *frequency* with which RecApps asked for feedback was adequate, while some others (P1, P8, P12) decided to lower it immediately.

> *"I lowered the frequency for the feedback, since due to some problems the entire procedure was becoming slower and slower, and if I needed to do something with the smartphone it would have become problematic to wait for so long."* (P1)

Other participants, instead, opted to lower the frequency after some time had already passed, in order to let the algorithm collect more initial feedback to produce the first recommendations:

> *"I lowered the frequency after the algorithm had received a good amount of information."* (P7)
>
> *"After a week from the beginning of the recommendations, I changed the rate of the feedbacks, since I found that the app combinations that were present at that time were useful enough."* (P9)

Another interesting comment addressed the possibility to not ask immediately for feedbacks (i.e., once the next phone session begins), but to send a notification each day to bring the user to the screen with all the recent phone sessions, and let the user manually insert the feedback in that way.

6.2.2 Content of the Feedback Form

The content of the feedback form (i.e., the questions that were asked to collect each kind of feedback information, and the available options) was considered adequate in most of the occasions, although some participants found it difficult to describe their experience in certain occasions through the available answers.

Participant P1, for example, reflected on the fact that some of the options for the same question could have been merged together, while P11 felt like a "neutral" option for the *overall* feedback regarding a phone session was missing.

6.2.3 Feedback Effectiveness on Recommendations

Adding *feedback* information to the collected data proved useful in order to promote more meaningful interactions with the smartphone.

Thanks to the analysis of the feedback and its integration in the recommending phase, it was possible for certain users to avoid some behaviors that they detected and considered negative.

> *"I often performed some 'compulsive' checking on certain applications, and of course those apps were included in the phone sessions. Thanks to giving a negative feedback to those sessions, those apps were not suggested when I opened the other ones."* (P2)

In other occasions, the recommendations were used as a sort of "memo" in order to remember to open certain apps in certain conditions.

> *"I found this app useful as a 'memo' tool, that reminded me to use certain apps that were shown in the widget, in some situations in which I had to do something important, like a phone call."* (P10)

Overall, participants were satisfied with how the inclusion of feedback managed to influence the received recommendations, and keep them from receiving suggestions about apps that they judged in a negative way.

6.2.4 Suggestions to Improve RecApps

Although they were mostly satisfied with their experience with RecApps, many participants were eager to share some tips that they found useful to further improve the recommendation system.

For example, a common thought addressed the possibility to hide the floating widget (or temporarily "suspend" its presence on the screen) when using certain applications. Despite working against the core mechanism of RecApps (i.e., to show the widget with the recommended apps as soon as a new application is opened), users reasoned on the fact that, in certain situations (e.g., when watching a video on YouTube), having the widget still present on the screen even if it has been closed is somewhat unnecessary.

Another interesting improvement that the participants thought of involved the possibility to differentiate the feedback for each app included in a single phone session, in order to better formulate the transitions and apply separate feedbacks to different portions of that session.

Finally, an alternative design for the feedback interface and collection method was proposed, in the form of a less intrusive interface, with less questions and options. Specifically, participant P2 suggested that the collected feedback may only include the overall opinion, which should be asked through a simpler and smaller interface, in order to take less time to compile, and potentially make it easier for users to provide a feedback.

Chapter 7

Conclusions and Discussion

This thesis provided a combined perspective on the topics of *app switching* and *digital wellbeing*, highlighting how the works that address the former subject tend to not take into consideration the latter. More specifically, usual app switching and recommendation systems tend to work without including information coming from the final user, thus potentially disregarding their digital wellbeing and promoting unhealthy behaviors that will instead be consolidated.

The new *recommendation system* based on the RecApps application that was presented and evaluated for this thesis, was designed with a greater focus on how to privilege users' *digital wellbeing*, and how to find effective mechanisms to improve and make the suggested shortcuts more meaningful for them.

After a literature analysis on the topic of *digital wellbeing* and *technology overuse*, which helped establish the most useful criteria that a system for monitoring and improving the smartphone usage, the chosen strategy was to collect user *feedback* regarding single phone usage sessions, and include it in the procedure that produced the *transitions* to show inside the floating *widget*.

The resulting system was evaluated during a three-week in-the-wild user study with 16 participants, and the results shown that the newly implemented features and mechanisms were considered useful by most of the users. Specifically, being able to receive *recommendations* that matched the given *feedback*, and potentially control the content of the suggestions through the appropriate feedback, helped the users both in spending their time with the smartphone in a more meaningful way, and in reflecting about how much time is potentially wasted when using the smartphone, thus making the decision to reduce it.

7.1 Limitations and potential improvements

Despite being appreciated by the participants of the study, and having proved the effectiveness and usefulness of the implemented strategies, the presented recommendation system still had some limitations, that could be resolved in future versions, along with potential and useful improvements that could be performed on the resulting system.

First of all, as many participants reported, the possibility to hide or disable the floating widget for limited amounts of time, or when using specific applications, may prove to be useful. Despite going against the core mechanisms or RecApps (i.e., present a set of possible apps to transition to, after a new application is opened), this may have a positive impact on users' digital wellbeing, since it would improve their experience when using certain applications, instead of making them feel frustrated since the content is obstructed with a floating widget that they are not using.

Next, even if the *feedback* strategy was considered useful and effective in order to restrict the set of possible *transitions* that are proposed to users, and keep only the most meaningful ones, according to them, there is still some space for fine-tuning it. Even if its purpose and effectiveness were clear to users, they also reported that they often avoided providing feedback due to the interface appearing in moments during which they had other important matters to attend with their smartphones. This suggests that a smaller and simpler interface for a quicker feedback could be used more frequently by users, since it would not obstruct more important usages of the phone, and may also encourage them to provide their feedback more frequently.

Another interesting modification should address the possibility to assign different feedbacks to distinct sets of apps inside the same *phone session.* This should improve the precision of the algorithm, since it would have more precise *feedback* at its disposal, making it even easier to detect negative and meaningless patterns, and instead promote those that users consider the most useful.

Bibliography

[1] Matthias Böhmer, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. «Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage». In: MobileHCI '11. Stockholm, Sweden: Association for Computing Machinery, 2011. ISBN: 9781450305419. DOI: 10.1145/2037373.2037383. URL: https://doi.org/10.1145/2037373.2037383 (cit. on pp. 1, 5).

[2] Xiang Ding, Jing Xu, Guanling Chen, and Chenren Xu. «Beyond Smartphone Overuse: Identifying Addictive Mobile Apps». In: CHI EA '16. San Jose, California, USA: Association for Computing Machinery, 2016. ISBN: 9781450340823. DOI: 10.1145/2851581.2892415. URL: https://doi.org/10.1145/2851581.2892415 (cit. on pp. 1, 13, 19).

[3] Kai Lukoff, Cissy Yu, Julie Kientz, and Alexis Hiniker. «What Makes Smartphone Use Meaningful or Meaningless?» In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.1 (2018). DOI: 10.1145/3191754. URL: https://doi.org/10.1145/3191754 (cit. on pp. 1, 10, 15, 18, 21).

[4] Alberto Monge Roffarello and Luigi De Russis. «Understanding and Streamlining App Switching Experiences in Mobile Interaction». In: *International Journal of Human-Computer Studies* 158 (2022), p. 102735. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2021.102735. URL: https://www.sciencedirect.com/science/article/pii/S1071581921001531 (cit. on pp. 1, 2, 4, 7–9, 17, 52, 54, 57).

[5] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. «Predicting Mobile Application Usage Using Contextual Information». In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing.* UbiComp '12. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2012, pp. 1059–1065. ISBN: 9781450312240. DOI: 10.1145/2370216.2370442. URL: https://doi.org/10.1145/2370216.2370442 (cit. on pp. 1, 4–6).

[6] Simon L. Jones, Denzil Ferreira, Simo Hosio, Jorge Goncalves, and Vassilis Kostakos. «Revisitation Analysis of Smartphone App Use». In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous*

Computing. UbiComp '15. Osaka, Japan: Association for Computing Machinery, 2015, pp. 1197–1208. ISBN: 9781450335744. DOI: 10.1145/2750858.2807542. URL: https://doi.org/10.1145/2750858.2807542 (cit. on p. 1).

[7] M. Gui, M. Fasoli, and R. Carradore. «"Digital Well-Being". Developing a New Theoretical Tool For Media Literacy Research». In: *Italian Journal of Sociology of Education* 9 (Jan. 2017), pp. 155–173. DOI: 10.14658/pupj-ijse-2017-1-8 (cit. on p. 2).

[8] Juan Pablo Carrascal and Karen Church. «An In-Situ Study of Mobile App & Mobile Search Interactions». In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: Association for Computing Machinery, 2015, pp. 2739–2748. ISBN: 9781450331456. DOI: 10.1145/2702123.2702486. URL: https://doi.org/10.1145/2702123.2702486 (cit. on pp. 4, 5).

[9] Nagarajan Natarajan, Donghyuk Shin, and Inderjit S. Dhillon. «Which App Will You Use next? Collaborative Filtering with Interactional Context». In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: Association for Computing Machinery, 2013, pp. 201–208. ISBN: 9781450324090. DOI: 10.1145/2507157.2507186. URL: https://doi.org/10.1145/2507157.2507186 (cit. on pp. 4, 5).

[10] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. «Predicting The Next App That You Are Going To Use». In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 285–294. ISBN: 9781450333177. DOI: 10.1145/2684822.2685302. URL: https://doi.org/10.1145/2684822.2685302 (cit. on pp. 4–6).

[11] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. «Fast App Launching for Mobile Devices Using Predictive User Context». In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. MobiSys '12. Low Wood Bay, Lake District, UK: Association for Computing Machinery, 2012, pp. 113–126. ISBN: 9781450313018. DOI: 10.1145/2307636.2307648. URL: https://doi.org/10.1145/2307636.2307648 (cit. on pp. 4–6).

[12] Rakesh Agrawal and Ramakrishnan Srikant. «Fast Algorithms for Mining Association Rules in Large Databases». In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. ISBN: 1558601538 (cit. on p. 8).

[13] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining.* 330 Hudson Street, NY NY 10013: Pearson, 2019 (cit. on p. 8).

[14] Google LLC. *Adapt your app by understanding what users are doing.* 2020. URL: `https://developers.google.com/location-context/activity-recognition/` (visited on 07/17/2020) (cit. on p. 9).

[15] Alberto Monge Roffarello and Luigi De Russis. «The Race Towards Digital Wellbeing: Issues and Opportunities». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* CHI '19. Glasgow, Scotland, UK: Association for Computing Machinery, 2019, pp. 1–14. ISBN: 9781450359702. DOI: `10.1145/3290605.3300616`. URL: `https://doi.org/10.1145/3290605.3300616` (cit. on pp. 10–12, 17, 18).

[16] Jonathan A. Tran, Katie S. Yang, Katie Davis, and Alexis Hiniker. «Modeling the Engagement-Disengagement Cycle of Compulsive Phone Use». In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* CHI '19. Glasgow, Scotland, UK: Association for Computing Machinery, 2019, pp. 1–14. ISBN: 9781450359702. DOI: `10.1145/3290605.3300542`. URL: `https://doi.org/10.1145/3290605.3300542` (cit. on pp. 10, 14, 18, 22).

[17] Google LLC. *Our commitment to DigitalWellbeing.* 2018. URL: `https://wellbeing.google/` (visited on 08/17/2018) (cit. on p. 10).

[18] Alberto Monge Roffarello and Luigi De Russis. «Achieving DigitalWellbeing Through Digital Self-Control Tools: A Systematic Review and Meta-Analysis.» In: *ACM Trans. Comput.-Hum. Interact* (2020) (cit. on p. 11).

[19] Katarzyna Stawarz, Anna L. Cox, and Ann Blandford. «Beyond Self-Tracking and Reminders: Designing Smartphone Apps That Support Habit Formation». In: CHI '15. Seoul, Republic of Korea: Association for Computing Machinery, 2015, pp. 2653–2662. ISBN: 9781450331456. DOI: `10.1145/2702123.2702230`. URL: `https://doi.org/10.1145/2702123.2702230` (cit. on p. 12).

[20] R.X. Schwartz, Alberto Monge Roffarello, Luigi De Russis, and Panagiotis Apostolellis. «Reducing Risk in Digital Self-Control Tools: Design Patterns and Prototype». In: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems.* CHI EA '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380959. DOI: `10.1145/3411763.3451843`. URL: `https://doi.org/10.1145/3411763.3451843` (cit. on p. 18).

[21] Paul Ekman. «Are there basic emotions?» In: *Psychological Review* 99.3 (1992), pp. 550–553. DOI: `10.1037/0033-295x.99.3.550` (cit. on p. 22).

[22] ROBERT PLUTCHIK. «A GENERAL PSYCHOEVOLUTIONARY THEORY OF EMOTION». In: *Theories of Emotion* (1980), pp. 3–33. DOI: `10.1016/b978-0-12-558701-3.50007-7` (cit. on p. 22).

[23] Agata Kołakowska, Wioleta Szwoch, and Mariusz Szwoch. «A Review of Emotion Recognition Methods Based on Data Acquired via Smartphone Sensors». In: *Sensors* 20.21 (2020), p. 6367. DOI: `10.3390/s20216367` (cit. on p. 22).

[24] *Save data in a local database using Room.* URL: `https://developer.android.com/training/data-storage/room` (cit. on p. 36).

[25] *Create and monitor geofences.* URL: `https://developer.android.com/training/location/geofencing` (cit. on p. 37).

[26] *Navigation.* URL: `https://developer.android.com/guide/navigation` (cit. on p. 40).

[27] *Cloud Firestore: Store and sync app data at global scale.* URL: `https://firebase.google.com/products/firestore` (cit. on p. 55).

[28] Frank Wilcoxon. «Individual Comparisons by Ranking Methods». In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83. ISSN: 00994987. URL: `http://www.jstor.org/stable/3001968` (visited on 09/22/2022) (cit. on p. 60).