Wait-Learning: Intelligent Systems for Making Productive Use of Wait Time

by

Carrie J. Cai

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author . . . . . .
Department of Electrical Engineering
and Computer Science
May 19, 2017

Certified by . . . . . .
Robert C. Miller
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Wait-Learning: Intelligent Systems for Making Productive Use of Wait Time

by Carrie J. Cai

Submitted to the Department of Electrical Engineering
and Computer Science on May 19, 2017,
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Every day, millions of people set long-term goals, such as learning a skill, developing a habit, or completing a project. Yet, in our busy, time-crunched world, most have difficulty making progress towards these goals. Despite this lack of time, there are numerous moments in a day when people wait, such as waiting for the elevator or waiting for an instant message reply. These fleeting moments could instead be used for completing short, concrete microtasks towards accomplishing long-term goals.

This thesis presents novel systems and approaches that enable wait-time productivity. First, WaitChatter enables *wait-learning*, by automatically detecting when a user is awaiting an instant message reply, and presenting foreign language vocabulary exercises during that time. Second, WaitSuite is a suite of systems that expand wait-learning to diverse kinds of waiting scenarios. Furthermore, given a variety of microtasks, I also demonstrate how chains of microtasks can be ordered in a way that maximizes efficiency and minimizes mental workload. Finally, Deadspace Finder enables peripheral microtasks to be displayed less intrusively, by automatically detecting and placing them into unused screen space. Taken together, these systems demonstrate that wait-time productivity is both feasible and more effective than traditional reminders, making meaningful use of time we didn't know we had.

Thesis Supervisor: Robert C. Miller
Title: Professor of Electrical Engineering and Computer Science

Preface

This thesis is in memory of my grandparents Sau-Nam Choi and Xiao-Lu Wang, and in honor of my mother, Ying Cai.

I learned to code in 2010, two years out of college. A few months later, I applied to the PhD program in Computer Science at MIT. Despite my initial apprehension, a combination of CSAIL's visit weekend, the Sidney Pacific website, and an evening stroll along the exquisite Charles River convinced me to take the leap to Boston. Since then, it has been an unforgettable journey. Although my career transition would not have been possible without lots of luck and a community of support, I hope this thesis proves that it is never too late to learn Computer Science, and that it is unproductive to cast people into technical and non-technical buckets. More importantly, I hope it motivates humanists and scientists to use programming as a means to solve the toughest challenges in their respective fields.

Acknowledgments

I would like to thank:

- Rob Miller, who taught me to distinguish between typical research ideas and truly innovative ones. Thank you for giving me both the support to get things done, and the freedom to be creative along the way. I am lucky to have such a responsive and organized advisor.
- Victor Zue and Stephanie Seneff, who took a chance on me and gave me a life-changing opportunity to make a career switch into Computer Science. Thank you Victor for pushing me to scope and define my thesis thoughtfully.
- Pattie Maes, for encouraging me to zoom out from my research and examine it with a more nuanced and critical lens.
- The entire UID group (a.k.a. lab siblings), for making me feel at home. You gave me a warm and collegial environment to both take risks and seek help when needed, elements that are all too often missing in traditional PhD life. I will miss our pair research, tea, Google docs, and numerous playtests. Special thanks to Elena Glassman for the daily moral support, Max Goldman for helping me solve mind-bending puzzles, and Philip Guo for your infinite PhD wisdom.
- My summer internship mentors Jaime Teevan, Shamsi Iqbal, George Zhang, Ed Chi, Dan Russell, and Michael Bernstein, who gave me a rare chance to experience research in new environments.
- Jim Glass, Leslie Kolodziejski, and Stefanie Mueller, for your mentorship and academic guidance.
- Maria Rebelo, Marcia Davidson, Adam Conner-Simons, and TIG, for all your behind-the-scenes magic that made my grad school years so much easier.
- Anji Ren, Rujia Zha, Steven Rivera, and Rob Soto, thank you for giving me the joy of mentoring. Robin Cheng and Joey Rafidi, thanks for giving me a glimpse into MIT undergrad life and for making 6.005 and 6.831 so fun.
- Small Feet, Friday Night Dinner, Sidney Pacific, Obama Fan Club, and the Spoken

Language Systems group, you make my days brighter and my snowy days warmer. Thank you for fueling my brain with food, for making me laugh, and for helping me learn not to take life so seriously.

- Lambda-bang, for making this journey possible.
- Movements in Time, MIT Lindy Hop, and MIT Contra, for giving me a healthy dance outlet on the east coast.

Finally, I would like to thank William Li for being my go-to sounding board for nearly every aspect of my life. I couldn't have asked for a more patient, brilliant, and understanding partner through the up-and-down roller coaster of grad school life.

I am extremely grateful to my family and especially my grandparents Sau-Nam Choi and Xiao-Lu Wang, who ignited and encouraged my life-long love for learning.

I owe my greatest gratitude to my mother Ying Cai, who made countless sacrifices so that I can be where I am today. Thank you, thank you, thank you for being my greatest source of support and inspiration, and for always putting my best interests above anything else in your life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every day, millions of people strive to achieve long-term goals, such as learning a new skill, developing a habit, or completing a project. Yet, in our busy, time-crunched world, most have difficulty making progress towards these goals. Unfortunately, 80% of new years' resolutions ultimately fail[1]. Similarly, drop out rates in adult education are as high as 70% [118], despite an initially high starting motivation. Despite having high initial motivation, people often lack the time to follow through on a regular basis, given more urgent priorities.

Despite the busynesss of daily life, almost everyone experiences fleeting moments in a day when they wait, such as time spent waiting at an elevator or lining up at a store. Because these waiting moments tend to be short, they are typically not perceived to be useful for doing meaningful work. As a result, wait time is often spent on activities like browsing social media, playing games, or email checking, many of which have become compulsive digital habits. These waiting moments could instead be used more meaningfully, for making incremental progress towards long-term self improvement. In particular, wait time could be an opportune moment for practicing skills that require repeated rehearsal, such as answering a vocabulary flashcard to learn a language, doing an ergonomic stretch to improve one's posture, or taking a deep breath to develop mindfulness.

[1] http://www.businessinsider.com/new-years-resolutions-courses-2016-12

This thesis introduces *wait-learning*, a novel approach for automatically detecting wait time and delivering a skill-building microtask that can be optionally completed while waiting. The proposed advantage of wait-learning is that, by triggering these tasks at moments when people are more available and motivated to perform the task, it lowers the barrier to skill-building practice. In the long term, wait-learning could increase the rate of skill acquisition, leading to greater progress towards long-term goals.

Currently, in the absence of systems to support wait-learning, wait time productivity is challenging for several reasons. First, the overhead of initiating these tasks is too high: users need to first remember to perform the task, make the decision to do it, then have ready access to a task that can be easily completed within the waiting period. Furthermore, if the task involves the use of an application, the user must then navigate to a separate application and open it, at which point the waiting period may have already ended. Secondly, current notifications and reminders tend to be poorly timed and interrupt the user in the middle of ongoing tasks, thus being more harmful than helpful. In sum, wait-time productivity currently fails because it demands *too much overhead* and is *poorly timed*.

In contrast to the status quo, wait-learning could enable productivity microtasks to be both *timely* and *easy to access*. First, by automatically detecting moments when users are waiting, productivity triggers could occur at times when users are more mentally available to complete the microtask. Second, by embedding the microtasks directly within the user's workflow, it would also decrease the decisional and physical overhead required to initiate the microtask.

Because wait-learning occurs during transient waiting moments amidst existing activities, there are potential costs and challenges to encouraging productivity during wait time. For instance, if the main activity is mentally demanding, it may be more beneficial to stay focused on the current activity. An existing activity could also be an interruption to wait-learning if waiting ends abruptly in the midst of learning. Furthermore, some people may already spend their waiting time meaningfully by relaxing or reflecting. A wait-learning task that is too intrusive could interrupt existing activites, whereas one that is too inconspicuous

may never be interacted with at all. Thus, this thesis seeks to address how wait-learning can be designed to encourage engagement while minimizing disruption, and to evaluate the cumulative effects of wait-learning, both on learning and on a user's existing activities.

Although wait-learning could in theory be applied to a wide range of long-term goals, it is most appropriate for learning tasks that are short, simple, and relatively context-free so that there is minimal interference with ongoing activities. This thesis explores and validates wait-learning in the domain of second language learning. We chose this domain because its progress can be easily measured with a digital device, which allows us to research and quantify productivity output. Furthermore, there is already existing work demonstrating the benefits of learning vocabulary through short repetitions spaced across time. Later in this thesis, we explore how wait-time productivity can be applied to more complex domains during longer periods of wait time, by chaining microtasks together strategically. We explore microtask chains in the domain of writing because writing subtasks vary widely in both content and complexity, from low-level proofreading to meaning-rich rephrasing and tone modification. Hence, this thesis explores wait-learning through applications in the domains of second language learning and writing.

1.1 WaitSuite

To explore the design space of wait-learning, we first charted a set of design dimensions for wait-learning. We then developed WaitSuite (Figure 1-1), a suite of diverse wait-learning applications that each targets a different kind of waiting, spanning dimensions in the design space:

1. **ElevatorLearner**: the user learns while waiting for the elevator.
2. **PullLearner**: the user learns after pulling to refresh mobile app content, while waiting for the content to load. Pull-to-refresh is a touchscreen gesture that involves dragging the screen downward, then releasing it to trigger a refresh.
3. **WifiLearner**: the user learns while waiting for their computer to connect to wifi.

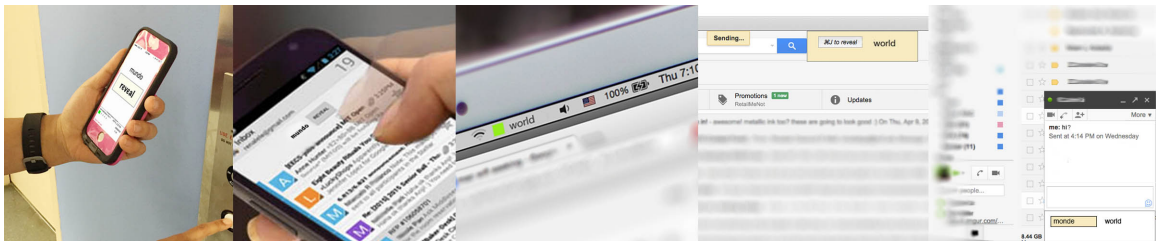


Figure 1-1: WaitSuite enables wait-learning during a variety of waiting scenarios: (left to right) elevator waiting, pull-to-refresh, wifi connecting, email sending, and instant messaging.

4. **EmailLearner**: the user learns while their email is being sent.
5. **WaitChatter**: the user learns while awaiting instant message (IM) responses.

First, to validate the feasibility of wait-learning, we developed WaitChatter (Figure 1-2), a system that extends instant messaging by automatically detecting when someone is waiting for a chat reply, and displaying an optional foreign language vocabulary exercise. WaitChatter also automatically extracts and translates words from the ongoing chat, and transforms them into flashcards on-the-fly. In a two-week field study, in which participants used WaitChatter as a Google Chat plugin on their own computers, we found that users learned an average of 57 new words over two weeks (4 new words per day) without taking extra time to study beyond their typical daily activities. Microtasks shown at the start of a waiting period were also more likely to receive a response compared to other timing conditions, validating the utility of wait time.

Second, we conducted a multi-system deployment to understand how people use WaitSuite in multiple kinds of waiting scenarios. We found that wait time, ease of accessing the microtask, and competing demands were key factors affecting engagement with wait-learning. This work produced a theoretical framework for wait-learning, illustrating the combination of constraints (wait time, ease of access, and competing demands) that is more effective for wait-learning. Furthermore, we found that the majority of people took advantage of multiple waiting moments throughout the day. Taken together, this research provides important implications for designing future wait-learning systems, and expands wait-learning beyond any single waiting scenario.

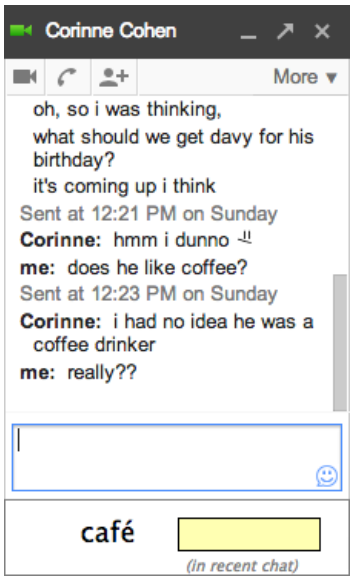


Figure 1-2: WaitChatter presents second language vocabulary exercises while the user is waiting for an instant message response.

Finally, we conducted a long-term study to measure the long-term impact of wait-learning compared to traditional reminders. To investigate the potential long-term benefits and drawbacks of wait-learning, we measured learning engagement, learning outcomes, user preference, as well as the impact of wait-learning on ongoing tasks. Results show that, compared to traditional reminders, wait-learning increases the frequency and regularity of learning practice, increases the amount learned, and is ultimately preferred by users. Furthermore, and perhaps surprisingly, we found that users continue their ongoing activity more promptly in the presence of wait-learning tasks.

1.2 Microtask Chains

WaitSuite enables people to learn short, bite-sized pieces of content during multiple kinds of waiting scenarios. Although microtasks can be performed in isolation, in practice people often complete a chain of microtasks within a single session. What if learners could complete more diverse and complex tasks during longer periods of wait time, by completing chains of microtasks? A fundamental challenge is maintaining efficiency, ease, and focus while

switching from one microtask to the next.

To investigate how chains of microtasks should be sequenced and organized, we conducted a series of crowd-based studies, using canonical microtasks in the writing process that span a range of operations, content, and complexity levels. Our studies concluded that the order of microtasks has a remarkable impact on user experience: participants completed easy microtasks significantly faster when they were preceded by microtasks with the same operation, whereas they found hard microtasks significantly less mentally demanding when preceded by microtasks on the same content (Figure 1-3). They were also faster at starting hard microtasks after first completing easy microtasks on similar content. As more and more tasks are transformed into microtasks in domains such as programming, creativity, and education, these findings provide important implications for how microtasks can be ordered to optimize transitions from one microtask to another.

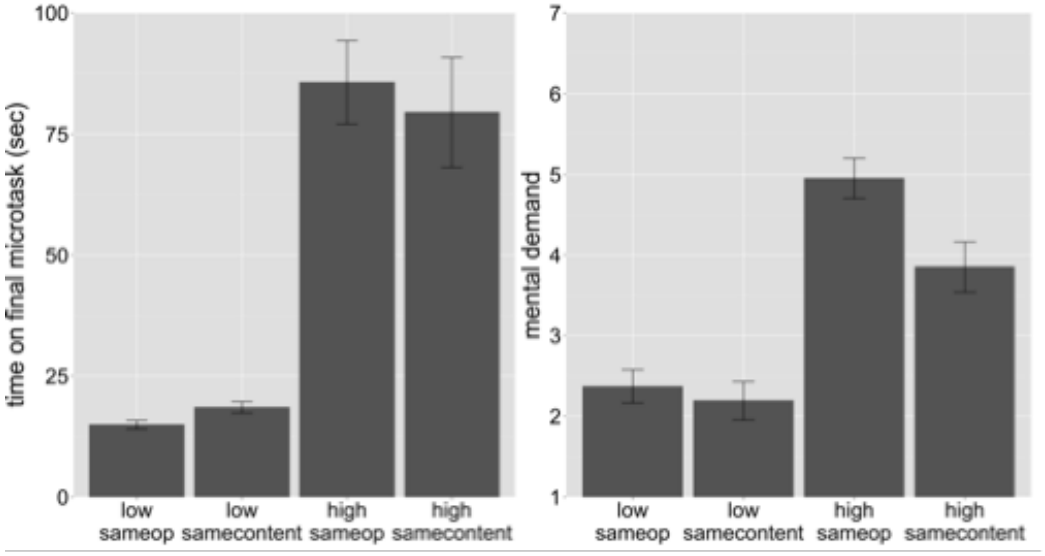


Figure 1-3: In low-complexity chains, the final microtask was completed faster in the same-operation condition. In high-complexity chains, participants found the final microtask less mentally demanding in the same-content condition.

1.3 Complementing Wait Time with Dead Space

Beyond the careful use of time, the use of *space* also affects the noticeability and disruptiveness of secondary tasks, and can often complement timing. On graphical user interfaces, limited screen space and competing tasks make it difficult to find space that is both within the user's field of view and non-disruptive to ongoing user activity.

This research introduces the idea of automatically detecting *deadspace* on the screen for the subtle display of occlusion-free, peripheral interfaces (Figure 1-4). Deadspace Finder is a novel approach for automatically finding and using the existing deadspace in any webpage a user is browsing for the subtle display of secondary content. It is implemented as a browser-based system that automatically detects deadspace, places content into deadspace while the user is focused elsewhere, and automatically removes the content if it collides with page content as a result of the page changing. Deadspace Finder is most useful for wait-learning content that would be beneficial if noticed occasionally, but too disruptive if shown in the form of typical popup notifications, such as informal learning content.

In an evaluation of Deadspace Finder on 500 webpages and a 10-day field study with 20 participants, results showed that deadspace is prevalent, and that Deadspace Finder can accurately detect unused space. Furthermore, deadspace content was perceived to be less disruptive compared to popup content, while still being noticed 73% of the times it appeared. Deadspace Finder paves the way for future interfaces that use deadspace to display wait-learning tasks more subtly.

1.4 Contributions

In sum, this thesis makes the following contributions: 1) the technical implementation of automatically detecting waiting and triggering a skill-building microtask across five waiting scenarios, 2) the design of the microtask and interface that facilitate this interaction, 3) a theoretical framework describing the combination of constraints most appropriate for

Figure 1-4: Deadspace Finder automatically detects deadspace and selects a location for placing deadspace content without occlusions.

wait-learning, 4) guidelines on how to increase microtask efficiency through microtask chaining, and 5) empirical evaluations of the effects of microtask completion during these transient moments.

These contributions draw on theories and techniques from the following fields: **peripheral interfaces** to create unobtrusive yet engaging microtasks; **attention and task management** to minimize interruption to ongoing tasks; **cognitive science** to reduce mental workload during task workflows; and **learning science** to maximize learning benefits. By combining wait time with productive work, this thesis introduces a new class of software systems that overcome the problem of limited time by leveraging existing waiting moments.

Thesis statement: Wait-learning leads to more frequent skill-building practice compared to existing alternatives, can be implemented across a variety of waiting scenarios, is preferred by users, and ultimately contributes to greater progress towards long-term goals.

1.5 Thesis Overview

Chapter 2 lays the groundwork for wait-learning by describing existing research challenges and findings in related fields.

The main part of the thesis describes the design and technical implementation of wait-learning, as well as three evaluations of wait-learning:

- Chapter 3 presents the design space for wait-learning, followed by the user interface and implementation of WaitSuite.
- Chapter 4 empirically validates wait-learning through a feasibility study of WaitChatter, demonstrating that wait-learning is effective for vocabulary learning. It also shows that timing wait-time tasks to appear at the start of a waiting period is more effective than alternative timing conditions.
- Chapter 5 is a multi-system deployment that evaluates WaitSuite using all five systems. The findings from this chapter culminate in a theoretical framework that describes the combination of constraints that is most appropriate for wait-learning.
- Chapter 6 examines how long-term usage of wait-learning compares to that of traditional reminders, and finds that wait-learning leads to more frequent skill-building practice and a greater amount learned, while being preferred over traditional reminders. In addition, wait-learning may also help facilitate in the continuity of existing activities.

Chapters 7 and 8 extend the core concept of wait-learning in two directions:

- Chapter 7 expands wait-learning to multiple kinds of microtasks, each with different properties and complexity levels. Focusing on the writing domain, Chapter 7 investigates how a chain of multiple microtasks can be best ordered to maintain efficiency, ease, and focus while switching from one microtask to the next.
- Chapter 8 presents Deadspace Finder, a novel approach for automatically detecting unused screen space so that peripheral microtasks can be displayed less disruptively. We find that deadspace is prevalent on the Web, that Deadspace Finder can accurately detect unused space, and that content shown in deadspace is perceived to be less disruptive than content shown in traditional popups.

Chapter 9 summarizes lessons learned from the sections above, discusses major design implications to assist researchers and practitioners in designing future wait-learning systems,

and proposes future research directions.

Finally, Chapter 10 concludes with the main contributions of this thesis.

Chapter 2

Related Work

The design and implementation of wait-learning draws on existing research related to motivation and behavior change, micro-learning, attention and interruption management, and the waiting experience.

2.1 Micro-Learning

A rich thread of research on *micro-learning* [59] aims to distribute learning into small units throughout a person's day-to-day life. Below, we provide an overview of learning theories underpinning micro-learning, followed by a summary of existing micro-learning applications.

2.1.1 Learning Theories

It is well known that learning a second language requires significant time and effort on a recurring basis, yet many language learners do not have the time to dedicate years of their lives to living in a foreign country or traditional classroom instruction. Recent work on

microlearning has introduced ways to distribute traditional language study into small units, using flashcard-based learning exercises presented at spaced intervals.

Micro-learning is motivated by strong evidence that retention of new information is enhanced when that information is presented in a spaced [42] and repeated [149] manner. The *spacing effect*, which posits that spacing apart learning events results in more long-term learning than massing them together, has been demonstrated in numerous experiments [12, 31, 44, 86]. According to the theory of *spaced repetition*, content should be presented in increasingly spaced intervals, so that it is encountered just before it is forgotten. Spaced repetition is based on memory research findings, which state that humans exhibit a negatively exponential forgetting curve [50]. It has been posited to benefit learning, even in the absence of contextual cues (e.g. seeing a word used within a sentence) [42].

Several approaches, such as the Pimsleur method [122], Leitner algorithm [61], and MemReflex algorithm [53] automate the scheduling of flashcards based on a history of prior exposures. In these methods, items that are known well are shown less frequently, whereas items that are known less well are shown more frequently. Despite the long-term benefits of spaced learning, such methods are seldom adopted in classroom learning practice [43]. Learners often neglect the effects of spacing when making study decisions, assuming that massed study is more effective than spaced study [49, 87]. This illusion may exist because learners temporarily perform better during massed study due to the items being presented close together in time [49, 86]. In contrast, spacing may reduce performance levels during learning while enhancing long-term retention.

In addition to spacing, *retrieval practice*, or the act of repeatedly recalling information from memory, is well known to strengthen memory. Tulving's pioneering work in 1967 revealed that tests not only assess learning, but also produce learning in ways that are more effective than simply studying or re-reading [146]. Hence, a common strategy for learning is the use of flashcards to strengthen memories by repeatedly prompting oneself to recall mappings, one card at a time. This recall strategy is posited to be more effective than simply reviewing both sides of the flashcards. Repeated retrieval spaced over time is posited to be powerful

because it offers opportunities to strengthen memory encodings through different exposures to memory cues. In some cases, retrieval practice has demonstrated an advantage even over more complex active learning strategies, such as elaborative studying with concept mapping [83].

According to theories on educational psychology, the complexity of instructional content depends on the number of interactive parts it contains [139]. For example, the study of vocabulary, computer terminology, and chemical symbols has few interactive parts, whereas learning algebra or language syntax involves more interactive parts. Instructional elements with few interactive parts impose a lower cognitive load, and can be more reasonably learned in isolation. Low-interactivity instructional elements, such as vocabulary learning, may be more appropriate for micro-learning and wait-learning since they can be learned independently from each other.

2.1.2 Micro-Learning Applications

Existing systems for micro-learning break learning into small units that can be encountered frequently in the context of a person's day-to-day life. For example, MicroMandarin [52] and Vocabulary Wallpaper [40] are mobile applications that present location-related vocabulary so that users can learn words in the context of their environment. Other mobile applications teach vocabulary that is related to nearby objects, by using RFID tags [18] or a remote human companion [63].

These micro-learning systems tend to focus more on *what* and *where* the user is learning, rather than *when* to present these learning opportunities. Despite being relevant to a learner's daily life, micro-learning can still fail to be effective if a learner is not available to learn at the time learning content is presented, or if the level of energy required to initiate learning is too high. For example, in MicroMandarin, users either received words that were contextually relevant to their location (the contextual version), or received words in order of word frequency (the non-contextual version). It was found that users learned more vocabulary in

the non-contextual version than in the contextual version, due to greater time availability while using the non-contextual version [52]. Given that learners may have a high initial motivation to learn, but lack the executive motivation to practice on a repeated basis [47], the timing of learning opportunities may be critical to a user's ultimate engagement with learning.

To decrease the barrier to learning, several systems have embedded learning into daily routines to help users form a regular habit. For example, Lernschoner [59] activates a learning program when a user's computer becomes idle, and asks users to answer a flashcard before they resume activity on their computer screen [59]. ALOE translates words within web articles a user is reading and displays them in a foreign language [144]. FeedLearn augments Facebook newsfeeds with inserted flashcard exercises, so that users can learn while casually browsing social media [88]. However, even when embedded within daily routines, learning may still be undesireable because it delays higher-priority tasks. For example, while using ALOE, some users disabled in-place translation of webpages because they felt it decreased the speed at which they could read web articles [144]. Similarly, asking the user to answer a Lernschoner flashcard before resuming computer activity could delay the user's next task, and potentially lead to abandonment of learning.

While micro-learning traditionally targets times when the user's existing tasks could conceivably continue, wait-learning instead targets times when they are temporarily blocked. Wait-learning is motivated by evidence that people need well-timed triggers to sustain desired behaviors, even if they are already motivated to perform the behavior [57]. Using wait time as a trigger, we hope to engage users at times when they are temporarily blocked on their existing tasks, making them more receptive to a secondary task [80].

2.2 Theories on Attention Management and Multitasking

Because waiting occurs amidst existing tasks, the timing and manner in which learning exercises are presented may have an impact on cognitive workload and user engagement. In

the following section, we present several theories surrounding attention management as a basis for understanding and designing for wait-learning.

2.2.1 Attentional Capacity

There have been multiple theories surrounding how attention is managed and allocated across tasks. Kahneman's resource theory posits that there is a single pool of attentional resources that can be freely divided among multiple tasks [82]. According to resource theory, our minds dynamically allocate and release resources throughout task execution, resulting in fluctuating levels of attentional capacity. Different activities receive different amounts of attentional resource depending on factors such as arousal, task complexity, and effort. For example, as a task is practiced and automatized over time, it requires less effort and less attentional capacity. In contrast to single resource theories, multiple resource theory proposes that several different pools of resources can be tapped simultaneously, such as different input modalities and stages of processing [150].

Some theories characterize attention as a bottleneck or selective filter. According to Broadbent's Filter Model, a filter selects which of many competing messages ultimately receives attention [26], separating incoming messages into those that are attended and those that are not. Only messages that pass through the filter are attended to and stored in short-term memory. Some posit that the filter makes its selection based purely on the physical characteristics or modality of the input [26], while others show that the meaning of the input is also a factor [45, 143]. For example, overhearing one's own name during a cocktail party can turn one's attention toward input that was initially unintended [36].

Because attentional capacity is finite, attempting to perform two or more tasks simultaneously can be costly. Research shows that the way in which a secondary task is presented matters. The timing of a secondary task relative to the user's ongoing task has significant effects on interruption cost, task performance, and levels of frustration [2, 13]. Many studies have shown that mental workload decreases at the boundaries between subtasks [103], and

decreases more between larger chunks of a task [13]. Therefore, presenting secondary tasks during coarse-grained boundaries, or boundaries between larger chunks of a task, is less disruptive than doing so during fine-grained boundaries [13], because there are fewer demands for mental resources. Computational frameworks have been developed for modeling and predicting performance during the concurrent execution of multiple tasks [133].

2.2.2 Task Switching and Interference

Beyond the simultaneous completion of tasks, a related line of research focuses on the effects of switching between tasks. A large body of evidence shows that there is a *switch cost* associated with switching between tasks. Task switching results in slower and more erroneous performance compared to continuing on the same task [104, 153]. This cost is due to the time required to inhibit the carry-over of activation from the previous task, as well as the time required to reconfigure to the new task. Due to the passage of time, reactivation of that task may also be more error-prone.

Furthermore, it has been shown that task switching is very common. On average, information workers spend on average 11 minutes on a work task before switching tasks, and 57% of work tasks are interrupted. On average, people check their phones 47 times per day[1]. Research shows that interruption from a secondary task harms subsequent performance on the primary task [13, 60] and increases stress [99], even though people tend to underestimate the cost of these interruptions [75].

Recently, researchers found that the need to remember a *problem state* affects the disruptiveness of secondary tasks because it adds to the execution time of task switching [25]. Problem state refers to the temporary, intermediate information necessary to perform a task. For example, during multi-column addition, one might need to remember the problem state of whether to carry a one. Under this framework, interruption effects are stronger when both the primary and secondary task have a problem state. Because only one chunk of

[1] https://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/global-mobile-consumer-survey-us-edition.html

information can be stored in the problem state module at a time, the problem states have to be swapped in and out frequently if multiple tasks require problem states [7, 24]. Moving a problem state to and from declarative memory requires processing time, adding to the cost of switching tasks. The overhead of problem state retrieval can be decreased if the state is encoded in the user's environment. For example, driving directions can externalize problem state by displaying arrows for intermediate turns so that the user does not need to remember which steps they've already taken.

2.3 Multitasking and Motivation

In light of the cognitive costs associated with multitasking, an emerging body of work has examined the motivational reasons behind why people multitask.

2.3.1 Multitasking as Emotional Fulfillment

Recent studies suggest that much of task switching is driven by emotional or motivational goals, rather than cognitive or performance goals [89, 148]. This motivation is driven by a drive to maximize positive affect through strategic activation of the appetitive system, and avoid negative affect through the aversive system. For example, self-interruptions can function as a desirable break from frustration, fatigue, or boredom with a primary task [80]. Students might exchange texts with friends to balance against the tedium of doing homework. Similarly, someone waiting in line might check their email to offset boredom and cope with their aversion to waiting. People may welcome or even seek out a secondary task if it helps them establish homeostasis or maintain internal equilibrium [100, 114]. According to Wickens et al's computational decision model, people are more likely to switch to secondary tasks that have high salience and low difficulty [151].

2.3.2 The Waiting Experience

There is strong evidence that waiting has a measurable effect on user satisfaction, mood, and level of frustration [41]. The longer the wait time relative to the expected waiting time, the lower the customer satisfaction [41, 98]. Due to the negative experiences associated with waiting, many approaches have been proposed to improve the waiting experience. Some minimize actual wait time using optimization algorithms [94], while others seek to improve subjective experience. Based on evidence that occupied time feels shorter than unoccupied time, and finite waits feel shorter than uncertain waits [98], a large number of approaches have been used to help improve the waiting experience. These range from filling wait time with music [70], content [5, 84, 116], and wait time estimates [84], to redesigning progress bars that alter the perception of time [66].

While these approaches primarily seek to increase satisfaction with the service provider, our work on wait-learning seeks to benefit the users themselves, by enabling personal productivity during waiting periods. Because filled waiting periods are perceived to be shorter than unfilled waiting periods [98, 109], wait-learning could enhance a user's subjective experience while waiting, in addition to helping a user make progress on learning goals. Indeed, people naturally task switch while waiting because they are temporarily unable to make progress on the primary task [80]. Given the motivational underpinnings for task switching, wait-learning may be more engaging if learning exercises are presented during types of waiting that are particularly frustrating or boring.

2.4 Peripheral and Ambient Interfaces

In contrast to interruption management systems, peripheral and ambient interfaces deliver non-critical information [126], often at the periphery of attention. While the completion of a wait-learning task requires focus, the act of noticing or ignoring a wait-learning opportunity is intended to involve only partial attention. Thus, we draw insight from existing work on peripheral and ambient interfaces in designing for wait-learning.

Peripheral interfaces aim to blend effortlessly into daily routines, by supporting shifts between the periphery and the center of attention [15, 51]. Facilitating peripheral interactions requires taking into account the user's context and understanding the mental resources required in existing routines. In particular, peripheral interactions should be easy-to-initiate and easy-to-discard. In other words, the interaction should involve minimal start-up time, and should not require extra effort to abandon. More recently, some have proposed designing more adaptive systems to support *casual interactions*, by allowing users to decide how much they would like to engage [123], or *implicit human computer interaction*, by automatically sensing a user's situational context and adapting to it [134].

In particular, ambient interfaces seek to provide a more passive way of providing subtle awareness, such as displaying ambient information on wall displays [106], showing personal analytics in a user's environment to motivate behavior change [35, 58, 79], or mapping sound and light patterns to changes in background information [78, 107]. In particular, the use of change blindness [129, 137] has been proposed as a potential way of delivering information without demanding a user's immediate attention, such as using scene changes so that there is no detectable motion when a change occurs [71].

2.5 Implications for Wait-Learning

Taken together, existing work suggests that there is a cost to switching from a primary task to a secondary task and back, and that peripheral interactions should be designed to be easy-to-initiate and easy-to-discard. Furthermore, the interruption cost is higher if the ongoing task utilizes substantial attentional resources or demands memory of an intermediate state at the moment the second task is introduced. Waiting is a state in which users may feel motivated to switch to another task, given the inability to make progress on the primary task.

Thus, wait-learning may be most engaging if users perceive a waiting period, and if the expected benefits of learning while waiting offset the cost of task switching. To minimize switch cost, wait-learning should be designed to take advantage of moments when problem

state is low and attentional resources are high. Low attentional capacity can potentially be offset by longer waiting times, during which there is not only more time available to switch, but also more waiting to endure and more opportunity to learn. Finally, wait-learning can be more effective if the wait-time task is simple, with as few interdependent components as possible.

2.6 Motivation and Behavior Change

It is well known that changing one's behavior is challenging. According to B.J. Fogg, behavior change requires that individuals be both able to perform the desired behavior, and motivated to do so [57]. Motivation also comes in different forms. Although many people have a high initial *choice motivation* to make changes, many lack the *executive motivation* to actually follow through on those changes [47]. Thus, in many cases, behaviors require the support of behavioral triggers, such as a notification or announcement, which become associated with the target behavior over time. Indeed, it has been demonstrated in practice that automatic triggers help people sustain desired habits better than passive applications [20].

An often missing element in behavior change is the *timing* of a persuasive trigger [57]. To be effective, a trigger ought to be timed at a moment when a person is both motivated and able to perform the target behavior, placing the person above a behavior activation threshold. Triggers that appear when a person is unmotivated or unable to perform the target can be distracting or even frustrating [57]. For example, pop-up ads are annoying because people rarely have the motivation to do what they ask for. Exercise reminders can also be frustrating if a person is motivated to exercise, but is unable to at that moment because of an urgent deadline. In this thesis, we view the timing of a trigger as being particularly critical given that users may have a long-term motivation to learn a new skill, but lack the in-the-moment ability or motivation to do so given many competing deadlines and other priorities.

2.7 Microtasks

Microtasks are small units of work designed to be completed individually, eventually contributing to a larger goal [33]. Beyond education, microtasks are used in a broad range of domains for many different purposes. In this section, we present an overview of microtask usage in general, as well as the challenges involved in chaining together multiple microtasks.

Historically, task decomposition has been used to support personal information management and task sharing among collaborators. For example, a person can organize a personal photo collection by completing a series of pairwise comparisons [141]. Breaking large tasks down into smaller ones can help people complete their tasks more easily, by presenting more concrete, achievable goals [4]. Microtasks can also be useful for collaborating and dividing responsibility [115, 125]. Small tasks have also been used to support a wide range goals related to personal well-being, such as health tracking and logging, exercise, experience sampling, and rehabilitation [17, 54, 117, 145].

Recently, microtasks have been used to enable crowds of unknown workers to complete large tasks, from copy-editing [21], transcription [90] and writing [48, 113, 142], to creative work such as prototyping [91] and video creation [85]. Some also leverage the wisdom of many people to provide better information than any individual alone, such as gathering sensor measurements of a city [95, 147, 154], estimating numerical values [138], or contributing to volunteer efforts [65, 112].

2.7.1 Microtask Chains

Prior work suggests that the order in which microtasks are done impacts people's ability to perform them. This is because transitions between consecutive cognitive tasks have measurable effects on ongoing mental processes. Lasecki et al. found that interjecting contextually relevant microtasks and delays can hurt worker performance [92]. While iterating on similar tasks can help people build familiarity and expertise [15][34], long

chains of similar tasks can lead to boredom and fatigue [97, 115].

Recent work has sought to improve people's experiences with microtasks by inserting micro-diversions to provide timely relief during long chains [39]. Large organizations have also explored re-designing assembly lines to build task specialization while still enabling task switching and creativity [3]. Others have aimed to mitigate the effects of task interruptions, by locating more optimal moments for interruptions [2, 74]. On crowd platforms, priming effects [105] and monetary interventions [155] can also improve performance. In our approach, we examine how to optimize transitions through the ordering of existing microtasks.

The process of writing consists of uniquely demanding subtasks that require transforming abstract concepts into prose that someone else will understand [68]. Task ordering appears particularly critical in this domain; the process of editing text can lead a writer to develop new insights about the text [56], and the very process of engaging in a task can increase self-efficacy and the motivation to continue [16]. This suggests individual writing microtasks are not done in isolation, but rather are affected by a person's experience with neighboring microtasks. While recent work has explored ways to scaffold complex writing tasks by breaking them into subgoals [21, 113, 142], in our work we assume an existing collection of microtasks and investigate the effects of ordering. For instance, simpler microtasks could be performed first, with the potential side effect of helping people ramp up to more difficult microtasks. Indeed, workflows that create short-term goals have been shown to help procrastinators by increasing the perceived likelihood of success [1, 55].

In summary, prior research has shown that microtask transitions are important. Writing, in particular, is a domain where thoughtful chaining seems likely to be important. This thesis builds on previous work by examining how microtask chains can support continuity, help people transition between different microtasks, and allow them to ease in to the overall process.

2.8 Occlusion and Space-aware Interfaces

In contrast to WaitSuite, which focuses on timing, Deadspace Finder explores the spatial dimension of microtasks, and builds on existing research related to mitigating occlusion. In 2D desktop environments, window managers provide ways to retrieve overlapping windows using taskbars or docks. Occlusion can also be mitigated by positioning windows in regions with less overlap [19], by using transparency to show occluded windows [124], or by selectively making unimportant window areas transparent [77]. While these approaches focused on occlusion in primary tasks, and require considerable knowledge of the underlying objects, Deadspace Finder instead uses deadspace opportunistically on the web for peripheral interactions, relies primarily on a screenshot, and is easy to install via a browser extension.

Recent work makes webpages easier to see by automatically magnifying them, eliminating unused space in the margins [22]. This approach iteratively increases web content until it detects problems such as overlapping DOM elements. Deadspace Finder instead detects deadspace anywhere on the page and also automatically monitors collisions over time. This can be useful given the diversity of web layouts and the many ways in which they could change due to user interaction.

In the next chapter, I present a design space for wait-learning that builds on this existing body of research. Guided by the dimensions of this design space, I show how we designed and implemented wait-learning within five diverse waiting scenarios within this space.

Chapter 3

WaitSuite

Because waiting occurs within the context of existing activities, a core challenge of wait-learning is designing wait-time interactions in a way that enables awareness of a productive opportunity, without interrupting ongoing tasks. In particular, wait-learning involves understanding when and how to trigger the learning task, so that switching from the primary task to the learning task and back is as seamless as possible. In our design process, we leverage a large body of existing theory in the domains of attention management and multi-tasking, and consider issues such as attentional capacity and switch costs. We establish a design space for wait-learning, charting design dimensions that characterize both the waiting moment and the learning interaction.

Waiting occurs for a variety of reasons, such as procedural delays (e.g. waiting in line), software inefficiencies (e.g. waiting for email to load), and social delays in communication (e.g. waiting for an instant message reply). To validate the feasibility of wait-learning and to understand which factors make wait-learning more effective, we selected five diverse waiting scenarios within the design space to explore further. We then created WaitSuite, a suite of wait-learning apps each targeting one of those waiting scenarios:

1. **ElevatorLearner**: the user learns while waiting for the elevator.
2. **PullLearner**: the user learns after pulling to refresh mobile app content, while waiting

for the content to load.

3. **WifiLearner**: the user learns while waiting for their computer to connect to wifi.
4. **EmailLearner**: the user learns while their email is being sent.
5. **WaitChatter**: the user learns while awaiting instant message (IM) responses.

Below, we describe the design space for wait-learning, our selection of wait-learning scenarios to explore further, the design and implementation of these five wait-learning systems, and the technical details of how we unified them into WaitSuite.

3.1 Design Space

Designing wait-learning interactions involves decisions surrounding both the *waiting moment* (which kind of waiting to use) and *wait-learning interaction* (how the user interacts with learning exercises during wait time). In this section, we describe the possible design space of waiting moments and wait-learning interactions, and explain our rationale for narrowing down each dimension of the design space to a subspace that is more suitable for wait-learning. Some decisions were based on existing research literature on attention management and learning, whereas others were informed by feedback on early design iterations.

3.1.1 Waiting Moment

To chart the space of waiting options, 10 researchers in our group brainstormed a list of 20-30 kinds of waiting (Figure 3-1). They were asked to individually write down different kinds of waiting they experience in a day. Observing that these waiting moments varied widely with respect to both *wait time* (duration of waiting) and *frequency* (how often the waiting occurs), we then further sorted each kind of waiting by wait time and frequency. These wait types are not intended to be exhaustive, but rather give a general sense of different categories of waiting that may exist. In this section, we describe various dimensions of

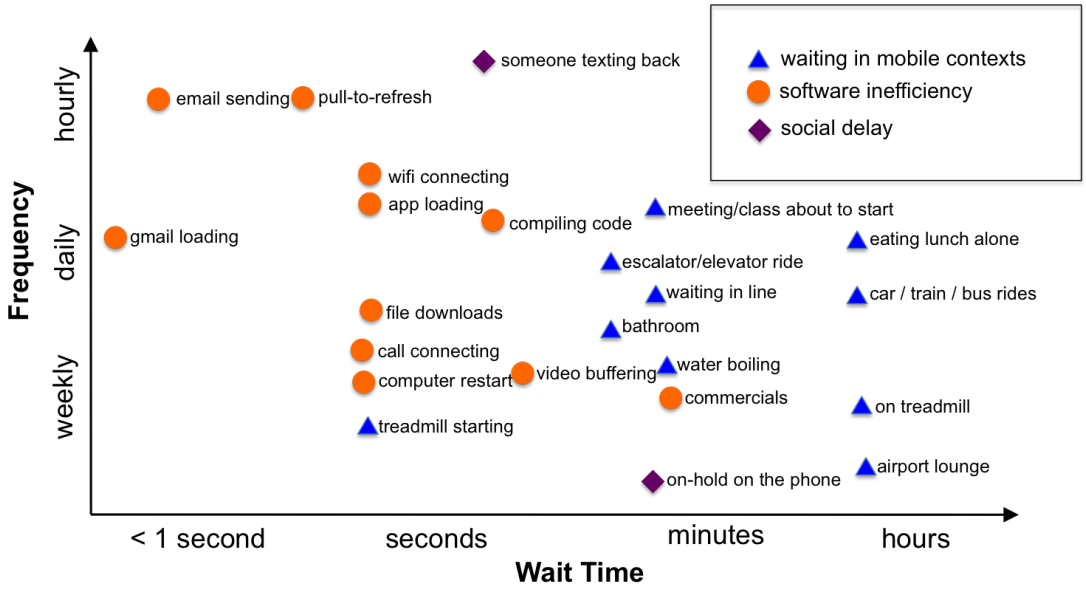


Figure 3-1: A diverse set of brainstormed waiting moments, displayed by wait time and frequency (in log scale). Some waiting moments occur in mobile contexts, while others are due to software inefficiencies or delays in social communication.

waiting moments and explain why we focused on particular subsets within each dimension.

Wait Time

Wait times can range from a few seconds to a few hours. To make the benefits of wait-learning offset the switch costs, wait time should be at minimum several seconds long to give the user enough time to switch into a learning activity.

From our initial brainstorm, we noticed that longer waits tend to occur in mobile contexts (e.g. waiting in line), whereas shorter waits often happen as a result of in-app inefficiencies (e.g. app loading). During waiting in mobile contexts, the physical switch cost may be high despite high attentional resources being available. For example, someone in line at a grocery store might not be mentally occupied, but may instead be holding items in their hands, and not attending to their device. Switching to learning would require the physical effort of pulling out the device and transferring one's gaze. Conversely, technical inefficiencies tend to occur while a user is already on a device or inside an app. Thus, switching into a

digital learning activity could require less physical effort. However, users might also be more mentally consumed by existing activities on their device, and moreover have less time to task switch during short waits. Given these tradeoffs, we explore both ends of the spectrum, from short, in-app waits to longer periods of waiting in mobile contexts. We do not explore hour-long waits, as these tend to be waits that are activities in and of themselves (e.g. running on a treadmill).

We additionally explored a class of situations where wait time is negligible, but competing mental demands may be low. In early iterations, we found that the moment after pressing Send on an email, form submission, or social media post is a time when users may have just completed one task (sending an email) before formulating a new goal. Many interfaces display a throbber while the item is being sent, causing a moment of waiting as the user watches the screen to ensure that the item in fact got sent or submitted. In this case, we do not expect the fleeting moment of waiting itself to be long enough for completing the exercise. However, if the moment coincides with a coarse-grained task boundary, with little need for remembering a problem state, the user could notice the learning opportunity while waiting, then complete the exercise between tasks.

Frequency

Some wait times occur regularly, whereas others are encountered only occasionally, e.g. movie previews, or only by certain groups of people, e.g. code compiling. A considerable body of research has shown that spaced, repeated exposure to educational content aids memory retention [42]. Therefore, we limit our exploration only to the subspace of wait types that occur at least once a day on average in order to have a meaningful impact on learning, since users will likely only respond to a subset of these opportunities.

Competing Demands

As described in related work (Chapter 2), secondary tasks are less likely to be disruptive if delivered at the boundary of coarser-grained tasks, or if the primary task demands low attentional capacity and low memory of problem state. For example, waiting for code to compile may consume high attentional capacity because a user must remember which bug they were trying to fix, which files and lines they are working on, and may also be contemplating how else to fix the bug. Conversely, taking the elevator tends to consume less attentional resources. The process of pressing the button and entering the elevator is almost automatic, and the user simply needs to remember where they were going next.

We filter out waiting situations that typically occur amidst tasks requiring high attentional capacity. Competing demands can also be higher during work-related or urgent tasks, because a user may devote more attentional resources to activities for which more is at stake. Therefore, we also filter out waiting scenarios that usually occur within a work-related setting (e.g. file downloading, code compiling).

Main Task Resumption

A problem with secondary tasks is the switch cost associated with resuming the primary task once the secondary task ends. As described in related work (Chapter 2), the user needs to inhibit activation of the secondary task, activate the primary task, and in some cases, retrieve the problem state of the primary task from memory [6, 25, 104, 153]. To minimize the effort required to retrieve problem state, we exclude waiting moments that do not display the intermediate state of the main task to the user, in cases where the main task has a strong problem state that must be retrieved. For example, a user who is learning while waiting for a phone call to connect might forget the purpose of the phone call in the absence of visual reminders. In contrast, instant messaging applications typically show a chat history that visually reminds the user what was last said. In cases where the original task intent cannot be externalized or visually conveyed, such as after wifi connects, the main task ought to be

one that demands low problem state to begin with.

While prior work on interruptions prioritized primary task resumption, in our work we also balance against interruptions to the learning task. An abrupt resumption of the primary task could disrupt the learning process. It could also interrupt the flow state [38] in cases where users are completing multiple flashcards in a row. To support the completion of the learning task, we exclude primary tasks that resume abruptly or demand the user's immediate attention when they resume. For example, a user waiting for a phone call to connect must promptly attend to the conversation once the waiting ends. Conversely, a user receiving an instant message reply can delay responding, because the other party is usually not aware of when messages are received [111]. In the case of elevator waiting, a user can continue doing more learning exercises after waiting and during the elevator ride, allowing for a gradual rather than abrupt transition back to the primary task.

Absorption into the secondary task could also delay resumption of the primary task. For example, a user waiting for an instant message reply might switch browser tabs to Facebook, and become so absorbed in Facebook activities that the chat conversation is abandoned or delayed. This behavior occurs not only in cases when a user has overestimated wait time [80], but also when monitoring the primary task is made difficult because it is no longer in view. Hence, rather than occluding the main task, the learning task should appear in a multiplexed manner [23], so that people can continue to view and monitor the main task even while completing the learning task.

3.1.2 Wait-Learning Interaction

Learning Task

In theory, wait-learning tasks could range from bite-sized pieces of knowledge to a full curriculum with complex concepts. However, according to education research, instructional elements with few interdependent parts are easier to learn in isolation [139]. For example, the study of vocabulary, special terminology, and chemical symbols impose a lower cog-

nitive load in comparison to learning algebra, which has more interdependent parts [139]. Furthermore, as described in related work (Chapter 2), interruption costs are lower when the secondary task does not require maintaining an intermediate problem state [25]. Therefore, wait-learning is best used for simple, context-free tasks, and is less appropriate for elaborate, complex tasks with many interdependent parts. This thesis applies wait-learning to vocabulary learning, which has low problem state because each word can reasonably be learned in isolation [139].

Manner of Appearance

Peripheral interactions should be designed so that they are easy-to-initiate and easy-to-discard [15]. Thus, the learning task should appear subtly at the periphery of the user's attention, and should also require little effort to ignore.

A learning exercise could be displayed as a hard notification, soft notification, or static notification. Prior work defined soft notifications as ones that appear gradually [152]. In this work we distinguish these categories more concretely as follows: hard notifications involve both animated panels and animated text (e.g. pop-up notification), soft notifications keep the panel static but animate only the text (e.g. news ticker), and static notifications keep both panel and text static (e.g. banner ads). Prior research has shown that hard notifications interrupt users suddenly, tend to increase user annoyance [14], and can result in a severe loss of productivity [13]. On the other hand, a completely static notification may result in the user becoming habituated to the display over time [81], leading to low engagement and potential abandonment.

After feedback from pilot studies [29], we decided on a soft notification that keeps a static, always-present panel containing text that subtly changes appearance. To minimize disruption while encouraging learning, the text should gently change its appearance but stay within its static, dedicated space. If ignored, the text returns to its default state after a period of time so that no user action is required to dismiss it. In the designs we implemented, the text changes from a default string to the vocabulary prompt (e.g. “cat") when waiting is detected.

We show the prompt so that the user can immediately see the word and decide whether to engage, without leaving the main task. In cases where this is not possible, an actionable phrase (i.e. “Translate a word!”) could be displayed instead.

To maximize the ease of both initiating the learning task and resuming the primary task, the learning component should be positioned near the most likely locus of attention, such as below the last line of chat, or in dead space near a loading icon. If requiring manual interaction, it should also be physically easy to reach, near where the user’s hands already are during the main task.

Modality

Wait-learning tasks could be delivered and performed through various modalities, such as through visual, auditory, speech, or haptic channels. Prior research suggests that a parallel task is less disruptive if its modality is different from that of the ongoing task, such as hearing a sound while typing. This is based on the hypothesis that using a different modality (e.g. auditory channel) allows the user to still associate the suspended task with an environmental cue (e.g. in the visual channel). However, these findings have been based on the assumption that the secondary task occludes the primary task if presented in the same modality. The interrupting effect may be diminished if the ongoing task state remains visible to the user, even if the secondary task is visual as well.

Furthermore, it is equally important to consider the practical and social constraints of the waiting context. For example, a user may find it socially inappropriate to receive auditory cues or to speak vocabulary words out loud when surrounded by others in the same room. A wait-learning task that requires users to put on headphones may deter them from learning altogether, due to the additional overhead required. Thus, from a practical standpoint, speech and audio wait-learning tasks are more appropriate if the user is *already* in that modality (e.g. during a loading podcast, or a video advertisement), so that the user is already prepared. In this thesis, we focus on visual tasks because they are least likely to cause social disruption or privacy issues. In addition, while smartwatches and wearables are becoming

increasingly common, we deliver wait-learning tasks on mobile phones and computers as they are currently the most widely used.

3.1.3 Target User

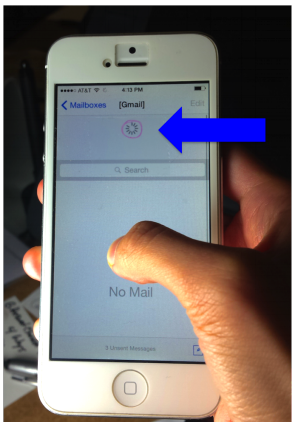
It is increasingly common for people to engage in technology use throughout the day, such as texting, browsing social media or checking email during fleeting moments. Due to the pervasiveness of technology use, we feel that wait-learning would help a wide range of people make more meaningful use of wait time, particularly those who tend to already engage in compulsive technology use while waiting. However, there is a potential cost to replacing wait time that is spent resting or reflecting. Thus, while wait-learning could theoretically be used by anyone, it is less appropriate for those who already spend wait time meaningfully.

3.1.4 Selecting Waiting Scenarios

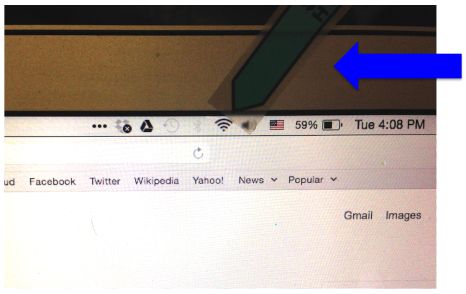
Within the design space defined above, we selected a set of reasonable waiting scenarios in an iterative process. Because the design space has high dimensionality, and the dimensions are already constrained by existing interactions during waiting, we could not manipulate each dimension in a controlled way without fundamentally changing the primary task. Instead, we explored a variety of wait-learning opportunities that met the criteria we defined above while spanning a range of wait times, frequencies, and likelihood of competing demands.

Tallying the Frequency of Waits

While some kinds of waiting occur multiple times per day, even within a single conversation (e.g. waiting for IM replies), for other types of waiting the frequency of occurrence may be much lower. To determine which kinds of waiting occur at least once a day, we conducted



(a) An example of pull-to-refresh tallying: users placed a marker over the spinner that appears during pull-to-refresh as a reminder to tally the waiting moment.



(b) An example of wifi connection tallying: users placed a sticker next to the wifi icon on their laptop as a reminder to tally whenever the wifi icon blinked.

Figure 3-2: For each waiting scenario, users pre-marked the screen location where they were most likely to be looking during the wait so that they could remember to record the wait. To tally the wait, users took a timestamped screenshot.

two rounds of data collection with ten users (round 1) and eight users (round 2), who were students and faculty at the Massachusetts Institute of Technology.

Participants tallied the daily frequency of each of the following situations: slow mobile internet, in-app pulls-to-refresh, computer wifi-seeks, emails sent, and elevator rides. We also collected usage logs with the number of outgoing phone calls, file downloads and videos watched (as an upper bound on waits for video ads). In cases where waiting moments could not be captured in usage logs, we needed a low fidelity approach to collect data from users without implementing full in-app extensions. Short waiting moments may not be salient enough for users to remember to record them. As a solution, we asked users to place and mark a piece of tape on the part of their screen where they are most likely to be looking during the wait. The unusual appearance of tape served as a reminder to record the wait. For example, users placed a small piece of tape over the progress icon that appears during a pull-to-refresh, as shown in Figure 3-2. Users took a timestamped screenshot to tally their waiting moment in a single action without leaving their current task.

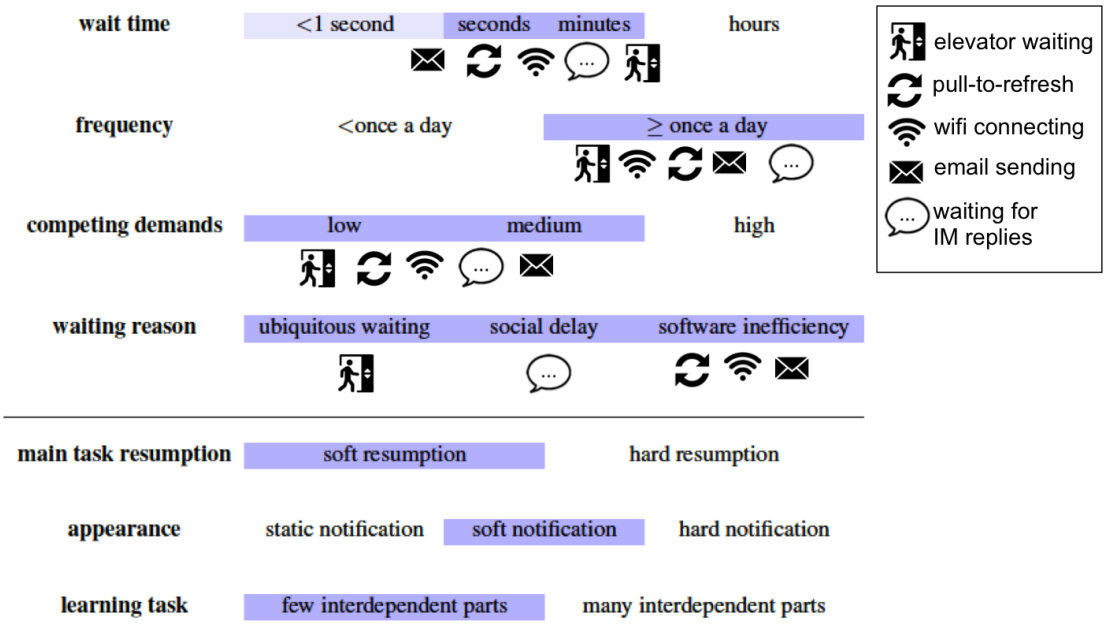


Figure 3-3: The design space of wait-learning applications. Our design choices are highlighted in blue. The five waiting scenarios we selected vary in wait time, frequency, competing demands, and waiting reason. For the dimensions listed below the line, we required all five scenarios to meet the requirement shaded in blue. Due to its exploratory nature, the class of situations with wait time shorter than a second is shaded in a lighter shade of blue. We consider these situations only when they happen to mark the end of a task, e.g. email sending or form submitting.

Situations encountered at least once a day on average included computer wifi seeks (1.6, σ=1.2), pulls-to-refresh (6.24, median=1.8, σ=11.5), elevator rides (3.0, σ=2.1), and sending emails (10.6, σ=6.5). Outgoing phone calls were made about once a day (0.94), but only 4 of the 10 users regularly made phone calls. Slow mobile internet, slow file downloads, and video ads were encountered less than once a day.

3.2 User Interface Design

Based on these design decisions and tallying results, we selected five instances of waiting within the design subspace we have defined. We created one user interface for each: ElevatorLearner, PullLearner, WifiLearner, EmailLearner, and WaitChatter. Figure 3-3

shows the design space we considered, the subspace (highlighted in blue) that we believe to be appropriate for wait-learning, and our initial estimates of where the five waiting scenarios lie on those spectrums.

3.2.1 Selected Waiting Scenarios

The five waiting scenarios vary with respect to wait time, frequency, likelihood of competing demands, and waiting reason. **Wait time** ranges from several minutes, e.g., elevator waiting, to a few seconds or less, e.g., pull-to-refresh and email. The **frequency** of these interactions ranges from waiting that could happen very frequently, e.g., instant messaging, to those that occur only a few times a day, e.g., elevator waiting. With regards to **competing demands**, we filtered out tasks that were necessarily high-stakes or work-intensive, e.g., code compiling, to keep mental workload low, but included settings that are sometimes work-related and sometimes not, e.g., instant messaging and email sending. The **waiting reason** also varies across waiting scenarios. Some occur in mobile contexts, e.g. elevator waiting, while others are due to social delays, e.g. instant messaging, or software inefficiencies, e.g. wifi, email-sending, and pull-to-refresh. While not necessarily due to in-the-wild waiting, pull-to-refresh might also occur in on-the-go settings. Lastly, for reasons discussed earlier, we keep the **learning task** bite-sized and design the **appearance** of exercises to use soft notifications for all five kinds of waiting. For practical reasons, we did not investigate scenarios where waiting could not be easily detected automatically, e.g., computer start-up, app-loading, and water boiling, or where we were unable to programmatically create dead space within the task environment, e.g., browser page loads.

3.2.2 Example Usage

WaitSuite integrates these wait-learning situations into a unified system. The range of contexts in which they occur leads to varying physical and mental constraints, and thus unique interaction experiences. Below is an example of how someone might use WaitSuite

(a) (b) (c) (d) (e) (f)

Figure 3-4: Components of WaitSuite vocabulary exercises. a) In Study Mode, user clicks Reveal to show the translation. b) After clicking Reveal, the translation is shown. c) The user indicates whether they already knew the word. d) In Quiz Mode at the easy level, the user translates from L2 to L1. e) In Quiz Mode at the hard level, the user translates from L1 to L2. f) The user can optionally click the arrow to fetch another exercise.

in the course of a day:

Judy is a college student looking to learn some French vocabulary before her trip to Paris next year. She decides to try WaitSuite because she is already juggling five classes and can't keep up the habit of reviewing flashcards on her own. In the morning, Judy enters the building where her History class is located. While waiting for the elevator, she receives a WaitSuite notification on her phone and starts doing flashcards. During the elevator ride, she continues answering flashcards until she arrives at her floor.

After sitting down in class, she puts her phone away and opens her laptop. While waiting for her laptop to connect to WiFi, she continues doing a few flashcards where she left off. During the few minutes before class starts, she instant messages her friend Wilma, asking to meet up for lunch. While waiting for Wilma's reply, she completes some more flashcards. She then remembers to ask Bob for lecture notes. After sending Bob an email, she completes more flashcards before going on Facebook. Later, while standing in the lunch line, Judy pulls to refresh email on her phone. While email is loading, she does one more flashcard.

3.2.3 Vocabulary Exercises

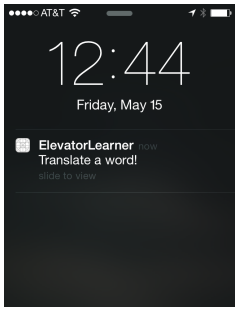
WaitSuite supports the learning of second language vocabulary, though it could be extended to other flashcard-style learning tasks. A word is displayed in *study mode* the first time it is presented, and in *quiz mode* during subsequent exposures. In study mode, the L2 (second language) is shown as the prompt (Figure 3-4a) and the user presses Reveal to see the L1 (native language) translation target (Figure 3-4b). After revealing the new word, the user

indicates whether they already knew that word (Figure 3-4c). If not, the word is added to the user's vocabulary list and later repeated for learning. In quiz mode, exercises are displayed at either the easy or difficult level. At the easy level, the user is shown the L2 prompt and types L1 (Figure 3-4d). On a mobile interface, the user self-grades by pushing Reveal, followed by Right or Wrong. At the difficult level, the prompt is L1 instead (Figure 3-4e). On desktop interfaces, users can submit a blank response if they don't remember the word.

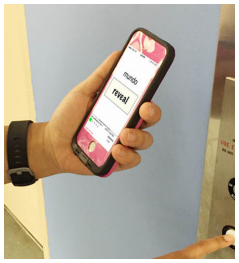
After the user completes an *initial* exercise, a *follow-up* one can be fetched by clicking an arrow or hitting the Enter key (Figure 3-4f). Fetching follow-up exercises can lead to chains of consecutive exercises. This feature was a result of early pilot testing [29] during which users indicated that they desired to learn more words during longer waiting periods.

3.2.4 ElevatorLearner

ElevatorLearner is an iPhone app that notifies the user to learn when it detects the user is near an elevator. ElevatorLearner detects elevator proximity by sensing Bluetooth iBeacons placed next to each elevator. This may become unnecessary in the future as indoor localization becomes more precise. When an iBeacon is detected, the app sends the user a notification with the message "Translate a word!" (Figure 3-5a). The notification appears either on the lock screen or, if the phone is unlocked, as a notification that slides in from the top. The user can swipe the lockscreen notification or manually open the app to access it (Figure 3-5b). To avoid sending the user a notification while exiting an elevator, we prevented any additional notifications from firing within 3 minutes of a notification, which we found was longer than a typical elevator ride. As described in Section 3.2.3 above, the user sees the vocabulary prompt in L2 at the easy level, and L1 at the difficult level (Figure 3-5c). The translation is displayed after the user hits Reveal, at which point he or she self-grades by pressing the buttons below the translation (Figure 3-5d). After self-grading, the next flashcard is shown, which the user can either engage with or ignore by leaving the app.



(a) When the user walks into an elevator area, a Bluetooth iBeacon triggers a mobile notification.



(b) The user can then swipe the notification to open the app.

(c) The flashcard prompt shows the L2 word at the easy level and the L1 word at the difficult level.

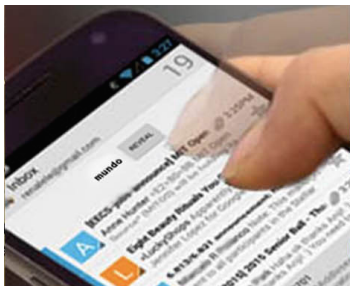
(d) After the user pushes Reveal, the translation is shown. The user self grades by pushing Right or Wrong.

Figure 3-5: ElevatorLearner user interface.

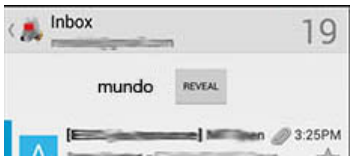
3.2.5 PullLearner

PullLearner was built on top of K9Mail[1], an open-source Android email client. It augments the existing pull-to-refresh mechanism with vocabulary exercises that display when users swipe down to refresh their email. The learning panel appears in the dead space above the inbox, where normally a "Loading..." label appears after the user pulls (Figure 3-6a). The exercise remains visible to the user for fifteen seconds, during which it is dismissed if the user swipes up or views an email. The learning panel retracts after the exercise is dismissed or completed. Pulling again fetches the next exercise but also triggers another email refresh. This design was informed by an exploratory study on a prototype built using animation software [128]. Similar to ElevatorLearner, the user presses Reveal to see the translation (Figure 3-6b), then presses right or wrong to self grade (Figure 3-6c), after which the learning panel retracts. The user can optionally fetch a follow-up exercise by pulling again.

[1]https://github.com/k9mail/k-9



(a) The learning panel appears in the dead space that is uncovered during a pull-to-refresh.



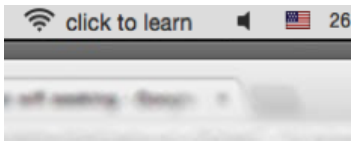
(b) The flashcard prompt shows the L2 word at the easy level and the L1 word at the difficult level.

(c) After the user pushes Reveal, the translation is shown. The user self grades by pushing Right or Wrong.

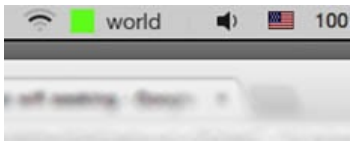
Figure 3-6: PullLearner user interface.

3.2.6 WifiLearner

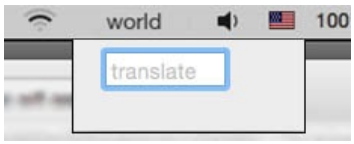
WifiLearner is a Mac application that displays a learning prompt when it detects that the computer is seeking a wifi connection. Since users typically glance at the wifi icon to see when internet has connected, we place WifiLearner next to the wifi icon in the menu bar. By default, WifiLearner displays “click to learn” (Figure 3-7a). When wifi is seeking, this text changes to show the prompt, and a colored square simultaneously blinks yellow and green to draw attention to the learning opportunity (Figure 3-7b). If the user clicks to start an exercise, a learning panel appears directly below, where the user can type the translation (Figure 3-7c). If the user ignores the prompt, WifiLearner returns to its default state once wifi is connected. The learning panel disappears if the user clicks anywhere else on the screen.



(a) In its default state, WifiLearner shows a “Click to learn" message next to the top menu bar wifi icon.



(b) During wifi seeking, icon blinks green and yellow and the vocabulary prompt (“world”) is shown.

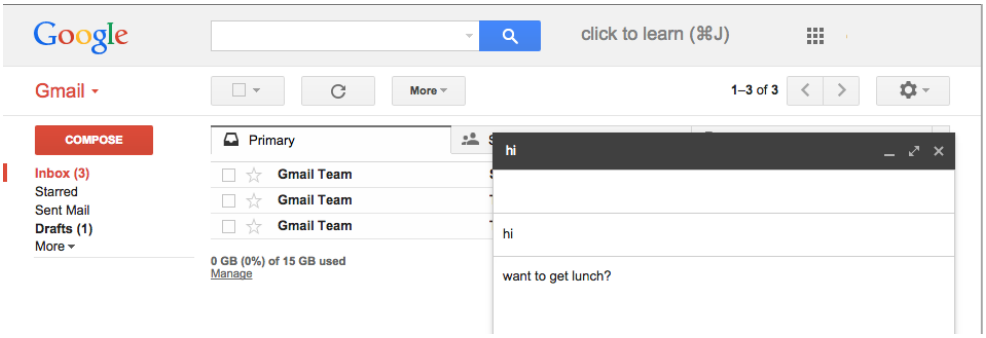


(c) After the user clicks, the learning panel appears where s/he can type in the translation.

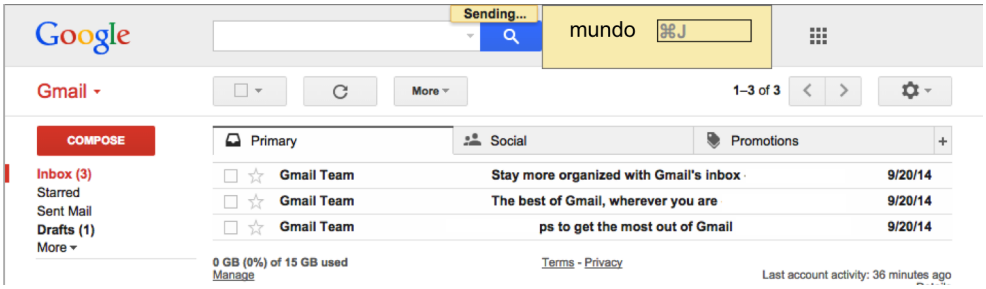
Figure 3-7: WifiLearner user interface.

3.2.7 EmailLearner

EmailLearner is a Gmail Chrome extension that appears when the user hits Send on an email. In many email clients, a status label indicates the state of the email being sent. In Gmail, it appears in the form of “Sending...” followed by “Your message has been sent,” which users may watch to make sure the email is sent. Thus, EmailLearner is placed in the dead space next to this status label. To minimize obtrusiveness, the learning panel displays a transparent background and the text “click to learn” in its default state (Figure 3-8a). Once the user sends an email, the panel displays the learning exercise (Figure 3-8b). If the user does not interact with the learning panel within 15 seconds, the panel returns to its default state of being transparent, displaying “click to learn” once again. Keyboard shortcuts were added to maximize efficiency.



(a) The user writes an email, then hits the send button or other keyboard shortcuts to send the email.

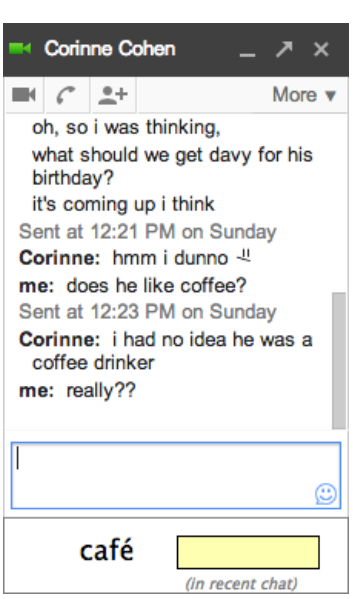


(b) As soon as the email is triggered to be sent, EmailLearner displays an exercise next to the “Sending..." label.

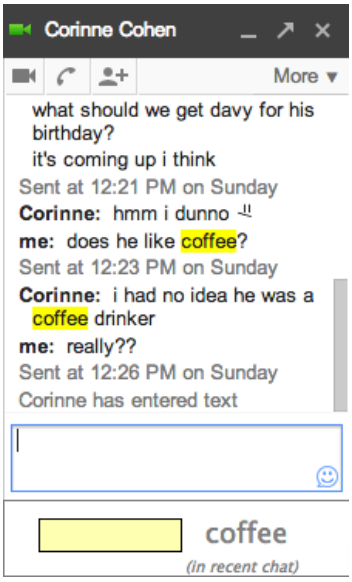
Figure 3-8: EmailLearner interface.

3.2.8 WaitChatter

We developed WaitChatter as a Chrome extension of Google Chat that runs in the Web browser when a Gmail page is in view.[2] Through early iterations, we found that placing the learning panel directly below the chat input box minimized the visual and motor effort of switching between chatting and learning activities. Like EmailLearner, WaitChatter displays "Click to learn" in its default state. When waiting is detected, the vocabulary exercise appears and remains in the learning panel for 15 seconds, during which the user can either do the exercise or ignore it by doing nothing. If the user has not interacted with the exercise within 15 seconds, it fades away. After the user completes an *initial* exercise, a *follow-up* one can be fetched by clicking the right arrow (Figure 3-4f) or hitting the Enter key on the keyboard. This functionality allows the user to continuously learn more words during longer wait times.



(a) At the easy level, L2 is displayed and the interface says "in recent chat" to indicate that it is a contextual word.



(b) At the difficult level, L1 is displayed and the corresponding words are highlighted in the chat history.

Figure 3-9: WaitChatter contextual interface.

Findings from prior research indicate that contextual and non-contextual vocabulary serve

[2] https://people.csail.mit.edu/ccai/waitchatter

complementary roles in learning, and that users may benefit from a combination of the two [52]. In WaitChatter, the IM conversation provides a ripe opportunity for in-context learning. The nouns in WaitChatter are either drawn from a built-in word list (*non-contextual*), or selected on-the-fly from words used by either conversant in the IM conversation (*contextual*). To indicate that a word is contextual to the conversation, WaitChatter displays “in recent chat" directly below the exercise (Figure 3-9a). To prevent user concern over whether the other person can view WaitChatter content, we keep learning exercises within the learning panel, and highlight a keyword inside the chat history if it is selected and presented for learning (Figure 3-9b). *Non-contextual* words could in theory be drawn from a number of sources, such as a word bank seeded by the learner or teacher.

For each chat message exchanged during an IM conversation, WaitChatter determines whether an adequate *contextual* word exists. First, it identifies nouns in the chat message using the Senna part-of-speech tagger [34]. Then, each noun is translated on-the-fly using Google Translate. To maximize the chances that the word is translated correctly in context, WaitChatter sends both the word and the entire chat message to Google Translate, and considers an L1/L2 pair *accepted* only if the L2 word appears in both translation results. To mitigate against inaccurate results, a dictionary icon (Figure 3-4f) is displayed which the user can click to see the word’s dictionary definition. For privacy reasons, WaitChatter logs only the length of a chat message and the L1/L2 pair displayed in an exercise, but not the content of the chat message itself.

Because exercises can only be displayed during appropriate wait-learning moments (described below), WaitChatter keeps a running set of *accepted* contextual L1/L2 pairs, for which the L1 word is still within the visible part of the chat history (*viewport*) but not yet displayed for learning. Once the word scrolls off the viewport, WaitChatter discards the L1/L2 pair from being considered for learning.

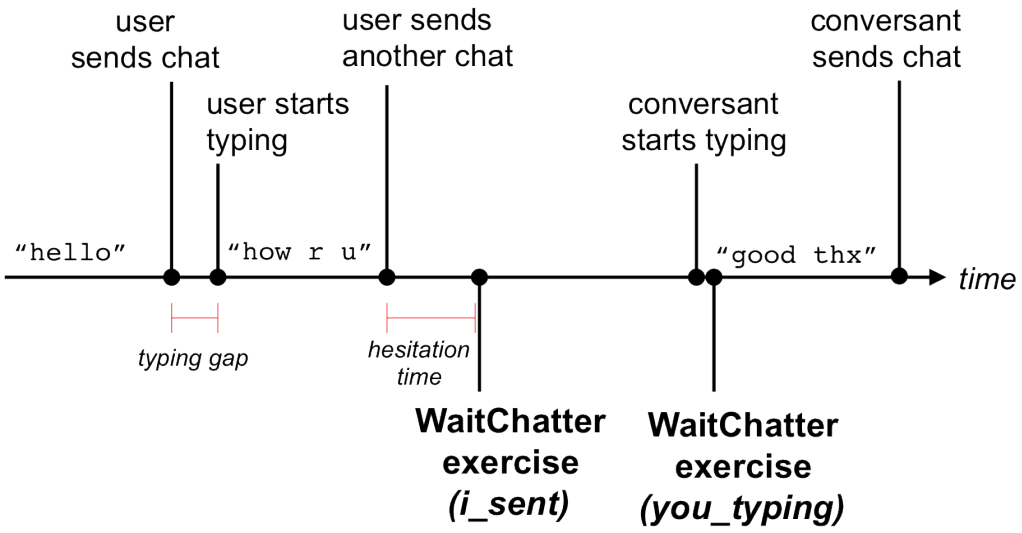


Figure 3-10: Detection of waiting opportunities in WaitChatter.

Detecting Waiting Moments

We identified two situations in which a user may be waiting during an instant messaging conversation: 1) while waiting for the conversant to start responding, and 2) while waiting for the conversant to finish responding. Figure 3-10 shows these two types of waiting opportunities in the flow of a typical conversation.

The first case (*i_sent*) occurs after a user has sent a chat message and is waiting to see whether the other person will respond. Because a common IM behavior is to type a sequence of short chat messages as part of one conversational turn [76, 96], an exercise that is naively delivered immediately after a chat is sent may interrupt a follow-up message that the user is in the midst of composing. For this reason, WaitChatter waits for 1.5 seconds of *hesitation time* after a message is sent, and subsequently triggers a learning exercise only if the user has not typed more. We keep the hesitation time short to balance against users leaving the chat window altogether. According to a prior study [10], the message window is substantially less likely to still be in focus the longer a user waits for a response.

The second case (*you_typing*) occurs when the conversant has started typing a response but has not yet sent the message. In instant messaging applications, users typically see an

indicator (e.g. “Corinne is typing...”) which signals that the conversant has started typing. WaitChatter triggers an exercise when the indicator appears in the user’s chat window and the user is not typing. In both *i_sent* and *you_typing* conditions, the exercise is only triggered if the cursor focus is inside the chatbox.

3.3 Multi-App System Implementation

WaitSuite is a common infrastructure that allows these apps to work together by synchronizing learning progress across apps. In this section, we describe the challenges we encountered while implementing multi-app functionality in WaitSuite and how we addressed these challenges.

3.3.1 Vocabulary Scheduling Algorithm

We use the Leitner schedule [61] for determining the order of learning exercises. The Leitner schedule is based on the principle of *spaced repetition* [11]. Given that humans exhibit a negatively exponential forgetting curve [50], repetitions should occur at increasingly spaced intervals so that they are reviewed again just as they are about to be forgotten. In WaitSuite, a flashcard is an L1/L2 vocabulary pair. WaitSuite maintains a set of five unlearned flashcards and a *correct count* for each flashcard, which represents the number of correct responses to that flashcard. This count is incremented when the learner answers the flashcard correctly and decremented if not. Flashcards with a correct count of n are displayed every nth Leitner session, so that better known cards are reviewed less frequently. In our implementation, flashcards are displayed at the easy level when the correct count is below three, and at the difficult level otherwise. When the correct count reaches four, a flashcard is considered learned, and retired, opening up a slot for a new card to be added. In Chapter 6, we adopt a more sophisticated version of this algorithm named MemReflex [53], which never retires a flashcard but rather continuously reduces the likelihood that it will be shown, even if it has been answered correctly multiple times.

3.3.2 Data Synchronization

So that users can continue to make learning progress on the same set of vocabulary when switching from one app to another, we used Firebase[3] to store and synchronize user data across the five platforms and devices. For apps like WifiLearner which specifically target internet delays, it was necessary to support operations when network was not available. Firebase caches data on the client side so that the user can continue to complete flashcard exercises while offline. Data is automatically synchronized once an app regains connectivity.

3.3.3 Resolving Staleness and Cross-App Conflicts

In some cases, a concurrency conflict may occur due to progress being made in one or more apps while offline. To handle such situations, we push updates as an atomic transaction using Firebase. In the case of a conflict, the system selects the value reflecting the furthest progress, which we define as the number of exercises completed. During pilot testing, we found that WifiLearner's exercises tended to be consistently stale because it usually fetched an exercise only when wifi was disconnected. We thus modified WifiLearner to force-synchronize to the server every time the internet connects. Because wifi tends to connect and disconnect frequently on laptops, we found that even when the device was not in use, this synchronization kept data reasonably fresh.

[3] https://www.firebase.com/

Chapter 4

Feasibility Study

Because wait time tends to be short and fleeting, and wait-learning requires temporarily switching away from and back to an ongoing activity, it is unknown whether it is feasible to be productive during these fleeting moments. In the case of learning, it is also unclear whether the knowledge gained while waiting can be retained over time. To empirically validate the feasibility of wait-learning, we first created WaitChatter. Using WaitChatter, we ran a two-week field study in which Google Chat users installed WaitChatter as a Chrome extension on their personal computers, and were allowed to use it during their instant messaging activities to learn foreign language vocabulary.

The questions our study sought to answer were:

1. Learning: To what extent can people learn vocabulary using WaitChatter?
2. Timing: What is the best time to present learning exercises?

Overall, we found that people are able to learn vocabulary during wait time, and that wait-learning is more effective when triggered at the *start* of waiting periods, compared to at random times or in the middle of waiting.

4.1 Vocabulary

For ease of user recruitment, our implementation of WaitChatter teaches Spanish and French, but could easily be extended to other languages. The vocabulary was drawn from high frequency English nouns as measured in the British National Corpus.[1] The words were translated to Spanish and French using Google Translate. Two native speakers of Spanish and French who were bilingual in English manually reviewed the word list for inaccurate translations, and removed highly ambiguous words. The final word lists consisted of 445 words in each language (Appendix A). In addition to these word lists, contextual words were also automatically extracted from the conversation and translated on-the-fly, as described in Section 3.2.8.

4.2 Procedure

Each participant used WaitChatter for two weeks, after which they completed a post-study questionnaire (Appendix A), interview, and vocabulary quiz. Participants were asked to interact with WaitChatter exercises as little or as much as they pleased. During the study, WaitChatter prompted participants to indicate whether or not they already knew a word the first time it appeared, and only unknown words were repeated for learning. The post-study quiz tested all vocabulary the user indicated they did not already know. Participants first translated from L1 to L2 in a recall quiz, then from L2 to L1 in a recognition quiz.

4.2.1 Timing Conditions

Different moments during a waiting period involve different levels of cognitive resource expenditure. To better understand how the timing of exercises may affect the learner's capacity to engage in learning, we exposed each participant to two versions of our application. The

[1] http://www.natcorp.ox.ac.uk/

detected_wait version uses the *i_sent* and *you_typing* waiting opportunities as described in the previous chapter. The *random* version displays prompts at random whenever WaitChatter determines that a user is actively instant messaging. We define a user's instant messaging activity as active if the Gmail page is in focus, has had keyboard or mouse activity within the last 30 seconds, and contains at least one open chat window which had keyboard activity within the last 5 minutes. Because a user may be waiting for an instant messaging response while engaged in nearby tasks on the same page, we consider moments when the Gmail page is in focus even though the chatbox is not as viable candidates for wait-learning, so long as the user is active as defined above. We also require at least one chatbox to be open because an open chatbox indicates the likely presence of an ongoing conversation [10].

We expect that at moments when attentional capacity is high, the user's likelihood of engaging with the exercise (*engagement rate*) will be higher, and the time taken to initiate an exercise (*response time*) will be lower, due to lower switch cost. We measured engagement rate as the percentage of exercises that the learner responded to and response time as the time between a prompt being displayed and the user's cursor entering the answer box. Each participant used the *detected_wait* and *random* versions on alternating days. To ensure that users were exposed to WaitChatter prompts at approximately equal frequencies on the *detected_wait* and *random* versions, the desired frequency on a *random* condition day was determined by calculating the total number of exercises shown on all previous *detected_wait* days, divided by the total seconds of user chat activity on those days. This gives us the probability of showing an exercise in a given second on a *random* condition day. To capture subjective impressions, users were asked to complete a daily survey (Appendix A) with two 7-point Likert scale questions: 1) "In the past day, [WaitChatter] exercises appeared at good moments within the flow of my daily activities" and 2) "I enjoyed using [WaitChatter] today." The survey was sent via email every evening, instructing users to complete it once they finish chatting at the end of the day.

4.3 Participants

21 participants were recruited by emails sent through university department, dorm, and foreign language course email lists. We selected only those who were regular users of Google Chat in the web browser, and desired to learn or were currently learning Spanish or French. One participant was dropped midway through the study because that participant stopped instant messaging and completing the daily surveys after the sixth day. Participants were given a $30 gift card for their time and were also entered into a raffle for one $100 gift card. The 20 participants who completed the study included 12 males and 8 females, ages 19 to 35 ($\mu = 25.5$). Most participants were undergraduate and graduate students (17 out of 20), as well as two alumni working in industry, and one research scientist. Participants chose to learn French (11) or Spanish (9). Ten users had prior formal education in the language, including elementary school (2), middle or high school (6), and university-level (2) classes. Eight of the participants had studied the language informally through language learning software, travel in a foreign country, or conversation with friends. Six participants had never studied the language before, either formally or informally. The participants typically use Google Chat on their computers "Several times an hour" (9) or "Several times a day" (11), mostly for social reasons or to chat casually with coworkers.

4.4 Results and Lessons Learned

Overall, we observed 47,393 instant messages exchanged by the 20 participants, who communicated with a total of 249 friends. Each participant exchanged an average of 170 chats per day.

4.4.1 Evidence of Learning

During the study, WaitChatter prompted participants to indicate whether or not they already knew a word the first time it appeared. Known words were not added to the user's vocabulary

list, nor were they quizzed. Participants were on average exposed to 87.7 new words (sd=64.8) that they didn't already know. In post-study quizzes, users translated 57.1 words (66%) correctly to L2 and 80.2 words (92%) correctly to L1. In quiz translations to L2, 15% of wrong answers appeared to be spelling errors or near-misses. The user who was exposed to the most words (256) translated 161 correctly to L2 and 232 correctly to L1. The most infrequent chatter (55 chats per day) learned 17 new words. Thus, in two weeks, participants learned approximately four new words per day, or 57 words over two weeks. Overall, these results suggest that wait-learning can serve as a viable channel for learning, at least for bite-sized information.

4.4.2 Evidence of Learning While Waiting

In post-study interviews, users reported behavior that resembled episodes in which they were learning while waiting. For example, users said they tended to complete exercises "while waiting for people to respond," or "while the other person is thinking."

Users indicated that they were most likely to interact with WaitChatter during a *continuous but casual* IM conversation. They were least likely to engage with exercises if they were having a particularly time-sensitive conversation or if the nature of the conversation was serious or work-related. As one user put it, *"The best times are when I'm talking continuously with one person, but we're not having a very heated conversation. Just like hi, how are you, and when the material is more light."* Thus, WaitChatter usage seemed to occur during periods of "outeraction" [111], when people communicate for the purpose of maintaining social connection and awareness, rather than for specific information exchange.

High-usage participants said that they frequently used the "fetch more" feature (Figure 3-4f) to do a long sequence of exercises if the conversation was particularly sporadic: *"At some points I needed to wait for the other person to respond. The longer they take, the more words I would go through."*

To understand the extent to which exercises could be feasibly completed during wait time,

we measured the *intermessage time*: the amount of time after the user sent a message before receiving a reply (Figure 4-1). We found that the time between the user sending a message and receiving a message from the conversant (Figure 4-1) exhibits a distribution with a peak and a long tail, similar to responsiveness distributions reported in a prior IM study [10]. Some intermessage times are short because the conversant had started composing a message before the user sent a message. Results show that the time taken to complete an exercise was short (median 1.83 seconds) and within the intermessage time (median 11 seconds). However, because intermessage time is short (mode=4 seconds), particularly during conversations with frequent exchanges, it is important that the exercise be lightweight, as described in Section 3.1.2.

4.4.3 Waiting Behavior

Feedback from users supported existing motivational theories about task switching as a means of coping with boredom while waiting. Users described existing habits of interleaving instant messaging with compulsive technology use, such as browsing social media or checking email. When well-timed, wait-learning served as a more productive replacement for other digital activities. As one user put it, *"Maybe I'm just chatting and looking at Facebook. Instead I would use WaitChatter because it's more productive."* Given existing tendencies to task switch, the timing of a learning task is critical not only for identifying moments of low cognitive load, but also for capturing the user's peripheral attention before they switch to an alternative secondary task.

Given the academic nature of flashcards, we were surprised that multiple users likened WaitChatter to mini-games they could "play with" in spare time. For example, users said *"It's like playing any of those low-stakes games"*, *"I just wanted to play with it because whatever was on TV was really boring,"* or *"I was just bored and I wanted to play."* Because exercises were quick to do and also optional, users found wait-learning to be casual and low commitment. Some users also found the contextual vocabulary surprising when shown in context of their conversation, which may have made the system feel more playful. The

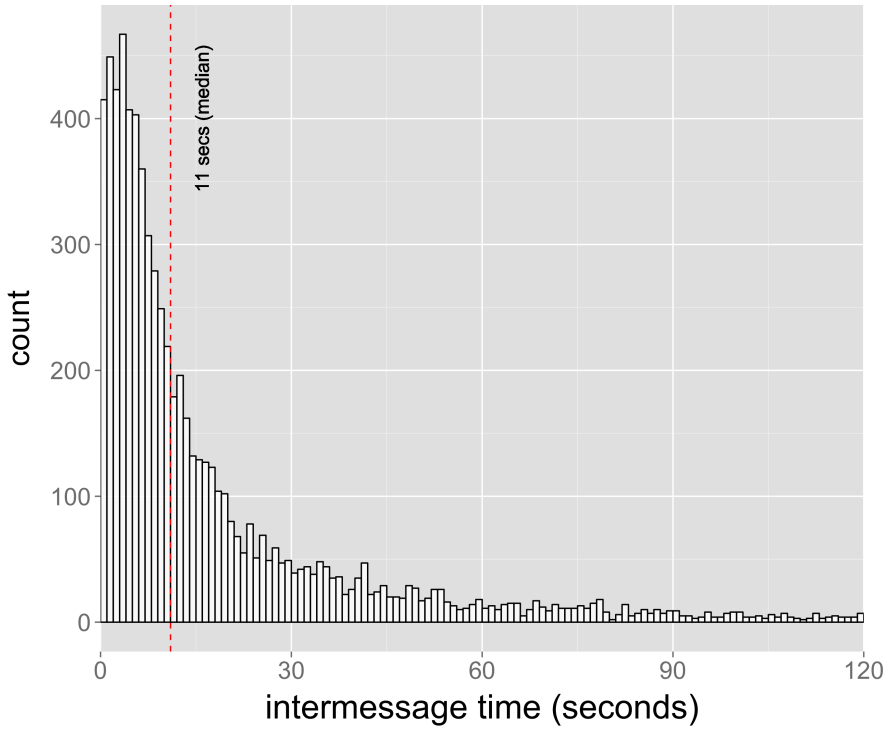


Figure 4-1: Histogram of intermessage time: the time between the user sending a message and receiving a message from the conversant. Bin size is 1 second.

potential for wait-learning to entertain, stimulate, or amuse is a direction worth exploring, particularly given the well-known negative experience associated with waiting [98].

4.4.4 Overcoming Limited Time

As shown in Figure 4-2, responses to 7-point Likert scale questions indicated that users found WaitChatter very enjoyable ($\mu = 6.15$). They also felt that they would continue using WaitChatter if they could ($\mu = 6.15$), and would engage in vocabulary practice more frequently than they would otherwise ($\mu = 6.6$). On the last question, 15 out of 20 participants submitted a rating of 7.

During interviews, the most commonly cited benefit of WaitChatter was that it felt less time-consuming compared to existing learning channels, because it did not require them tos set aside time for learning. As one user stated, *"The key thing is that I didn't feel like I was*

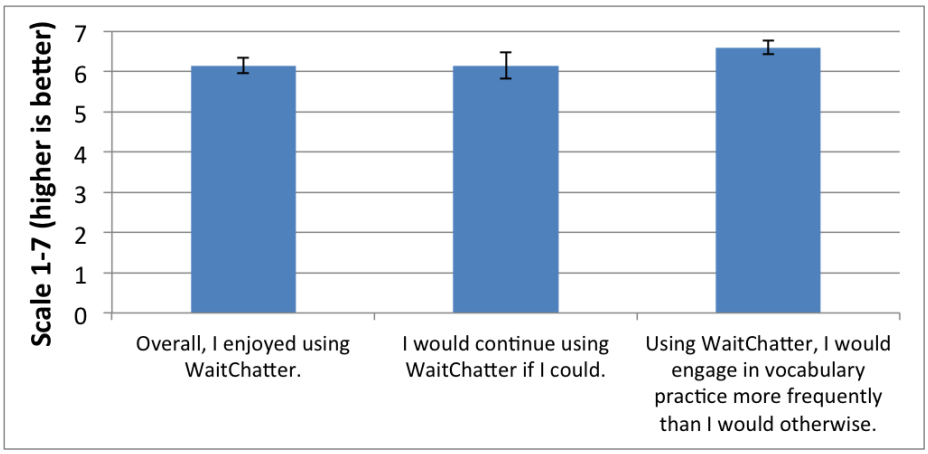


Figure 4-2: Participant responses to the post-study Likert scale survey. On average, users found WaitChatter enjoyable ($\mu = 6.15$), felt that they would continue using WaitChatter if they could ($\mu = 6.15$), and felt that they would engage in vocabulary practice more frequently than they would otherwise ($\mu = 6.6$).

taking extra time out of my day to dedicate to learning vocabulary words. It was just sort of time that would be wasted otherwise." Another compared it to typical break-time activities: "*Some people play Angry Birds, but for me, I would play with [WaitChatter]. At least I'm learning some French words.*"

Many contrasted WaitChatter to language courses and software, which they felt required a conscious effort to schedule time for learning. One person commented, "*With Duolingo you have to think 'I have to go do this now', whereas with [WaitChatter] it's already done for you, spoonfed to you.*" Another said, "*With this I never had to make time or put away things to the future. Whereas learning from Rosetta Stone, you have to schedule time.*" Most who had used vocabulary-learning mobile applications indicated that they eventually gave up, citing time as a major factor.

Several users noted that the little time required to complete a WaitChatter exercise ironically encouraged them to interact more with it overall. One person described the low time commitment as follows: "*It's just like, here, literally just take 2 seconds!*" Another appreciated the regularity of exposure: "*You're just constantly getting new words. It might be a slower rate overall [compared to classes], but it's neat in the aspect that you're always chugging along, learning new vocabulary.*" These comments suggest that user engagement

Condition	Day Shown	Condition Requirement
i_sent	detected_wait (odd days)	1.5 seconds after user sends a chat, provided he has not started typing again
you_typing	detected_wait (odd days)	"[conversant name] is typing..." indicator appears and user is not typing
random_inside	random (even days)	Chat window is in focus
random_outside	random (even days)	Chat window is not in focus

Table 4.1: Summary of conditions.

could hinge more on the perceived than the actual time spent.

4.4.5 Evidence of Sensitivity to Timing

To understand how the user's capacity to wait-learn could be affected by timing, we evaluated the engagement rate (whether the user responded to the exercise) and response time (the time taken before the cursor focused inside the exercise) on exercises within the three timing conditions described above: *i_sent*, *you_typing*, and *random*. For each user, data on the first *detected_wait* condition day and the first *random* condition day were excluded from analysis to avoid novelty effects. Furthermore, because follow-up exercises are requested by the user whereas initial exercises are not, we focus our analysis only on initial quiz exercises.

Because *i_sent* and *you_typing* exercises occurred only while the user had cursor focus inside the chatbox, we subdivided exercises on the *random* condition days into *random_inside*, when the chatbox had focus, and *random_outside*, when the chatbox did not have focus. We compared *i_sent* and *you_typing* only to *random_inside* trials. In all cases, the user had been actively instant messaging as defined in the Timing Variation section above. Table 4.1 displays a summary of conditions.

Engagement Rate

First, we found that 43.5% of the exercises received a response in the *random_inside* condition, whereas only 31.2% received a response in the *random_outside* condition. A generalized linear mixed effects analysis with the Timing condition (*random_inside* and *random_outside*) as the fixed effect and Participant as a random effect found that *ran-*

dom_outside deliveries were significantly less likely to receive a response ($p < 0.0001$). This analysis excludes one participant who did not receive any *random_outside* exercises. Consistent with a prior study [10] which found that chat window focus may be a strong indicator for chat responsiveness, our results imply that this applies to learning interactions as well.

Comparing *i_sent*, *you_typing*, and *random_inside*, we found that engagement rate was highest for *i_sent* (49.1%), followed by *random_inside* (44.5%) and *you_typing* (41.2%). In a logistic mixed effects analysis, p-values were 0.0085 (*i_sent*>*you_typing*), 0.0498 (*i_sent*>*random_inside*, and 0.397 (*you_typing*=*random_inside*), of which only the first passed the Bonferroni-corrected threshold of 0.0167 (*i_sent*>*you_typing*). Thus, users were significantly more likely to respond when exercises appeared just after the user sent a chat message (*i_sent*) relative to the *you_typing* condition, with an odds ratio of 1.4. One reason why the *random_inside* condition faired reasonably well could be that, during instant messaging, the difference in time between waiting and not waiting is slim, due to the brevity of chat messages. For instance, a flashcard that appears while the user is typing can still be completed once the user has finished typing, if the chat message is short. We believe the difference in engagement would be greater in scenarios where there is a greater time difference between waiting and non-waiting.

The relatively low response rate in the *you_typing* condition could be due to a number of factors. In contrast to the *i_sent* condition, in which the user is likely still watching the chatbox right after sending a message, *you_typing* occurs further into the waiting period, when users may already be visually focused elsewhere even if their cursor is focused inside the chatbox. In cases where the user is looking at another screen or away from their computer, the learning task may be too far away to enter the periphery of attention. Or, if they have simply shifted focus to another part of the page (e.g. gmail), it would be too costly to transition the learning task from the periphery to the center of attention, a process that is critical for facilitating peripheral interaction [15]. Moreover, in the *you_typing* condition, it is also possible that seeing the typing indicator makes users believe they are about to receive a response, particularly given fast typing speeds in desktop environments and the fact that

chat messages tend to be short [96]. Thus, the expected wait time becomes so short that it no longer justifies the switch cost. Users may already be receiving a response by the time they are able to attend to the exercise.

Response Time

Response time was shorter in the *random_inside* condition ($\mu = 3973$ ms, $\sigma = 792$) than the *random_outside* condition ($\mu = 4888$, $\sigma = 1282$). A paired t-test found this difference to be statistically significant (F(1,16)=13.24, $p < 0.005$). This analysis excludes three participants who did not respond to any *random_outside* exercises. The longer time taken to switch in the *random_outside* condition could be due to the mental cost of switching from another activity, or the physical distance traveled to reach the answer box.

Comparing *i_sent*, *you_typing*, and *random_inside* (Figure 4-3), we found that *i_sent* had the fastest response time ($\mu = 3628$, $\sigma = 637$), followed by *random_inside* ($\mu = 4009$, $\sigma = 792$), and *you_typing* ($\mu = 4209$, $\sigma = 969$). A repeated measures ANOVA found a significant effect of condition (F(2,38) = 4.00, $p < 0.05$). Post-hoc Bonferroni tests revealed that learners were significantly faster to switch to learning in the *i_sent* condition than in the *random_inside* condition ($p < 0.05$). Users were also quicker to switch in the *i_sent* condition than *you_typing*, a difference that was marginally significant ($p = 0.05$). These small but significant differences can have large cumulative effects over the course of hundreds of chat messages and conversations.

These results suggest that the best time to present learning exercises may be at the *start* of a waiting period, when mental workload may be lower because the user has just completed one subtask (sending a chat) before beginning the next [103]. Users were slower to respond to randomly timed exercises and *you_typing* exercises. It is possible that, during those times, users are already in the midst of planning their next message, or concentrating on what their friend will say in their response. The mental resources available may be small due to an intermediate state being actively maintained in short-term memory, which is characteristic of low-level task boundaries or non-boundaries [13].

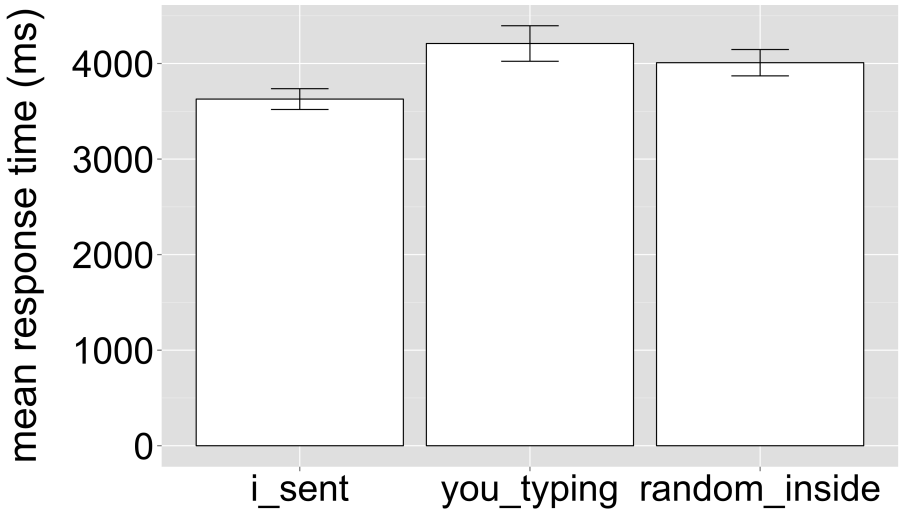


Figure 4-3: Mean response times for *i_sent* ($\mu = 3628$ ms), *you_typing* ($\mu = 4209$ ms), and *random* ($\mu = 4009$ ms). Error bars show SE of the mean.

During the study, it appeared that in the *random_inside* condition, many users responded quickly to an exercise even if it appeared while they were typing. We thus examined response times depending on the number of keystrokes remaining in the message that the user was in the midst of composing (Figure 4-4). We split these instances into two equally sized groups, and found that the response time for flashcards arriving just before the user sent a chat (within the last 8 keystrokes) is lower (median = 2216 ms) than those arriving earlier (median = 3446 ms). This result is interesting as it suggests that microtasks arriving immediately prior to message completion may receive a prompt response, possibly because the user is almost finished typing and can thus anticipate that waiting is about to start.

Hesitation Time

Lastly, we evaluated the 1.5 second hesitation time that WaitChatter used to ensure the user was not typing a followup message before it delivered a learning exercise in the *i_sent* condition. Due to missing log data for some users, we report preliminarily on the seven users for whom we had a full set of millisecond-accurate data, including all times they pressed the enter key. We limit our analysis to events where the user re-started typing within 30 seconds, based on prior research findings that 28-30 seconds is a typical amount

of time between conversational turns [8]. Among these instances, 70% occurred within WaitChatter's hesitation threshold of 1.5 seconds, making it a reasonable estimate (Figure 4-5). Nevertheless, the hesitation threshold ought to be balanced against the user tendency to leave the chat window after sending a message. A more lenient implementation of WaitChatter might set the hesitation threshold to be even lower than 1.5 seconds.

4.4.6 Importance of Non-intrusiveness

Despite the differences we found between timing conditions, users indicated during interviews that they did not notice systematic differences in the timing of exercises. In the 7-point Likert scale questions sent daily (1=strongly disagree, 7=strongly agree), users on average felt that the exercises "appeared at good moments within the flow of my daily activities" ($\mu = 5.45$, $\sigma = 1.05$) and that they "enjoyed using WaitChatter today" ($\mu = 5.61$, $\sigma = 1.02$). A Wilcoxon signed-rank test found no significant difference between user ratings

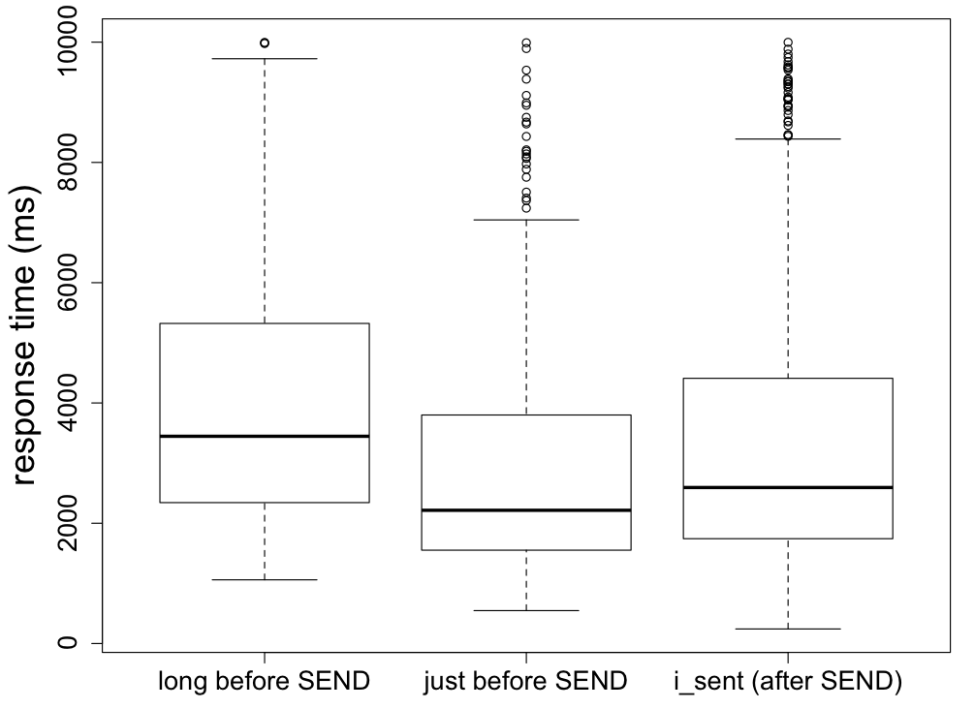


Figure 4-4: Box plots of response time for exercises arriving just before a chat is sent (≤ 8 keystrokes left), compared to longer before a chat is sent (> 8 keystrokes left).

on *detected_wait* versus *random* condition days, for either question ($p = 0.71$ and $p = 0.57$). In the post-study questionnaire, users also indicated that the frequency of learning exercises felt neither too high nor too low ($\mu = 4$, $\sigma = 1.2$) on the questionnaire item "Please rate the frequency of [WaitChatter] exercises", where 1 was too infrequent and 7 was too frequent (Appendix A).

We posit that the use of soft notifications (static learning panel, dynamic text) was key to minimizing intrusiveness when flashcards appeared at non-wait times. During interviews, users indicated that, because the exercise did not occlude any other tasks they were doing and did not require any extra action to dismiss, they could easily ignore the exercises without feeling interrupted. For example, one participant said, *"It was just a matter of choice. Subconsciously I would see that a word has appeared. Sometimes it would pique my interest and I would look at it, but if not I just wouldn't look at it so it wasn't really disrupting anything."*

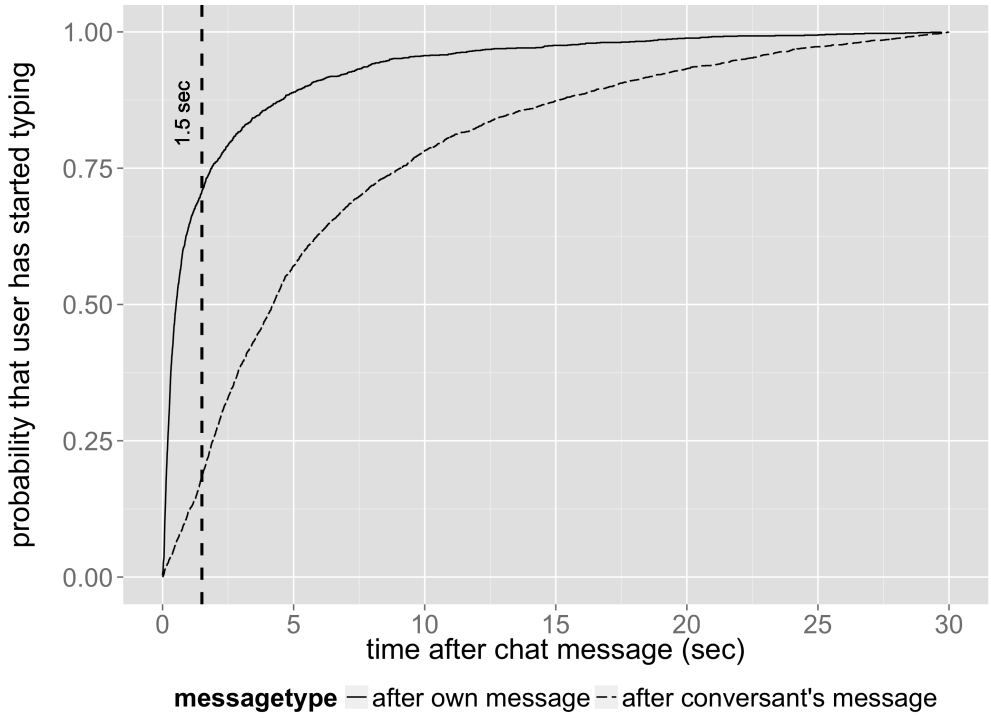


Figure 4-5: If the user has started typing within 30 seconds of sending a chat, there is a 70% chance that this re-typing occurs within 1.5 seconds. In contrast, if the user starts typing within 30 seconds after *receiving* a chat, there is only a 18% that it occurs within 1.5 seconds.

These findings are consistent with prior research showing that interruptions which do not occlude the primary task are perceived to be less mentally demanding and less annoying [23]. Unlike hard notifications, which often appear on top of existing tasks and immediately draw attention, the learning panel was in a persistent self-allocated space, making it potentially less distracting even in cases when timing was sub-optimal.

However, some users reported feeling frustrated when they could not attend to the exercise in time because they were still typing a long message. Hence, while users did not perceive the appearance of an exercise to be intrusive, they may be less tolerant of the premature disappearance of an exercise. As a result, some wished WaitChatter had a feature for self-triggering an exercise at any time.

4.4.7 Contextual Relevance

Because the focus of this study was timing, and because there is already empirical evidence of the benefits of in-context learning [93], we did not explicitly manipulate contextual vs. non-contextual vocabulary as separate experimental conditions. However, we collected user feedback on this feature in the post-study questionnaire (Appendix A) and interviews. Comparing user satisfaction with non-contextual and contextual vocabulary, we did not find substantial differences in either the quality of the words ($\mu = 4.9$ for non-contextual, $\mu = 4.9$ for contextual) nor variety of words ($\mu = 5$ for non-contextual, $\mu = 5.1$ for contextual). During interviews, users indicated that they were generally happy to see vocabulary words that were relevant to their conversation, with many finding these instances delightful. Some found blatantly incorrect translations amusing and memorable (e.g. “totes” was translated as a bathroom tote but meant “totally”), while others wished they could veto cases when the system made incorrect translations or selected words they did not wish to learn (e.g. acronyms). Beginning learners also tended to prefer seeing basic vocabulary over seeing contextual words, particularly when the contextual words were difficult.

4.5 Conclusion

Results from this study show that learning is indeed feasible during wait time, and that a good time to present learning opportunities is at the start of the waiting period. While this work investigated the effectiveness of wait-learning within a single waiting scenario (instant messaging), the general classes of measures that were investigated – the amount learned during wait time and the timing of learning opportunities – occur in other situations and apply to other potential forms of wait-learning. In the next chapter, we demonstrate how wait-learning can be expanded to a variety of waiting scenarios by evaluating the usage of multiple wait-learning systems in the wild.

Chapter 5

Multi-System Deployment

In this chapter, we expand our analysis to multiple waiting scenarios by deploying all five apps of WaitSuite, and evaluating each one in the context of the design dimensions we have described. In the deployment, participants used multiple wait-learning apps on their personal devices during their regular activities for a period of two weeks. Since a core goal of wait-learning is to allow busy people to regularly engage in learning practice, we focus on engagement rate as a key metric in our evaluation. To keep exercises consistent across apps, all five apps used second language vocabulary exercises.

We explore the following research questions:

1. RQ1: To what extent do users engage with learning during different waiting situations, and which factors are instrumental to engagement?
2. RQ2: How does wait-learning impact perception of mental workload?

We found that wait time, ease of accessing the learning exercise, and competing demands were key factors affecting engagement with learning. In a supplementary analysis, we also discovered that most participants regularly used more than one wait-learning app, supporting a need for wait-learning across diverse waiting contexts. We end with design implications and a theoretical framework for wait-learning that extends existing work on attention management. The framework illustrates a combination of constraints (wait time,

ease of access, and competing demands) that is more effective for wait-learning. This work provides insight into how future wait-learning systems can be designed to enhance learning engagement during wait time while minimizing disruption to ongoing tasks.

5.1 Procedure

To answer these questions, we deployed WaitSuite on the participants' own devices for two weeks. Each participant met with a researcher to install the apps and complete a pre-study questionnaire. In order to capture natural usage, we told participants to use the apps as little or as much as they pleased. During the study, interaction with the apps was logged extensively, including metrics such as wait time and completion of learning exercises. Participants returned after two weeks to complete a post-study questionnaire, semi-structured interview, and vocabulary quiz. The quiz format and languages studied were the same as that of the Feasibility Study. As before, the words were translated from high frequency English nouns taken from the British National Corpus. There were 445 words in each language.

To evaluate the extent to which wait-learning affected the perceived mental workload of existing activities (RQ2), e.g. riding the elevator, we included NASA TLX questions in the pre-study and post-study questionnaires before and after the learning apps were used (Appendix B). The NASA TLX questions were measured on a 7-point scale. These questions have been used to measure effects of secondary tasks in prior studies [73].

5.2 User Interface Modifications

Based on timing results from the Feasibility Study, we modified WaitChatter to automatically trigger exercises only after the user sends a chat message, the *i_sent* condition from the Feasibility Study. To be consistent with the other four apps, we also turned off WaitChatter's contextual feature which detects words within conversations.

Because users in the Feasibility Study expressed that wait-learning helped them identify other moments when they wished they could trigger additional exercises on their own, we added a feature in WaitSuite for users to self-trigger exercises at any time. In addition to *system-triggered* exercises that are presented automatically by the system, each interface also allows the user to *self-trigger* exercises. Depending on the app, users can fetch an exercise by opening the app (ElevatorLearner), pulling again (PullLearner), or interacting with the learning panel (WifiLearner, EmailLearner, WaitChatter).

5.3 Participants

27 participants were recruited through university email lists. Users were selected based on the platforms they already used so that we could examine natural usage. Early on, it became clear that it was unlikely for any one person to encounter all five waiting scenarios regularly, due to existing habits. Thus, we selected only users who indicated they were regular users of at least three of the WaitSuite app platforms: Android (PullLearner), Mac (WifiLearner), Gmail (EmailLearner), and GChat (WaitChatter). For ElevatorLearner, this meant owning an iPhone and working in the building where iBeacons were located. Two participants were dropped from the study at the time of installation because we discovered they met fewer than three of the requirements.

The 25 participants (11 female) who completed the study were ages 19 to 47 ($\mu = 25.8$), consisting of students and staff at a university. Sixteen chose to learn French and nine chose to learn Spanish. Eleven had prior formal education in the language. Thirteen had studied the language informally through language learning software, traveling in a foreign country, or talking to friends in the language. Eight participants had never studied the language before. Participants were given a $40 gift card for their time.

5.4 Data Analysis

Before analyzing interaction data, we excluded the first day of the study to discount novelty effects, and times when a user reported technical difficulty. We also excluded PullLearner data for 3 users who stopped using PullLearner due to email formatting problems on the K9 email client, which prevented rich text from displaying properly. Lastly, because one user's pre-study questionnaire data was lost, we only report questionnaire results from 24 of the 25 participants.

Tables 5.1 and 5.2 present a summary of results across apps. The metrics shown were computed using data from event logs, as follows:

- *% Days present*: The percent of days that the user was present on the app platform. For example, on any given day, a user can only encounter EmailLearner and WaitChatter system-triggers if they happen to be using Gmail in their web browser.
- *System-triggers/day*: The number of system-triggers per day. A system-trigger occurs when the app triggers an exercise due to automatically detected waiting.
- *% Engaged*: The percentage of system-triggered exercises that the user responded to.
- *System-triggered submissions/day*: The number of exercises submitted per day that were either system-triggered, or within the chain of exercises fetched by the user following a system-triggered exercise. Note that this value can be greater than system-triggers/day, because a single system-trigger can result in the user responding and completing multiple exercises.
- *Wait time*: The waiting duration. Depending on the app, this refers to the time between pulling and email loading (PullLearner), the time taken for wifi to connect (WifiLearner), the time between a user hitting Send and the email being sent (EmailLearner), and the time between the user sending a chat and receiving a chat from the same friend (WaitChatter). For ElevatorLearner, wait time was estimated using the time between iBeacons on two different floors triggering, an upper bound which may include the elevator ride time.
- *Response time*: The time taken for a user to start interacting with an exercise, after the

App (# users)	% Days present	System-triggers/day	% Engaged	System-triggered submissions/day
ElevatorLearner (12)	60 (11)	1.5 (1.2)	46.7 (21.3)	5.8 (5.6)
PullLearner (8)	82 (29)	5.9 (5.6)	62.5 (12.1)	4.8 (4.8)
WifiLearner (19)	90 (14)	4.2 (2.5)	17.5 (14.2)	2.8 (3.1)
EmailLearner (23)	89 (15)	4.1 (4.3)	39.8 (31)	5.9 (8.1)
WaitChatter (16)	61 (28)	20.7 (34.8)	13.4 (11)	6.6 (10.9)

Table 5.1: Summary of results across the 5 WaitSuite apps. Exercise submissions (system-triggered submissions/day) depended on both the frequency of waiting moments (system-triggers/day) and the user's engagement rate (% engaged). The highest engagement rate was observed on PullLearner and ElevatorLearner, and the lowest on WaitChatter. However, the greatest number of system-triggered exercises were completed on WaitChatter, and the lowest on WifiLearner. Unless stated otherwise, numbers report the mean and standard deviation across users.

exercise appears.

5.5 Results: Engagement with System-Triggered Exercises (RQ1)

5.5.1 Overview

The rate at which exercises are completed depends on both the frequency of waiting opportunities (frequency of system-triggered exercises), and the likelihood that the user engages with an exercise given that one is triggered (engagement rate). We first computed the engagement rate on each app, measured as the percentage of system-triggered exercises that the user responded to. We found that engagement rates varied substantially between apps, with the highest engagement on PullLearner (62.5%, $\sigma = 12.1\%$), followed by ElevatorLearner (46.7%, $\sigma = 21.3\%$), EmailLearner (39.8%, $\sigma = 31\%$), WifiLearner (17.5%, $\sigma = 14.2\%$), and WaitChatter (13.4%, $\sigma = 11\%$). A generalized linear mixed effect analysis with the App as the fixed effect and the Participant as a random effect found a significant effect of app on engagement rate. Post-hoc analysis found that engagement was significantly different between all apps ($p < 0.001$), with the exception of PullLearner and ElevatorLearner whose difference was not significant ($p = 0.997$).

Next, we observed the total number of exercises completed per day that were part of any system-triggered exercise chain. A system-triggered chain consists of an initial exercise triggered by the system, as well as followup exercises that the user might have fetched after completing a system-triggered exercise. The greatest number of exercises were submitted on WaitChatter (6.6 per day), and the lowest on WifiLearner (2.8 per day). Despite a relatively low engagement rate, many exercises were submitted on WaitChatter due to a high frequency of system-triggers (20.7), resulting from the frequent back and forth waiting that occurs while chatting. Conversely, ElevatorLearner was system-triggered only 1.5 times per day on average. Users sometimes took the stairs instead of the elevator, and were often not at work on weekends, which limited their exposure to the specific elevators where we had placed bluetooth iBeacons. Nevertheless, the number of exercises completed on ElevatorLearner was still reasonably high (5.8), due to a high engagement rate (46.7%). Lastly, WifiLearner triggered a reasonable number of exercises (4.8 times per day) due in part to regular computer usage (observed on 90% of days). However, it had the fewest submissions (2.8 per day) because engagement rate was low (17.5%).

These engagement results can be explained with respect to the switch cost of wait-learning in different situations. Aside from the design dimensions already enumerated (e.g. wait time, competing demands), we found that the physical ease of accessing an exercise was an important factor that contributed to switch cost, but was not explicitly enumerated in our design space. Although we had designed each app to maximize *ease of access*, the inherent constraints of different waiting contexts meant that the effort required to access an exercise naturally varied between apps. For example, pulling out and unlocking a phone (ElevatorLearner) took much longer than hitting Tab to focus into a textbox (WaitChatter).

The ease of accessing an exercise relative to the wait time had an important impact on user engagement. We define this relationship between ease of access and wait time to be the *access-to-wait ratio*. If the access-to-wait ratio is low, there is sufficient time for the user to switch to the learning task and back, so the value gained in learning and the emotional fulfillment gained in occupying wait time justify the switch cost. However, if the access-to-wait ratio is high, the time taken to switch exceeds the waiting period, so the motivation

App (# users)	Wait time	Response time	% Engaged
ElevatorLearner (12)	med=52.9 sec (21.3)	med=10.0 sec (11.3)	46.7 (21.3)
PullLearner (8)	med=2.3 sec (1.9)	med=1.6 sec (0.2)	62.5 (12.1)
WifiLearner (19)	med=6.8 sec (3.1)	med=7.3 sec (3.3)	17.5 (14.2)
EmailLearner (23)	med=0.7 sec (0.5)	med=2.7 sec (1.6)	39.8 (31)
WaitChatter (16)	med=10.2 sec (13.1)	med=2.9 sec (2.7)	13.4 (11)

Table 5.2: For each app, the engagement rate (% engaged) was related to the ease of accessing the exercise (estimated using response time), the waiting duration (wait time), and the likelihood of competing demands. Engagement was highest when response time was lower than wait time (low access-to-wait ratio), and when competing demands were low. The table shows the median for time values (wait time and response time), and the mean for engagement rate, with standard deviation in parentheses.

to fill the waiting period no longer exists. However, if competing demands are low, a user might still engage if they are highly motivated to learn. Lastly, if the access-to-wait ratio is high and competing demands are also high, then not only is there little benefit to switching, but the potential harm to the primary task is also high.

In the following sections, we describe user engagement and switch cost on each app with respect to wait time, ease of access, and competing demands. Table 5.3 shows a summary of dimensions for each app. We estimate ease of access using response time, or the time taken to begin interacting with an exercise after it appears. In the case of EmailLearner and WaitChatter, we also discuss how frequency may have affected engagement.

5.5.2 ElevatorLearner

Relative to other apps, ElevatorLearner had a high engagement rate (46.7%, $\sigma = 21.3\%$), likely because the access-to-wait ratio was low and competing demands were low. From usage logs, we observed that ElevatorLearner had the longest response time (median=10 sec), but also the longest wait time (median=52.9 sec). Even though it took some time to pull out and unlock the phone, there was enough time to complete several exercises within the waiting period. The time taken to access an exercise was substantially less than the wait time.

Furthermore, competing mental demands were low while waiting for and riding the elevator. Users described elevator waits as times when they tended to have more mental space, as they were *"not doing anything else"* or *"looking for a distraction."* Waiting for the elevator involves planning where to go, pressing a button, and standing until the elevator arrives, which are steps that typically require low attentional resources. The intermediate problem state is also low because, in normal circumstances, the user simply needs to remember where they are headed next. Thus, the low switch cost kept the engagement level high.

Although competing mental demands were low, many participants described physical demands that prevented them from engaging. For example, exercises were sometimes triggered while they were holding food or clothing in their hands, walking in a rush, or talking to others. In these cases, low ease of access and social costs were not worth the learning gains. In some cases, notifications felt disruptive when they did not match user expectations. Two users reported feeling disappointed when they expected the notification to be a text message, but got an ElevatorLearner notification instead: *"It's that feeling of oh, it's not actually important."* For some who kept their phones inside purses, the exercises were often not noticed until after the elevator ride. In these cases, users saw the notification only after getting back to their desk or while walking elsewhere, at which point they felt that the notifications had lost their intended purpose.

Because the elevator is often located in transitional areas, some users reported doing exercises even when ElevatorLearner falsely triggered during other low-attention activities, such as while walking to the bathroom or the water fountain. Although false triggers stand the risk of surprising or frustrating users, in these cases, they were welcomed because users were not only mentally available, but also able to reason why the notification happened: *"I went to the water fountain to get water, so I think the bluetooth was close enough [to get triggered]."* Thus, wait-learning triggers in ubiquitous contexts may be more effective if they are situated near other areas associated with low-attention activities, so that false triggers can fail softly.

	wait time	response time	competing demands	engagement
ElevatorLearner	high	<wait time	low	high
PullLearner	low	<wait time	low-med	high
WifiLearner (reliable)	low-med	= wait time	med	low
WifiLearner (spotty)	med	<wait time	low	med
EmailLearner	low	>wait time	varies	med
WaitChatter (primary)	low-med	varies	low	med
WaitChatter (secondary)	low-med	varies	med	low

Table 5.3: A summary of important dimensions. For each app, the table displays wait time, ease of access (measured as response time), and competing demands in relation to user engagement. WifiLearner is further subdivided into reliable internet and spotty internet, and WaitChatter is subdivided into cases where chatting is the primary task or secondary task.

5.5.3 PullLearner

PullLearner also exhibited a high engagement rate (62.5%, $\sigma = 12.1\%$). Even though wait time was very short (median email loading time = 2.3 sec), response time was the lowest across all apps (median = 1.6 sec). Unlike other apps, PullLearner exercises appear only upon a user-initiated action of pulling. Thus, when the exercise appears, the user is already looking at the learning panel with their thumb on the phone nearby, making the exercise easy to access.

Although the response time was not substantially shorter than wait time, competing demands were typically low, which contributed to the high engagement rate. Users reported that they often pull-to-refresh when they have nothing to do, such as while bored waiting in line or sitting on the toilet. As one user put it: *"Most of the time I don't have to check my email but I check it anyway. Even if I see new emails coming in, I still pull just to make sure."* According to user logs, 89% of pulls resulted in no new mail. The tendency to pull without a clear goal in mind, combined with the lack of email arrival, meant that competing mental demands were relatively low.

5.5.4 WifiLearner

Although wifi took a moderate amount of time to connect (median=6.8 sec), engagement rate was overall low on WifiLearner (17.5%). Through interviews, we found that engagement with learning varied depending on whether internet delays were expected by the user.

For the majority of users who had reliable internet, low ease of access, lack of perceived waiting, and moderate competing demands dampened engagement. First, the time taken to access an exercise (median=7.3 sec) was typically as long as the wait time itself (median=6.8 sec). Many users said that because their wifi usually connected automatically once they opened their laptops, they had little reason to click near the wifi icon, where WifiLearner was situated. Responding to WifiLearner would have required extra effort to access the top of their screen. Secondly, because these users expected internet to connect automatically,

they had developed the habit of spending the connection time opening a browser, typing in a url, or plugging in their computers, which helped them get set up for their next task and already filled the wait time: *"The time is spent typing something into the address bar already, and by then it's already connected."* The habit of setting up for the next task during this time meant that competing mental demands may have also been higher. It is possible that users were already planning what they were going to do next, leaving fewer attentional resources available for wait-learning and increasing the switch cost.

Although engagement was low for most WifiLearner users, three participants who regularly experience internet delays reported that they engaged frequently with WifiLearner. Because they expected wifi delays, these users had an existing habit of clicking on the wifi icon after opening their laptops: *"I'm one of those people that keeps clicking the wifi button. Rationally I know it'll connect on its own, but for some reason I'll check anyway."* Thus, not only was the wait time longer, but accessing the exercise also took less physical effort, because they were already clicking on the wifi icon. Compared to users with reliable internet, competing demands might also be lower because they were less likely to have initiated set-up tasks: *"When it's still connecting, you can't get anything else without internet, so learning a couple new words made it more fun."*

In contrast to expected internet delays, unexpected delays may impose greater competing demands due to the effort required to resolve or come to terms with the delay. For example, one user described being very preoccupied while looking for internet at an airport: *"You see which one isn't locked, go through all these steps to join. It's a very active process, when you're desperately looking for wifi."* The multiple steps taken to find internet, combined with the need to keep track of intermediate problem states (e.g. remembering which networks one has tried), consume attentional resources and increase the cost of task switching. Thus, unexpected waiting may be less effective for wait-learning compared to regularly encountered waiting.

5.5.5 EmailLearner

According to user logs, the time taken for email to send was negligible (median=0.7 sec). Since wait time was negligible, user engagement was largely determined by the user's attentional capacity at the time an email was sent. We observed a moderate negative correlation between the frequency of system-triggers per day and the user's engagement rate (Figure 5-1, Pearson's correlation = 0.3). Users who encountered fewer system-triggered exercises had higher engagement, a trend we did not observe in the other four apps.

These results support existing theories on attentional capacity: the more frequently exercises are triggered, the less likely they occur at the conclusion of a coarse-grained task and the more likely some will occur while the user still needs to remember an intermediate problem state. For example, one user described an instance in which he sent an email to plan a party, which involved juggling multiple logistics. He ignored the learning task because he needed to make a restaurant reservation after sending the email. Doing the learning task would delay the restaurant reservation task, and also require him to remember to book the reservation afterwards. In other words, it would require him to store and retrieve an intermediate problem state. Aside from the need to remember intermediate states, other users described their decisions in the context of efficiency: *"The way I do email, it very much feels like a to-do list going from one to the next."* Some described having a habit of sending emails in batches, and thus did not engage with EmailLearner until the last email in the batch. This behavior makes sense in the context of switch cost: sending multiple emails in a row is more efficient than switching to a learning task and back.

Despite the lack of waiting, EmailLearner had a moderate engagement rate (39.8%, $\sigma = 31\%$) because, in some cases, exercises did coincide with moments of higher attentional capacity. For example, some reported that they answered an exercise before transitioning from email to a new task, a moment when greater attentional resources may be available and problem state is low. Others said they sent emails while waiting for long-running job-related tasks to complete, in which case emailing was itself a long wait time activity. However, on the whole users did not perceive the time taken to send email itself to constitute wait

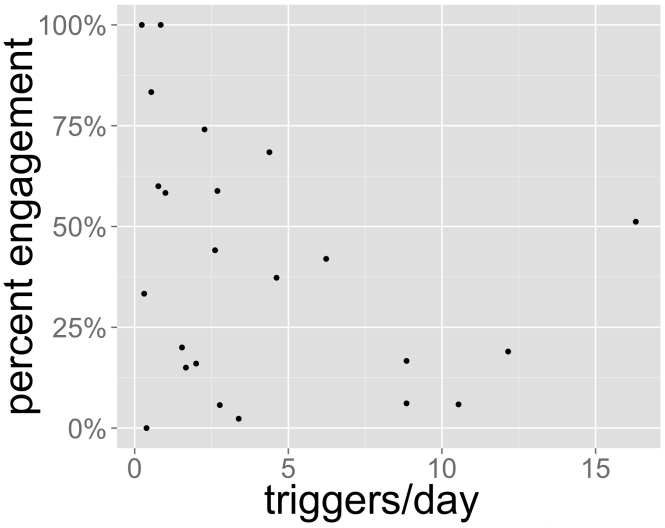


Figure 5-1: On EmailLearner, there was a mild negative correlation between frequency of exposure to system-triggered exercises and likelihood of engaging with an exercise.

time. In the absence of waiting, user engagement may be particularly sensitive to the mental demands of the primary task because the motivations for filling wait time no longer exist.

5.5.6 WaitChatter

Lastly, WaitChatter had a relatively low engagement rate (13%), even though wait time was moderate (median=10.2 sec, σ=13.1) and response time was short (median=2.9 sec, σ=2.7). However, WaitChatter users submitted the highest number of system-triggered exercises per day among all apps (6.6). Upon further analysis, we found that engagement varied depending on whether instant messaging was a primary or secondary task.

In cases where chatting was itself a secondary task amidst a primary work-related task, low engagement may be due to a lack of perceived waiting and high competing demands. In interviews, some users described chatting as a sporadic, secondary activity that interleaved with a work-related primary activity they were doing. In these cases, wait time was already being filled by a more important primary activity. In cases where chatting was already secondary, learning was a tertiary task, meaning that attentional capacity would be particularly low. For example, if the user is doing homework while instant messaging with

friends, attentional resources are already being dedicated to problem solving, composing a homework solution, switching to instant messaging, recalling the state of the instant message thread, and recalling the state of the homework problem. Switching to learning while instant messaging would add to the already high cognitive expenditure.

WaitChatter was more effective during prolonged conversations during which instant messaging was the primary task. In these scenarios, there were likely more attentional resources available and also more waiting involved: *"It's the perfect little gap where I know they're going to respond within a minute. Without the app I would probably just sit there and wait."* Frequent chatters may have been more likely to chat as a primary task. Whereas 10 users chatted fewer than half of the study days, the four most frequent chatters sent an average of 222 chats per day. Engagement rates were slightly higher among frequent chatters (17%) than infrequent chatters (11%). Frequent chatters also completed a very high volume of exercises per day (23), much higher than infrequent chatters (1.8).

For frequent chatters, some exercises may have still been ignored due to an unusually high frequency of opportunities. One user said it felt natural to ignore a large number of exercises because she knew she would have another opportunity within a few seconds: *"I knew it would show up very soon again."* Beyond a certain frequency, users may feel content in the number of exercises they have already completed, so the desire to learn more did not justify the switch cost. During particularly rapid conversations, engagement may have also been tempered by short wait times. In the future, smarter heuristics could be used to approximate whether chatting is in fact the primary task. For instance, future iterations of WaitChatter could compute the intermessage time on the fly and trigger exercises only when intermessage frequency is moderate: low frequency could indicate that chatting is not the main task, whereas very high frequency could suggest that the wait time is too short for wait-learning.

5.5.7 Self-Triggered Exercises

Although our wait-learning apps were designed to detect waiting automatically, we found that some users eventually developed a habit of self-triggering exercises even if they didn't receive a system-trigger: *"Maybe one or two times I was at the elevator and didn't get a notification, I would go in. I guess as part of the habit."* Others eventually identified waiting moments that were not already handled by WaitSuite, such as during elevator rides in other buildings, or while in the bathroom, *"Even when I wasn't near the elevator – when I was walking or getting coffee, I would realize oh, I could learn some words."* Interestingly, for some users, system-triggers may have helped facilitate the eventual adoption of self-triggers: "Near the beginning I would do more of the prompted ones. Over time, I would also seek it out on my own without the prompts."

Users were more likely to self-trigger exercises if the learning panel was positioned in always-present dead space that was easily noticed during moments when the user was waiting or bored. We found that EmailLearner had the highest number of self-triggered submissions per day ($\mu = 9.4$), followed by ElevatorLearner (5.8), WifiLearner (2.7), and WaitChatter (0.7). We did not include PullLearner in this analysis because we were unable to distinguish between system-triggered and self-triggered exercises, since exercises are always triggered when a user pulls. Users indicated that because EmailLearner is in view for a large part of the day, they often self-triggered exercises while waiting for other tasks to complete: *"When I was running an experiment that was going to take like one minute to finish...I would do a couple of them."* WifiLearner also received some self-triggers (2.7 per day) despite being peripherally located on the screen, because it was situated near other status-management tools in the menu bar: *"It's near my dropbox and bluetooth icon. I'm always looking at the bluetooth to see if my keyboard or speakers are connected, or I'm checking the dropbox syncing."* Hence, the location of WifiLearner may have helped users identify other moments that could be used more productively. In contrast, WaitChatter was in view only when a chat box was open, and received very few self-triggers (0.7 per day).

5.6 Results: Perceived Workload (RQ2)

Examining NASA TLX results, we found no evidence that users perceived an additional workload with wait-learning enabled. Because exercises were purely optional, it may be that users engaged only when the switch cost was sufficiently low. Alternatively, the additional workload could have been too small to be noticeable or measured. Regardless, these findings support our design goals of keeping exercises both bite-sized and easy to ignore.

Additionally, we saw evidence that wait-learning can reduce the frustration of waiting in certain situations. In Wilcoxon signed-rank tests, we found that WifiLearner users rated the wifi-connecting process significantly less irritating (pre=3.68, post=2.19, $p < 0.05$, $r = 0.43$) and less hurried (pre=3.75, post=2.3, $p < 0.05$, $r = 0.41$) with WifiLearner (measured post-study), than without it (measured pre-study). One participant commented: *"Waiting for wifi to load is something I get really annoyed by, and having something to pass the time by was very nice."* Unlike other kinds of waiting (e.g. elevator waiting, instant messaging), waiting for internet may be particularly frustrating because other digital tasks are necessarily delayed.

5.7 Supplementary Findings: Using Apps in Combination

As our aim is to extend wait-learning beyond any one waiting scenario, we conducted a supplementary analysis to understand the extent to which WaitSuite apps were used in combination. Overall, we found that WaitSuite provided benefits beyond being a collection of five isolated apps. A majority of participants interleaved between multiple apps, whereas a smaller fraction focused primarily on one app. Users reported benefits such as seamless synchronization of learning progress across diverse kinds of waiting, the ability to make productive use of multiple kinds of waiting moments, and the ability to sustain learning progress even on days when a specific kind of waiting was not encountered.

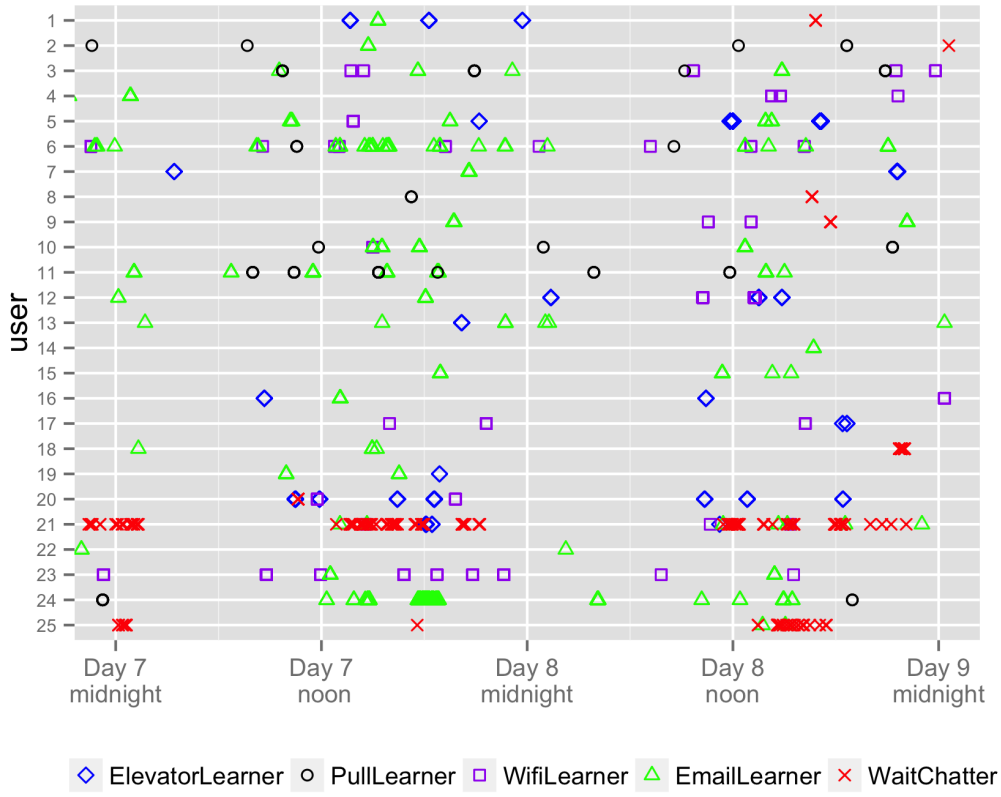


Figure 5-2: A two-day snapshot of exercises submitted by each user. Most users (i.e. users 1-17) interleaved between different apps within a single day. We label these users *generalists*. The remaining users (i.e. users 18-25) could be considered *specialists*, submitting more than 75% of exercises in just one app. These usage patterns suggest that there are multiple kinds of waiting in a day that may not be captured by a single app alone. The graph shows days at the midpoint of the study (days 7 and 8) as they were most representative of user activity overall.

5.7.1 Multi-App Usage

Analysis of our log data showed that a majority of users interleaved usage between apps, even within a single day (Figure 5-2). For example, users 1-17 could be described as *generalists*, using a combination of different apps, whereas the remaining 8 users (users 18-25) could be considered *specialists*, completing more than three quarters of all exercises in one app. In addition, all five apps had specialists, suggesting that no single wait-learning situation was redundant.

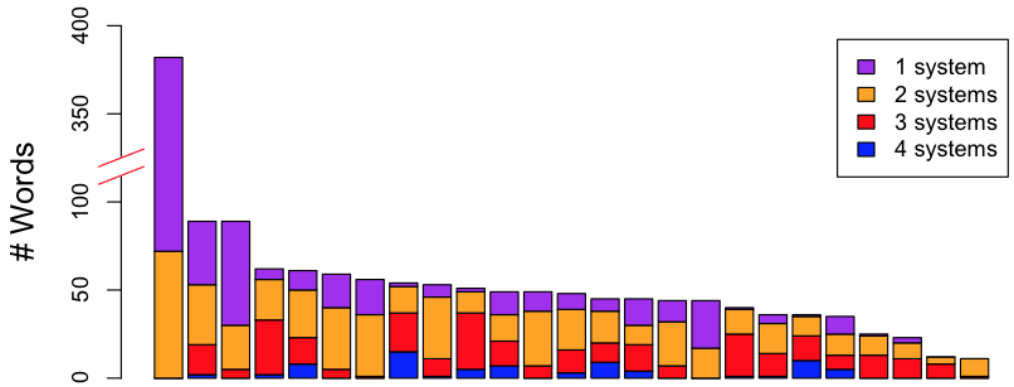


Figure 5-3: For each user, the number of vocabulary words whose exercises appeared across 1, 2, 3, and 4 systems. Users are sorted from most (left) to least (right) number of exercises completed.

For each user, we also observed the portion of vocabulary exercises appearing on multiple apps. We found that 65% of vocabulary words for each user appeared in 3 or more apps, and 77% of words appeared in 2 or more apps (Figure 5-3). These usage patterns resemble behavior that is shaped by fleeting moments of waiting within different kinds of daily activities, rather than deliberate engagement within a single app.

Lastly, to understand the potential impact of leveraging multiple wait-learning opportunities, for each user we determined the most common app, used most heavily by that user, and computed the proportion of exercises completed on that app compared to the other apps. Figure 5-4 shows the relative portions of exercises completed per user, ordered by the percentage of exercises completed on the most common app. Across all users, a non-trivial portion of exercises (35%) were completed on apps that were not the most common app, ranging from 68% (user 1) to 2% (user 25). These usage patterns suggest that there are multiple kinds of waiting in a day that may not be captured by a single app alone.

5.7.2 Unified Progress

Many described wait-learning during multiple kinds of waiting as a novel experience: *"It felt kind of cool there are all these different things that fit into these gaps and they were all*

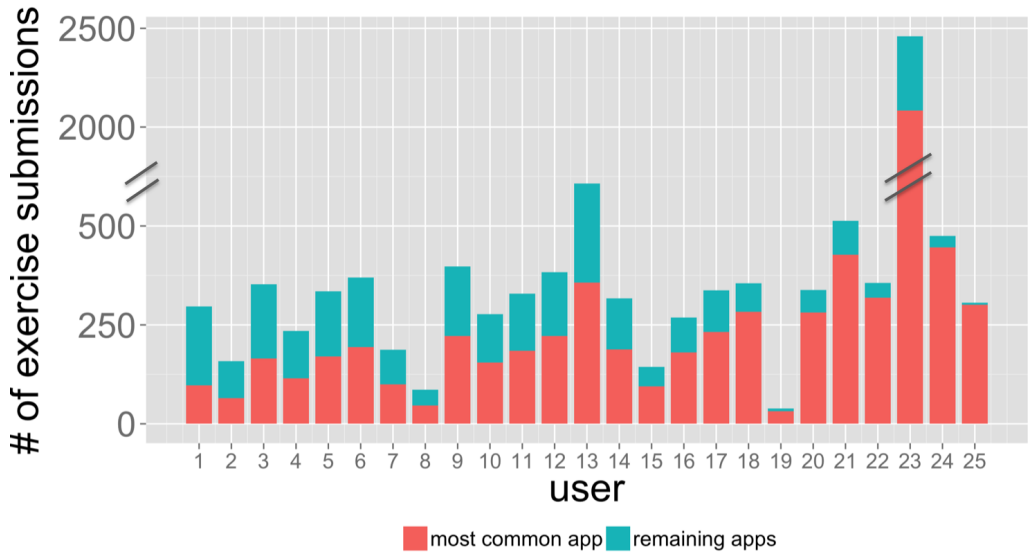


Figure 5-4: For each user, the number of exercise submissions that were completed on the user's most common app, compared to the user's remaining apps. Users are ordered from the lowest (left) to highest (right) percentage of exercises completed on the most common app. Averaged across all users, 65% of exercises were completed on the most common app, and 35% were completed on the remaining apps.

unified." In contrast to previous single-app platforms they had used, users felt that WaitSuite offered a unique experience because progress carried over from one app to another, yet the moments used for learning were themselves diverse. Several also indicated that the apps served complementary learning needs. For example, one user said that pressing buttons on his phone was better for seeing new words, while typing in translations on his computer was more effective for testing his memory.

Although one user said that he encountered the same new word on two different apps, indicating stale data, the vast majority of users described progress as flowing naturally between apps. Since vocabulary was repeatedly presented according to the flashcard scheduling algorithm, it is possible that even a stale flashcard could seem fresh, being simply another repetition.

5.7.3 Resilience to Absence

In interviews, users reported that they naturally segmented their usage of different platforms for different purposes, and some also faced unusual circumstances that prevented usage on particular apps. By targeting multiple kinds of waiting, WaitSuite enabled them to continue learning in these situations: *"If you don't use the certain app that day, you might forget everything by the next day. But by having four, they help each other to increase the frequency of you being exposed to the word."* This varied exposure was not an anomaly, but rather part of the natural fluctuation of user activities. For example, some did not ride elevators on days they had lunch with co-workers, and others used Google Chat only to communicate with family. Despite recruiting participants who reported to be regular users of multiple platforms, we found that 6 users kept non-gmail accounts in addition to gmail accounts; 3 kept their laptops only at work and one kept it only at home; 6 used desktop computers instead of their laptops at work. Several also encountered unusual circumstances during the study, such as travel, deadlines, and illness, but the combination of apps allowed them to continue learning. For example, a user was unable to use her phone while in a foreign country, but continued learning using her desktop apps. Another actively limited email usage due to a significant deadline, but continued learning on ElevatorLearner: *"This week was pretty rough, so I decided to concentrate as much as possible so only checked email once a day. The elevator app was unaffected, because being right by the elevator that didn't count as work anyway."* Over weeks and months, the unavoidable existence of such circumstances necessitates multi-faceted approaches.

5.7.4 Security and Privacy Concerns

WaitSuite may be less suitable for those who are concerned with a transfer of information across apps. Although all users were informed pre-study that only learning-related activities would be recorded, one indicated post-study that he was concerned with privacy: *"I like to have my applications isolated, I'm afraid things I do in Gmail can be logged in some other part of my life."* To address this issue, WaitSuite could in the future allow users to pick and

choose which apps they would like to be included.

5.7.5 Learning Across Apps

During the study, participants were exposed to 88 ($\sigma = 101$) words on average, 61 ($\sigma = 70$) of which they didn't already know. After two weeks, participants translated 50 words (86%, $\sigma = 11\%$) correctly to L1 and 35 (60%, $\sigma = 18\%$) words to L2. User 23 (Figure 5-4) ended the study one day early because he had completed all words available. Some users wished stale words could be revived even after having been "learned" as defined by the Leitner algorithm. Because the Leitner algorithm retires a flashcard after a number of successful repetitions, a user could forget words that were rehearsed early in the study. A spacing algorithm that incorporates temporal effects, such as that described in [53, 121], should be used in the future to improve long term retention.

In analyzing quiz results, we found that the average number of words retained (35) was not as high as that of the Feasibility Study (57), even though participants used multiple apps. Whereas the Feasibility Study recruited all regular users of Google Chat, the current study required participants to be regular users of at least 3 of the 5 platforms, which meant that some were not regular Google Chat users. As instant messaging triggered substantially more exercises than on any other app, it's possible that users in the Feasibility Study simply had more opportunities to learn. Furthermore, even though our recruitment required that participants be regular users of the indicated platforms, we found it was difficult for some participants to accurately estimate their usage a priori. For instance, one user discovered only during the study that he used Gmail primarily on his phone, and thus rarely encountered EmailLearner and WaitChatter, which were desktop apps. We believe these discoveries to reflect real-world circumstances that a wait-learning platform could face.

5.8 Discussion and Lessons Learned

Our work aims to overcome the problem of limited time by engaging people in educational activities during diverse waiting scenarios. We have presented a design space of options for wait-learning, then narrowed it to a subspace that is more effective for wait-learning. In this section, we discuss how our findings can be used by researchers and practitioners to design future wait-learning systems.

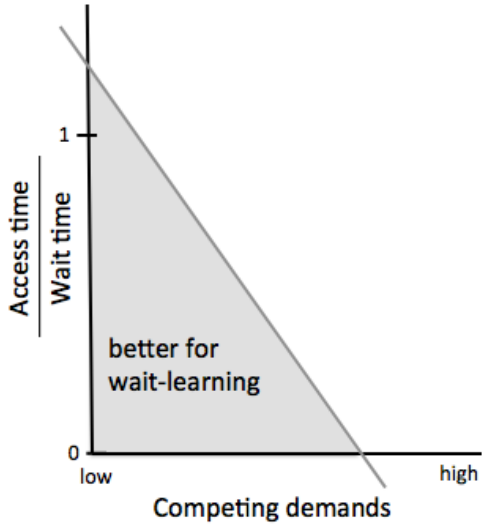


Figure 5-5: Wait-learning is more engaging and less disruptive when the time taken to access an exercise is shorter than the wait time, and when competing mental demands are low. In situations where the access-to-wait ratio is high, competing demands ought to be even lower. This depiction assumes that the learning task is short and bite-sized.

5.8.1 Theoretical Framework: Balance Between Wait Time, Ease of Access, and Mental Demands

We present a theoretical framework that illustrates a combination of constraints under which wait-learning is more engaging and less disruptive (Figure 5-5). The framework is based on our findings that wait time alone did not account for fluctuations in engagement, and that

ease of access and competing demands were also instrumental because they affected switch cost.

Under this framework, the time taken to access the secondary task should be sufficiently less than the wait time, so that the user can expect to spend enough time on the secondary task to justify the cost of task switching. In this situation, wait time is also perceived to be long enough to motivate task switching as a way of averting the boredom and frustration of waiting. Conversely, when the access-to-wait ratio is high, the waiting period may have ended by the time the user has switched. This makes the secondary task less attractive because primary task resumption might interfere with the learning task, and the motivation to avoid waiting also no longer exists. Thus, competing mental demands ought to be even lower so that the expected benefits of switching outweigh the switch cost. This theoretical framework combines existing theories on attention management [82, 104], motivation [89], and the waiting experience [98] to characterize when wait-learning is most engaging and least disruptive.

Although ease of access can often be improved through good design practices, it may be equally constrained by the main task and existing platforms within which waiting occurs. Thus, in addition to the design of seamless interactions, the effectiveness of wait-learning systems also depends heavily on the selection of appropriate waiting moments. In certain cases, it may also be necessary to analyze these dimensions separately for different types of users. For example, we found that wait time, ease of access, and competing demands varied substantially depending on whether users habitually experienced wifi delays. These observations suggest that engagement with wait-learning depends on a complex interaction of factors, and that it is necessary to consider both the existing waiting behavior and the feasibility of embedding an easy-to-access learning exercise.

5.8.2 Make Use of Frustrating Waits that are Habitual

In our study, we found that wait-learning can potentially reduce perceived mental workload during particularly frustrating waiting situations, such as during wifi connections. However, this was only true if waiting was habitual and somewhat expected. Those who only seldomly encountered wifi delays were too preoccupied with resolving the delay, making wait-learning less suitable in those situations. Hence, wait-learning is most appropriate for those who regularly experience the frustration of waiting, and have encountered it frequently enough so as not to be preoccupied with resolving it when it occurs.

5.8.3 Consider Nearby Tasks and User Expectations in Ubiquitous Scenarios

Systems using location detection for ubiquitous wait-learning should consider the nature of nearby tasks, and the physical limitations users might have. Compared to on-device waiting, we found that users were more mentally available during ubiquitous waiting, but also less physically available. Thus, wait-learning apps should avoid targeting activities in which people almost always have their hands full. To accommodate false triggers, apps for ubiquitous wait-learning should also select waiting areas that also happen to be situated near other activities involving low mental workload.

Because ubiquitous waiting often occurs when the user is not already on their device, wait-learning notifications could lead to surprise or disappointment if a user expected a different notification. Future systems should consider what the user might expect when receiving a wait-learning notification, and prevent surprises as much as possible. For example, the wait-learning notification could be customized to have a different sound or vibration from that of other apps, so that the user is aware of what they are receiving *before* opening their device. Overall, ubiquitous wait-learning is a particularly promising area to explore further, given a paucity of mental demands versus physical demands.

5.8.4 Habit Formation

In this study, our intent for supporting both system-triggers and self-triggers was to improve WaitSuite as a system: self-triggers acted as a safeguard if a user happened to be unavailable during a system-trigger, but still wanted to retrieve the exercise later. One limitation of this design was that we were unable to make study conclusions regarding the specific value of system-triggers over that of self-triggers, since WaitSuite supported both simultaneously. However, given qualitative feedback from users and existing research on habit formation, we have reason to believe that system-triggers were instrumental to engagement.

First, multiple users described system-triggers as something that helped them form a habit. For example, some learned over time to associate system-triggers with the idea of learning, such as thinking *"the elevator is the time to learn French"* when approaching the elevator (ElevatorLearner), or *"turning my focus to under the box, because now I expect [the learning task] to show up"* (WaitChatter) after sending a chat. Some felt they would eventually forget to do exercises if system-triggers were removed, but indicated that system-triggers helped facilitate the eventual adoption of self-triggering: *"I think the reminder is key. I think I need the reminders. When you get the reminder, then you're used to getting a reminder, then all of sudden you're like oh I should do it on my own."* This is in line with existing evidence that automatic triggers help people sustain desired habits better than passive applications [20]. Future work could investigate how to support habit formation more systematically, such as understanding how system triggers can facilitate self-triggers over time, or gradually reducing the frequency of system-triggers to see if self-triggers increase.

5.8.5 Integrate Multiple Wait-Learning Opportunities

In our study, we found that the extent to which a user encounters any particular waiting moment varies, and that this varied exposure is very common. Despite recruiting users who indicated they were regular users of the platforms required, in practice we found remarkable variation in the kinds of waiting that were encountered from one day to another. Some were

due to temporary constraints, e.g. deadlines or illness, while others were byproducts of existing habits, e.g. taking the elevator only on specific days of the week. For the most part, users were not practicing on one dedicated app, but were instead engaging in micro-moments of learning shaped by the diverse constraints of existing activities. A system would be naive to assume that users encounter the same set of activities to the same extent on a daily basis. Hence, a combination of wait-learning approaches should be used to increase exposure to learning opportunities.

Because our evaluation sought to understand engagement within different kinds of waiting, we preserved the same flashcard-style exercises across apps to avoid introducing additional factors and confounds. However, future work could explore learning exercises that are tailored to different waiting scenarios. For example, systems might use longer waiting periods, e.g. elevator waiting, for the presentation of more complex exercises that take longer to complete. Future systems could also integrate diverse wait-learning apps in complementary ways by capitalizing on the form factors available, such as allocating quiz-style exercises to typing-based interfaces and introducing new words in mobile settings.

5.9 Limitations

Because the focus of this research was on exploring the design space of wait-learning and understanding how people make use of multiple kinds of waiting, the evaluation was not a controlled study, but rather a real-world deployment. Because wait-learning augments existing interactions, the unique constraints of those existing interactions meant that controlled study would be challenging to perform without fundamentally disrupting the nature of the existing tasks. For example, the dead space uncovered by pull-to-refresh made wait-learning feasible despite the short wait time; moving the learning panel elsewhere would have changed the very essence of pull-to-refresh. Likewise, artificially increasing wait time or introducing competing demands would not allow us to capture the naturalistic interactions one would have in real-world waiting moments. Due to the wide spectrum of

platforms we wished to explore, it was infeasible to recruit participants who met all five waiting scenarios. The statistical analyses used in our evaluation are intended to provide some grounding to our insights, but should not be interpreted in isolation.

5.10 Conclusion

In this chapter, we evaluated five wait-learning applications targeting a variety of waiting scenarios, each with unique interaction constraints. We found no evidence that wait-learning increased the perceived workload of the primary task, and it alleviated frustration during internet waiting. Furthermore, the availability of multiple kinds of wait-learning helped sustain learning progress during absence on particular apps. Finally, we presented a theoretical framework that describes how the combination of wait time, ease of access, and competing demands affects learning engagement and interruption. Taken together, these findings provide important implications for designing effective wait-learning systems, extending wait-learning beyond any single situation. In the next chapter, we explore and validate the long-term impact of wait-learning through a controlled study comparing wait-learning to traditional reminders.

Chapter 6

Long-term Study

An important question to answer is whether wait-learning can increase productivity beyond existing alternatives in the long term, and the extent to which wait-time microtasking might affect ongoing activities. On the one hand, because wait-learning enables microtasks to appear just-in-time and to be easily within reach, it may lower the barrier to learning practice and thus increase the rate of learning compared to existing approaches. On the other hand, because wait-learning takes place during fleeting moments, and is embedded directly adjacent to ongoing activities, wait-learning could affect the user's mental workload and ability to perform, resume, or sustain an ongoing task.

In this chapter, we investigate wait-learning's long-term impact, both on learning outcomes and on existing activities. We thus ran a 2-month-long field study, in which we compared wait-learning to traditional reminders. For this study, we used WaitChatter as our implementation of wait-learning. To measure learning impact, we tracked vocabulary practice and evaluated knowledge retention in the two conditions. To measure the impact of wait-learning on the existing task, we compared instant messaging behavior and perceived mental workload between the two conditions. On the one hand, if wait-learning distracts from the existing instant messaging activity, users may take longer to resume chatting, find it more mentally demanding, or abandon it altogether. On the other hand, because wait-learning tasks are bite-sized, and are designed to support easy task-switching, it could help users

continue chatting after the waiting period has ended.

Overall, we found that users practiced vocabulary significantly more frequently with wait-learning than with reminders, and ultimately retained more vocabulary at the end of the study. Furthermore, users preferred wait-learning over traditional reminders, felt that it was easier to find time to learn, and found it significantly less annoying than traditional reminders. Finally, users replied to instant messages more promptly in the wait-learning condition and were more likely to still be near the chatbox when receiving replies. Taken together, these results show that wait-learning has significant empirical benefits over standalone applications with traditional reminders, both objectively in terms of productivity output, and subjectively in terms of user experience. Furthermore, it may also have the potential to help people stay on existing activities while waiting, potentially combatting common tendencies to task switch.

6.1 Research Questions

The questions we sought to answer were:

1. RQ1 (Learning Retention): How much do users learn via wait-learning, compared to traditional reminders?
2. RQ2 (Learning Practice): How frequently do users engage with learning practice when given wait-learning triggers, compared to traditional reminders?
3. RQ3 (Existing Activity): How does wait-learning affect the existing activity of instant messaging?

6.2 Evaluation

To explore these research questions, we conducted a two-month (8-week) within-subject field study, in which participants experienced the *wait-learning* condition and the *reminder*

condition on alternating weeks. The order of which condition appeared first was randomized. On wait-learning weeks, WaitChatter was made available, so users received wait-learning prompts from within their Google Chat instant messaging interface. On reminder weeks, users had access to a standalone flashcard website, and received an email reminder if no flashcards had been completed that day. The reminder linked to the website so that users could click into the website and complete flashcards if they'd like. We designed this condition to mirror the behavior of many current applications, which attempt to boost engagement by sending reminders to users through email and mobile notifications. Rather than simply using a standalone website as the control condition, we included these reminders as a more robust control, since existing research has shown that regular reminders boost engagement over passive applications [20]. Note that in the reminder condition, WaitChatter was not available, but users could still instant message as usual.

To keep flashcard interactions consistent between conditions, the flashcard user interface was identical in the two conditions, except for being situated under the chatbox in the wait-learning condition and in the standalone flashcard website in the reminder condition. Similar to the Multi-System Deployment (Chapter 5), we removed the contextual feature of WaitChatter so that the two conditions present vocabulary in the same way. Furthermore, both conditions used the MemReflex flashcard algorithm [53] to schedule the order of flashcards.

Evaluation of Impact on Learning

So that we could evaluate learning outcomes in the two conditions separately, each user was exposed to two disjoint sets of vocabulary words, one for each condition. Similar to the Feasibility Study and Multi-System Deployment, the languages being learned were Spanish and French, and the vocabulary words were drawn from high frequency English nouns as measured in the British National Corpus. To keep the difficulty levels similar between the two sets, we first ordered the vocabulary list by frequency based on the British National Corpus, then placed every other word into the same set. Finally, we randomized which

vocabulary set was matched to which condition for each participant.

At the end of each condition (end of week 7 and week 8), users completed an online vocabulary quiz on words from the respective condition. The quiz tested all vocabulary the user had been exposed to (but didn't already know) within that condition, and was divided into two parts: they first translated from L1 to L2 on the recall quiz, then from L2 to L1 on the recognition quiz. Within each part, the order of questions was randomized.

Evaluation of Impact on Existing Activity

So that we could compare the impact of wait-learning on the existing activity of instant messaging, users were free to instant message during all weeks regardless of condition, even though WaitChatter was on during wait-learning weeks and off during reminder weeks. We then compared instant messaging behavior and subjective experience between wait-learning weeks and reminder weeks. Specifically, we sought to evaluate the following behavioral metrics:

- Reply time: the time taken for a participant to initiate a reply after receiving a chat message. A longer reply time could indicate that it is more difficult to continue chatting while wait-learning.
- Correction Attempt: whether the user attempted to correct a message by hitting delete or backspace while typing. More corrections could mean that a user had made more mistakes while typing chat messages.
- Session Duration and Message Count: The length of time and number of messages exchangedbetween the first and last message in the sesion, respectively.
- Session Speed: The Message Count of the session, divided by its Duration.

To compute Session Duration, Message Count, and Speed, we define an IM session to be a series of instant messages in which no two consecutive messages are separated by more than 5 minutes, following existing work [9, 76]. On wait-learning weeks, we define a Learning Session to be any session during which a user responded to one or more flashcards.

To capture perceived workload, users answered the NASA-TLX mental workload survey at the end of each condition week, with respect to their chatting experience that week on a 7-point scale. To keep the survey short, we used only the Mental Demand and Frustration questions from the NASA-TLX questionnaire (Appendix C).

6.2.1 Procedure

Participants first met with a researcher to have WaitChatter installed as a Chrome extension on their personal computers. They were then given a walkthrough of how WaitChatter could be used, as well as how the standalone flashcard website could be used. Users were told that they could do the flashcards as little or as much as they wished during the course of the study, and told to behave as if they had installed an app on their own.

At the end of each condition week, users completed the NASA-TLX survey with respect to their chatting experience that week, as described above. On the last day of each condition (end of week 7 and end of week 8), users completed the post-study quiz for that condition. Finally, at the end of week 8, users completed the post-study questionnaire and semi-structured interview. The questionnaire consisted of Likert scale questions regarding their experience using WaitChatter and their experience using the reminder and flashcard website combination (Appendix C).

6.3 Participants

21 participants were recruited through email lists and social media. We selected only those who regularly used Google Chat in the web browser, desired to learn or were currently learning Spanish or French, and were not traveling extensively during the two-month study period. Participants were given a $50 gift card for their time, and were told that the payment was not tied to the amount learned.

The participants included 9 males and 12 females, ages 18 to 32 ($\mu = 24$). They consisted of

9 undergraduate students, 4 graduate students, and 5 professionals with fields ranging from consulting and information work, to engineering and architecture. 14 chose to learn French and 7 learned Spanish. Eight users had prior formal education in the language ranging from elementary school (4) and middle or high school (5) to university-level classes (2). 18 of the participants had studied the language informally through using language learning software, traveling in a foreign country, talking to friends, or helping their children with homework. Two participants had never studied the language before, either formally or informally. The participants typically use Google Chat on their computers “Several times a day" (16) or “Several times an hour" (5), mostly for social reasons or for keeping in touch with loved ones.

6.4 Results: Learning

In the three sections below, we report results corresponding to our three research questions.

6.4.1 Learning Retention (RQ1)

First, to answer RQ1, we computed the number of words answered correctly on the post-study quizzes, which consisted of both the recall quiz (translate from L1 to L2) and recognition quiz (translate from L2 to L1). The quiz results indicate that participants ultimately learned more words through wait-learning than through traditional reminders (Figure 6-1). On the recall quiz, participants on average translated 28 words ($\sigma = 20$) correctly in the wait-learning condition, and 15 words ($\sigma = 22$) correctly in the reminder condition, a difference that was statistically significant ($p < 0.01$, $t = 3.1$). On the recognition quiz, they translated 34 words ($\sigma = 21$) correctly in the wait-learning condition, and 19 words ($\sigma = 25$) correctly in the reminder condition ($p < 0.005$, $t = 3.4$).

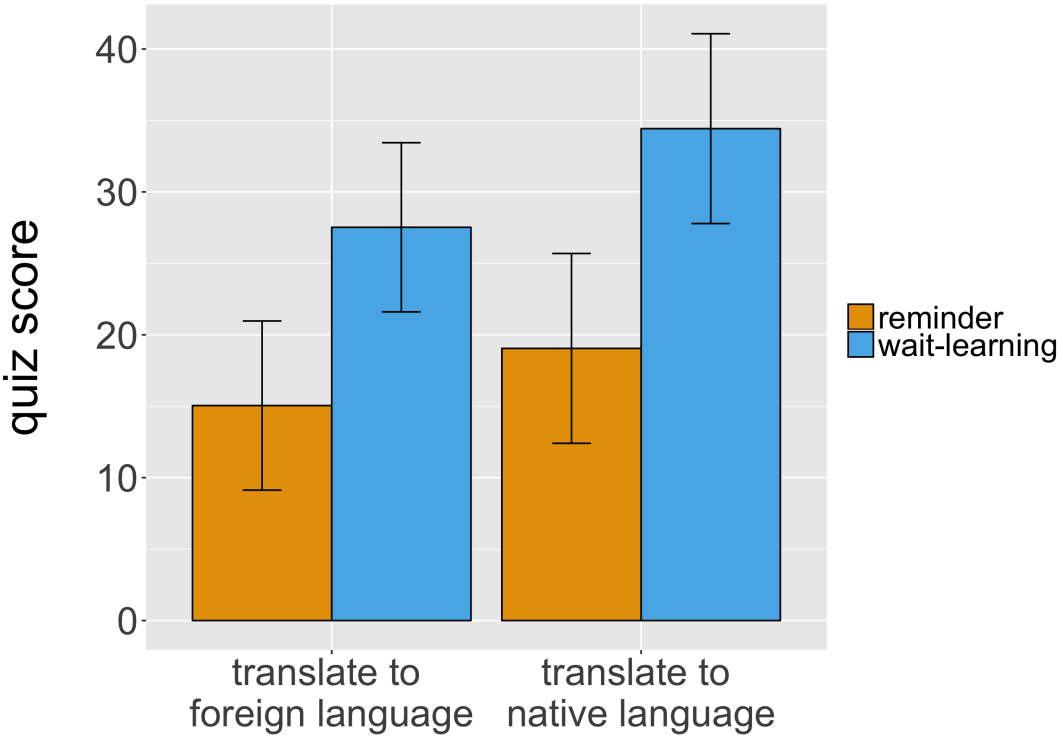


Figure 6-1: The number of correct translations made on the post-study quiz. Users learned significantly more words in the wait-learning condition than in the reminder condition.

6.4.2 Learning Practice (RQ2)

To analyze the frequency of learning practice, we computed the number of learning exercises completed by participants in each condition. As shown in Figure 6-2, about twice as many flashcards were submitted in the wait-learning condition (415, $\sigma = 307.3$) than in the reminder condition (217, $\sigma = 322.5$), a difference that was statistically significant (paired t-test: $p < 0.01$, $t = 3.69$). However, users completed approximately three times as many flashcards *per chain* in the reminder condition (40, $\sigma = 39$), compared to the wait-learning condition (14, $\sigma = 8$), a difference that was statistically significant (mixed-effects linear regression, $p < 0.0005$, $t = 11.1$). Thus, the wait-learning version closely resembled episodes of microlearning accomplished within existing time constraints, whereas the reminder version was more similar to “cramming" a large number of flashcards into a single sitting, a behavior that is rarely beneficial for long-term retention [119].

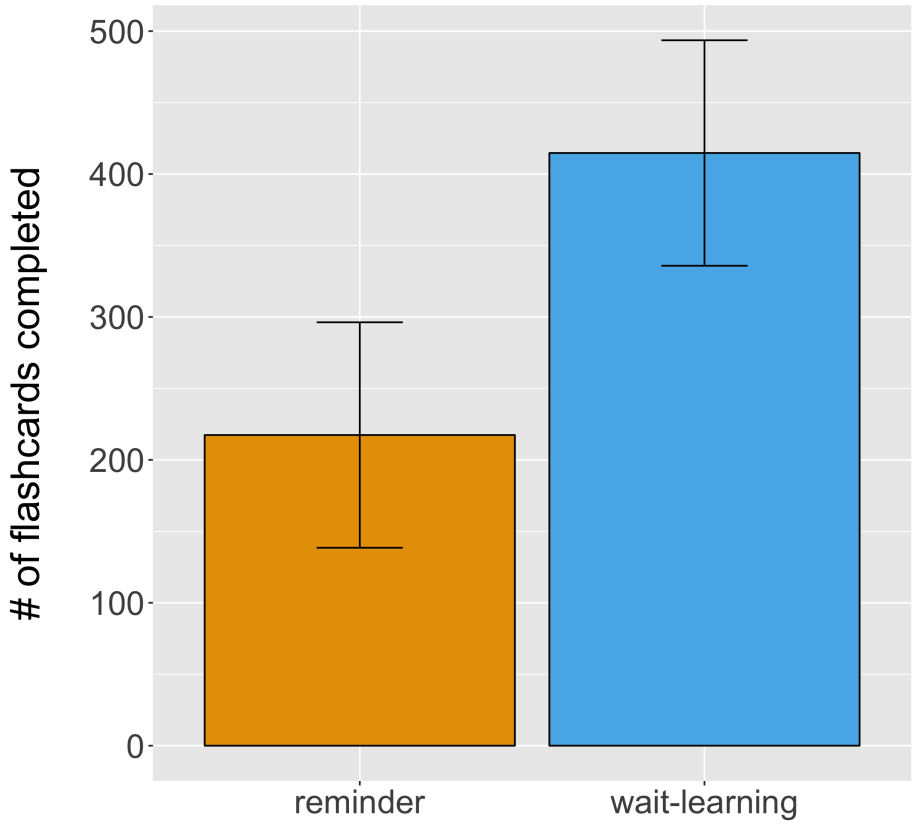


Figure 6-2: The total number of flashcards submitted during the study, in each condition. Users completed significantly more flashcards in the wait-learning condition than in the reminder condition.

In addition to the overall quantity of flashcards completed, we also examined how *regularly* users engaged with learning. To do so, we computed whether or not participants engaged in learning each day. We found that learning occurred on more wait-learning days (35%, $\sigma = 65\%$) than reminder days (16%, $\sigma = 48\%$), a difference that was statistically significant (mixed-effects logistic regression: $p < 0.001$, $t = 8.01$). Thus, users not only completed more flashcards overall, but were also engaging in learning practice more regularly in the wait-learning condition.

Finally, we examined the efficiency of learning between conditions. On the one hand, wait-learning could be a more efficient way to learn because it spaces out learning across time, which is known to increase long-term retention. On the other hand, the reminder condition may offer a more focused environment for learning, with fewer distractions. To compute learning efficiency, we divided the total number of words learned (correctly translated to the

foreign language on the post-study quiz) by the total number of flashcards completed per user. Because the flashcard algorithm introduced words in order of difficulty, users who had only been exposed to a few words appeared to have very high learning efficiency because those easy words were easy to translate, regardless of learning practice. We thus excluded users who had been exposed to fewer than 10 words in either condition. On average, the learning efficiency was 6.4 ($\sigma = 2.9$) in the wait-learning condition and 6.7 ($\sigma = 3.8$) in the reminder condition, a difference that was not found to be statistically significant ($p = 0.78$).

6.4.3 Subjective Experience

On the post-study Likert scale survey (Figure 6-3), users were asked to rate their impressions of the two conditions. In the wait-learning condition, they found it easier to find time to learn vocabulary ($\mu = 4.2$ vs. $\mu = 2.4$, $p < 0.0005$, $V = 222$), believed they were more likely to engage in vocabulary practice ($\mu = 6.2$ vs. $\mu = 3.3$, $p < 0.0005$, $V = 167.5$), were more likely to continue using this approach ($\mu = 6.1$ vs. $\mu = 2.4$, $p < 0.0001$, $V = 246.5$), and found the experience more enjoyable ($\mu = 5.95$ vs. $\mu = 3$, $p < 0.0005$, $V = 225$). Compared to wait-learning, users found the reminder prompts to be significantly more annoying ($\mu = 1.8$ vs. $\mu = 4.0$, $p < 0.005$, $V = 7.5$). Overall, 19 participants preferred wait-learning, and 2 users preferred the flashcard website with reminders. Below, we discuss several themes surrounding why wait-learning was preferred by the majority of participants, and the concerns raised by the two remaining participants.

Wait-Learning is Perceived to be Less Time Consuming

First, users unanimously felt that the flashcard website and reminder demanded greater time commitment: *"I felt like it would take extra time to do the flashcards and I didn't want to purposefully take the time to do it. Whereas in chat it didn't feel like extra time, it was just while I was doing something else."* In contrast, wait-learning occurred at times when users were less preoccupied and more mentally available, making the transition into learning feel

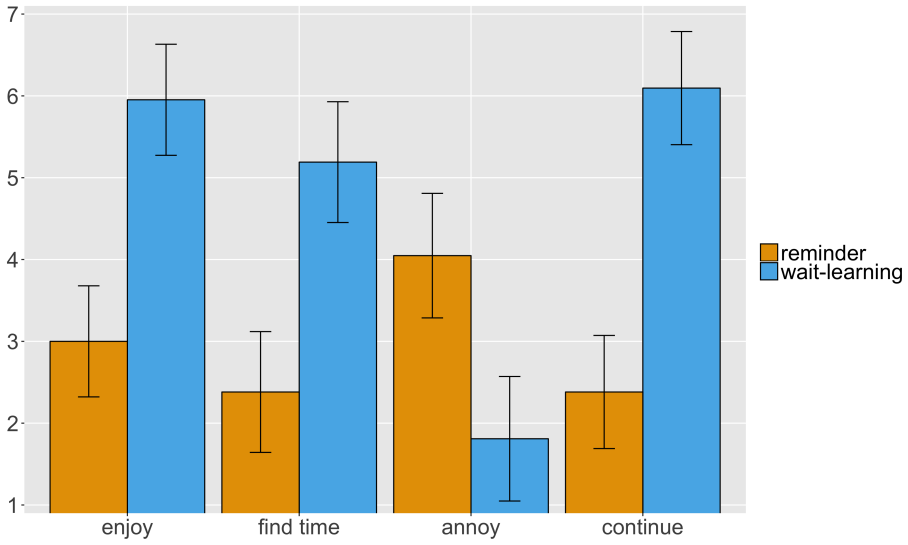


Figure 6-3: On the post-study Likert scale survey (Appendix C), users indicated that they enjoyed wait-learning more, found it easier to find time for learning, felt that wait-learning triggers were less annoying, and wanted to continue wait-learning more, compared to traditional reminders.

less effortful: *"The in-chat one I liked a lot better because otherwise I would just be waiting. It lowers the activation energy barrier to get started on learning vocab."* Many participants also described chatting as an activity they do when they're already in a bored or less focused mood: *"I'm not already in a focused state, and I'm not interested in work at the moment. I'd just be talking to a friend because it's not that important. I like utilizing that time, because it's time that I would've lost doing something useless."*

Many users described entering a separate application to be a substantial context switch from their ongoing activities. The effort required to switch was greater than their casual desire to learn: *"Learning the language is sort of important to me, but it's not particularly high up in my to-do list, so the activation energy to open the email and then go to the site and break my workflow is more than I'm willing to pursue."* Some appreciated that wait-learning offered a seamless transition into learning, so that the decision to learn did not need to be made explicit: *"Part of the disadvantage of email is that you have to choose to do it very actively, versus in chat it's less active. You just press tab so you could do it without thinking too much about choosing to do it. It doesn't require all my attention."* By making learning well-timed and convenient to access, wait-learning better matched users' goals of learning

informally without sacrificing their higher priorities.

Two users preferred the reminder condition, each for different reasons. One had never been exposed to the language before, and struggled to learn new words while chatting: *"I guess I'm kind of an emotional vocabulary practicer, but I would get frustrated with myself if I was wrong several times in a row. I found it easier to use the website because it meant that I was setting aside just vocabulary time."* She wished WaitChatter would be more sensitive to her complete-beginner skill level. Although WaitChatter currently requires the user to recall the translation, in the future it should provide more scaffolding by providing recognition-based exercises as well, such as multiple choice questions.

The other user who preferred the reminder condition indicated that he uses Google Chat primarily for work-related conversations and Facebook Chat for leisure chat. Thus, contrary to its purpose, WaitChatter almost always presented learning opportunities when he did not have time to engage in a secondary task. Although we had attempted to recruit users who use only Google Chat on a regular basis, the reality is that many people use different chatting platforms for different purposes. In addition, the email reminders were relatlively effective for this user because he had an existing habit of addressing all emails once at the end of the day, rather than checking emails throughout the day: *"I generally don't reply to emails at all until the end of the day. Emails are such a distraction, why not just go home and have dinner and then reply to emails? I just go chronologically and go through all of them."* This user expressed enthusiasm about implementing WaitChatter for other chat platforms like Facebook Chat, so that he could use it during leisure chatting.

Reminders Appear Amidst Higher-Priority Tasks

A majority of participants attributed their lack of engagement with email reminders to their general habits of reserving emails and notifications for higher-priority tasks: *"When I'm doing email I'm often in a tasks mode, I'm not in let's learn languages for fun mode. I have a set of stars to keep track of various emails I need to deal with."* Some described typically being overwhelmed with the sheer volume of mobile notifications they would receive in a

single day, and described a compulsion to keep their notification bars short and tidy: *"I hate when notifications clutter up my notification bar so I usually get rid of them pretty quickly. If you have too much then you can't see the new notifications."*

Because language learning reminders often appeared adjacent to more important mobile notifications and work-related emails, they tended to be ignored: *"I don't think I clicked on the flashcards more than once because the thing above or below I needed to finish first."* Others deleted the email reminders immediately because they felt more like spam than like work: *"I get a lot of useful spammy stuff in my emails, like mailing lists...I would just archive it along with the other things that are low priority."* Even for those who used email management strategies to remind themselves to do the flashcards, such as starring the language reminders or leaving them unread, most admitted those emails would eventually fall further and further down their inbox, giving way to more important emails. Because emails and mobile notifications remain until they are deleted or dismissed, users felt that low priority messages introduced an unecessary extra to-do item that needed to be handled. As a result, some users developed a habit of ignoring the language learning email reminder altogether: *"I got into the habit of seeing it and thinking oh that's a thing, but I'm doing something else."* In contrast, they liked that wait-learning flashcards disappeared on their own if ignored, and thus felt more transient.

Wait-Learning was Preferred over Existing Wait-time Activities

Most users described filling their wait time with other low-effort tasks such as browsing social media, handling email, watching youtube, reading news articles, or listening to a podcast. Some who tend chat at work described using software applications, searching on google, or reading pdfs.

Compared to existing wait-time activities, users indicated that wait-learning was more attractive because it was easier to interleave with chatting, without needing to swap windows or change tabs. Unlike wait-learning, whose learning panel is positioned directly adjacent to the chat window, other tasks usually occlude the chat window: *"Sometimes [the other*

task] covers up the chat window so takes a tiny bit longer to go back and forth." Aside from the physical effort required to switch tasks, wait-learning was also less mentally demanding compared to more involved wait time activities: *"It felt easier than looking at a comic or watching a video or reading a longform article. Those would get me into a different mindset and maybe higher cognitive load."*

Wait-learning was effective not only because it was low-effort and easier to access than existing wait-time activities, but also because users felt it led to longer-term gains. For instance, one user likened wait-learning to crocheting, which she often does while waiting: *"It's more interesting and better for me than just passively taking in things on the internet. I get something out of both of them. Crocheting I get a hat or a scarf. With flashcards I learn new things."*

6.5 Results: Impact on Existing Activity (RQ3)

To analyze the potential impact that wait-learning had on instant messaging, we computed the following instant messaging metrics, and compared them for wait-learning days and reminder days: Reply Time, Corrections, Session Duration, Session Message Count, and Session Speed. Because WaitChatter was not available on reminder days, data collected on those days served as a control, representing the participants' normal instant messaging behavior.

6.5.1 Instant Message Characteristics

Analysis

For all analyses below, extreme outliers were removed using the standard range of 1.5 x the interquartile range. For any timing data that was not normally distributed, we applied a log-transformation before running significance tests.

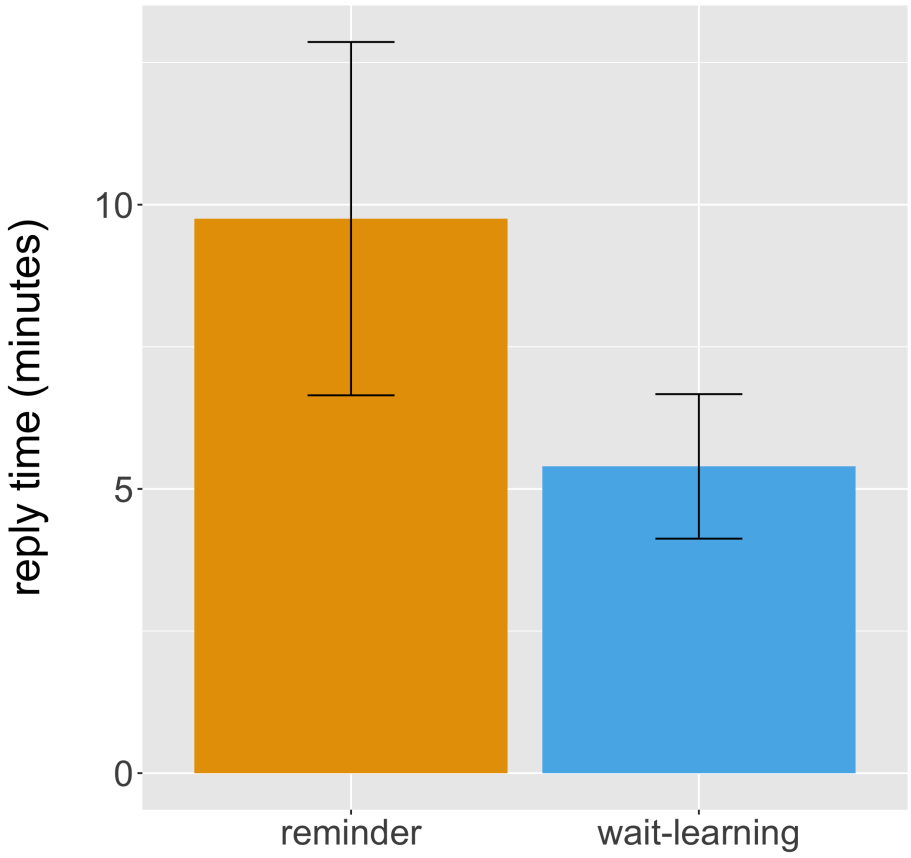


Figure 6-4: In the wait-learning condition, users initiated replies to chat messages significantly faster than in the reminder (control) condition.

Reply Time

We first compute the time taken between a user receiving a chat message and initiating a reply, defined as the first keystroke taken after receiving a message. Participants replied faster in the wait-learning condition ($\mu = 5.4$ minutes, $\sigma = 11.4$) than in the reminder condition ($\mu = 9.8$ minutes, $\sigma = 43.4$). A linear mixed effects regression, with Prompt Type as the fixed effect and Participant as a random effect, found that the reply time was significantly shorter in the wait-learning condition ($p = 0.0095$, $t = -2.6$).

Upon further analysis, we found that users may have been more readily available to reply in the wait-learning condition because they were less likely to have switched to other activites while waiting, which would typically take them to other browser tabs and application windows. To investigate this possibility, we determined whether or not the browser tab

containing the chatbox (the gmail tab) was in focus for every chat message received. We found that the gmail tab was in focus 55% of the time ($\sigma = 15\%$) in the wait-learning condition, and 43% of the time ($\sigma = 13\%$) in the reminder condition. A logistic mixed-effects regression found this difference to be statistically significant ($p < 0.0005$, $t = 3.5$).

In post-study interviews, users described wait time task switching behaviors that are consistent with these observations. For example, one participant said: *"To go to facebook I have to go to a new browser tab. But the flashcards were easy because it was right there. I didn't have to do anything extra or open any windows. Just tab down to the next textbox and do the flashcards."* Because wait-learning could be done near the chatbox, users were more likely to reply promptly to chat messages: *"I bet my response time improved [with wait-learning] because I...wouldn't pull up other windows over [the gmail window], I would actually wait for a response because I'd have something to do while waiting. Otherwise I might do 12 other things and come back hours later and my roommate will have asked a question I haven't answered."* Hence, wait-learning may have dampened the tendency to switch to other tabs and activities while waiting, a behavior that often causes a chain of further diversions away from the existing activity [75].

Corrections

For each chat message, we computed whether the user typed at least one Backspace or Delete, which is indicative of a correction attempt. On average, 37% of messages had a correction attempt in the reminder condition ($\sigma = 1\%$), compared to 38% in the wait-learning condition ($\sigma = 0.9\%$). A logistic mixed effects regression found no significant difference between the likelihood of making a typing correction while chatting in either condition. Hence, we found no evidence that participants made more typing mistakes as a result of wait-learning. However, given that we did not have access to the content of the messages themselves, and users may not always recognize or care to correct their own mistakes, these findings remain preliminary. While most users described the transition between chatting and learning to be seamless, some expressed being initially wary of accidentally typing the vocabulary word

into the chat message, but became comfortable with the system over time.

Session Duration, Message Count, and Speed

Overall, we found no significant difference in Session Duration, Session Message Count, and Speed between the wait-learning and reminder conditions. Since learning only took place during a portion of instant messaging interactions on wait-learning weeks, we further identified chat messages that were sent during Learning Sessions, defined as sessions during which the user responded to at least one flashcard. On wait-learning days, Learning Sessions had a longer Duration (median=6.9 minutes, $\sigma = 3.3$) compared to Non-learning Sessions (median=5 minutes, $\sigma = 2.5$). Similarly, Learning Sessions also had a higher Message Count (median=15.6, $\sigma = 13.3$) than Non-learning Sessions (median=11.5, $\sigma = 9.2$). These differences were statistically significant ($p < 0.001$, $t = 3.9$ for Duration and $p = 0.003$, $t = 2.9$ for Message Count).

One reason why chatting sessions were longer could be that wait-learning enabled users to remain productive without physically or mentally leaving the chat conversation. In contrast, existing wait time activities such as browsing social media or reading articles may break the flow of the conversation by taking the user to other tabs and applications. This is supported by the observation that users replied to chat messages more promptly, as described above. However, because wait-learning was optional, it is unclear whether the presence of learning tasks enabled longer conversations, or whether users were more likely to engage in wait-learning at moments when they anticipated longer conversations. For example, many users indicated that they were more likely to do flashcards if they expected to receive a response sometime soon; if they did not believe the other person was present, there was less reason to wait. It is possible that both factors were at play: anticipating a back and forth conversation could have motivated users to wait-learn, which in turn kept them attentive on the conversation and thus kept the conversation going.

In addition, we found that the overall speed of the conversation was not any slower when interleaved wait-learning. This was surprising given that users sometimes reported delaying

their reply for a few seconds while finishing a chain of flashcards: *"It was kind of like oh, let me finish this word. Oh there's another word, let me do that one too."* While replies may have been briefly delayed due to flashcard completion, they were presumably also delayed by existing wait time activities, many of which involve looking elsewhere on the screen or leaving the window altogether. As one user put it, *"I would already procrastinate in other ways, like open up a web browser and look at crap news articles."*

Perceived Workload

From NASA-TLX survey results, we saw no evidence that users found instant messaging more mentally demanding or frustrating when wait-learning was available, compared to regular instant messaging. Although the overall means were higher for wait-learning (mental demand=2.2, frustration=2.1) compared to the control (mental demand=1.7, frustration = 1.8), Wilcox signed-rank tests did not find these differences to be statistically significant. However, because the difference in mental demand was marginally significant ($p = 0.06$), we elaborate on the feedback we received from the weekly survey comments and participant interviews below.

In post-study interviews, users were asked to describe their experience switching between chatting and learning. All but one participant described the experience to be low-effort and seamless. Some reasoned that the context-free nature of flashcards made them easy to interleave with chatting: *"I think it would be a little more challenging if it were translating a sentence, but it's literally just a vocabulary word. It's a pretty discrete task."* Others noted that their chat conversations tended to be lightweight: *"A lot of the conversations aren't super substantial, so that made it easier to switch back and forth. Like 'what are you up to,' which is fairly isolated anyway."* Several participants also indicated that the ongoing state of the conversation was always visible, making it easy to keep track of the conversation: *"The chat has a little log right above it so you know what you were talking about in the past, so I don't think there was any stress added."* Taken together, these reports are consistent with existing research findings on multi-tasking: the cost of multi-tasking is decreased when

little context needs to be maintained in working memory, and when the intermediate state of the ongoing activity can be accessed from an external source (e.g. a chat log).

As mentioned above, it was more challenging to learn completely new words if the wait time was short. Those who have no prior exposure to the language may need to devote more mental energy acquainting to new word structures. In the future, wait-learning systems could distinguish between introducing brand new content and rehearsal of existing content, so that the *first* appearance of a flashcard is timed at a moment when the chatting workload is extremely low. This may be critical for complete beginners approaching a language for the first time.

Finally, several participants indicated that they weren't sure how to answer the weekly NASA-TLX surveys because the survey asked how mentally demanding it was to chat, an activity they felt was not mentally demanding at all. Furthermore, because the survey was weekly, their answers were based on their overall impression for the entire week, rather than in-the-moment impressions. Therefore, several users indicated that they answered entirely based on whether they had a particularly stressful conversation that week due to a stressful event (e.g. about national election results), which may have unecessarily skewed the results drastically from one week to another. While we limited the NASA-TLX surveys to once a week given the long duration of the study, future studies of this nature should consider alternatives so that data is collected closer to when users actually experience chatting.

6.6 Discussion

Our findings demonstrate the empirical benefits that wait-time learning tasks can have on learning practice, as well as its impact on the primary task. Below, we discuss the benefits and tradeoffs of wait-learning in comparison to existing alternatives.

6.6.1 Timing of Behavioral Triggers

The findings in our studies support existing theories that well-timed triggers are key to sustaining a habit [57]. In the Multi-System Deployment, some contrasted the timing of WaitSuite system-triggers to that of daily app reminders they had received in the past: *"One app sent me a daily notification at 8pm. I learned to ignore it after a while because the timing was just not convenient. But with these, at least I would open them because I had nothing else to do at the time."* Results from the Long-term Study provided further evidence that wait time reminders are more effective than traditional reminders. Future systems could explore how wait-time triggers and end-of-day reminders can better complement each other.

6.6.2 Low-effort Interfaces for Low-priority Endeavors

Because wait-learning tasks were timed to appear at moments when users were more available, they decreased the perceived task-switching effort and time commitment required to learn. In contrast, traditional reminders typically appeared while users were handling more important to-do items, making it more costly to task switch and thus demanding a more conscious commitment. Regardless of the actual time it took to complete a flashcard, what appeared to matter more was the *perceived* time commitment and the level of conscious effort required to initiate learning.

Interestingly, users found it particularly compelling when features of the wait-learning interface reflected the casual, hobby-level priority they devoted to language learning. For example, the self-disappearing nature of a wait-learning flashcard gave the impression that it was optional, low-stakes, and transient. In contrast, email reminders remained permanent until actively removed, and opening a separate website made learning feel like starting a big task. The standalone website felt incompatible with the low-commitment, casual nature of informal learning. Future work could explore the possibility of dynamically transforming the interface depending on the user's level of attention and commitment to learning. For instance, the learning panel could gradually transform the more flashcards

a user has completed within a chain, eventually resembling the standalone website when the user has demonstrated a commitment to learning. Alternatively, email reminders and notifications for low-priority hobbies could be made more lightweight by self-dismissing after a time threshold.

6.6.3 Impact on Existing Activity

This study sought to examine whether wait-learning might make an ongoing activity more difficult to resume, given that it occurs in the midst of an existing activity. Contrary to expectation, we found that users were *faster* to resume chatting in the wait-learning condition, and might also lead to longer chatting sessions. We attribute this to the reality that users already have a habit of task switching while awaiting instant message replies, a self-interrupting behavior that can sometimes lead to a subsequent chain of diversions [75]. Relative to those competing activities, the vocabulary exercises ironically made it easier to sustain an IM conversation due to their short, context-free nature and physical proximity to the chatbox. This effect is similar to that of progress bars and waiting indicators which, in addition to providing information, keep users from turning to other tasks by giving them something to watch while waiting.

Wait-learning may not be desireable in cases where a learner is struggling to learn words. In this situation, not only would learning be impeded by the conversation, but the learner may also devote more mental energy to the flashcards, potentially delaying the conversation. In these situations, the system ought to offer intermediate exercises with more scaffolding, such as multiple choice questions or fill-in-the-blank questions with some letters filled in.

Although our results show that wait-learning supports the continuity of an IM conversation, one scenario worth considering is the case where instant messaging is itself an interruption to another bigger task. In such a situation, wait-learning could make it easier to perpetuate the IM conversation, which may not be desireable. In interviews, users reported that they tended to ignore flashcards if they were working or doing homework while instant messaging. Due

to the length of the study and privacy considerations, however, we logged only chatting activity and not all computer activity, and thus cannot draw firm conclusions regarding this. With more data, wait-learning could be made more intelligent by determining whether instant messaging is itself a primary activity or secondary activity, and only show wait-learning tasks if instant messaging is the primary activity.

6.7 Conclusion

In this chapter, we showed that wait-learning has significant benefits on the frequency of learning practice, the enjoyment of learning, and ultimately the amount learned. Contrary to taking people away from the existing activity, we found that wait-learning tasks can *aid* in resuming and perpetuating the existing activity, provided the learning task is lightweight, context-free and situated close to existing activities. Finally, compared to a standalone website, wait-learning was perceived to be more compatible with the notion of casual, informal learning, due to its transient user interface, opportune timing, and low-effort mechanics. Overall, wait-learning reminders were considered significantly less annoying, and were preferred over traditional reminders. These findings provide evidence of the real-world productivity benefits of wait-learning, as well as its impact on existing activites.

Chapter 7

Chain Reactions

While microtasks like flashcards require limited time to complete, in practice multiple microtasks are often completed one after another as part of a longer *microtask chain* within a single session. For example, we saw evidence in the previous studies that language learners voluntarily fetched chains of flashcards during micro-moments. On crowd platforms, batches of tasks done in chains are not only prevalent, but are also preferred by crowd workers because they leverage a worker's growing familiarity with the task [37]. Furthermore, microtask chains can potentially be designed to help a person learn more complex concepts, by using easy microtasks as a way of easing someone into more difficult ones.

The way that consecutive microtasks are chained affects the extent to which people complete tasks productively and continue to engage in the task at hand. Prior research suggests that task interruptions and delays can slow down performance [75, 92]. Boredom and fatigue from completing long chains of tasks can also lead to under-performance and task abandonment [39, 120]. However, past studies have demonstrated situations in which people continue doing tasks until a certain milestone has been reached, both on personal tasks [30] and crowd work [69].

A way to keep people engaged during a chain of microtasks is to order the tasks in a way that minimizes cognitive load, which could lead people to complete tasks more easily, more

efficiently and with greater enjoyment. The set of microtasks performed by an individual can vary: multiple operations may need to be performed on a single piece of content (e.g., describing and categorizing an item), multiple pieces of content may require the same operation (e.g., transcribing audio segments), or operations may have different complexity levels (e.g., performing easier vs. harder search tasks). For example, an email-organizing application might present consecutive emails from the same thread to preserve continuity of the content, or it might instead group on the same operation (e.g., rate all emails, then categorize all emails). In crowd work, microtasks are often routed to workers in different orders [92], leading to potentially diverse experiences within a single session.

To understand task chaining, we now shift from simple microtasks like flashcards, to more complex tasks in the writing domain. Not only is writing an important and common part of information work [64, 72], but it also offers a particularly interesting case for effective task chaining because subtasks in writing vary widely in both content and complexity, from low-level proofreading to meaning-rich rephrasing and tone modification. Moreover, writing is a canonically difficult task to start doing [56, 68], a hurdle that could potentially be addressed using microtasks.

We identify 11 common writing microtasks, and use crowd workers to evaluate the effect of chaining on microtask continuity (continuing microtask chains of the same complexity level), microtask transitions (transitioning across microtask complexity levels), and microtask ease-in (using simpler microtasks to ease people into more complex microtasks). Using operation, content, and complexity level as key properties in forming microtask chains, we find that microtasks have carry-over effects on subsequent microtasks within the same chain. Ordering affects 1) Continuity within the same complexity level: low-complexity microtasks chained on the same operation contribute to faster completion, while high-complexity microtasks chained on the same content are perceived to be less mentally demanding; 2) Transitions between complexity levels: microtasks are completed faster and are perceived to be better aided by preceding microtasks of the same complexity than by those of a different complexity; and 3) Easing in to complex tasks: people develop a sense of momentum and are faster to engage with a high-complexity microtask when it is preceded by lower-complexity

microtasks. As more and more tasks are transformed into microtasks, these findings, these findings provide insight into how microtasks can be chained to optimize transitions from one microtask to the next.

7.1 Research Questions

Our observations about the role of text content and task operation in easing transitions between low and high complexity tasks motivate the following research questions:

1. Continuity: How are performance and subjective experience of a microtask affected by whether preceding microtasks share the same operation or the same content as the current microtask?
2. Transitions: How are performance and subjective experience of a microtask affected by whether previous microtasks are of the same complexity or different complexity as the current microtask?
3. Easing in: How are performance and subjective experience of a complex microtask affected when it is preceded by simpler microtasks?

To address these questions, we begin by identifying a number of different writing microtasks that are of varying degrees of complexity. We then present the results of three crowd-based studies designed to explore these three questions using the microtasks we identified.

7.2 Selecting Writing Microtasks

The studies in this paper explore chains of microtasks. In each study, the microtasks may vary with respect to their:

1. Operation: There are a number of operations that could make up a writing microtask. We focus specifically on editing tasks that modify or build upon preexisting text.

2. Content: The preexisting text that the operations are applied to can vary. We identify a corpus of sentences for editing with similar style and reading level.
3. Complexity: Operations vary by complexity, depending on the level of involvement the task requires with the text and its meaning. We establish low, medium, and high-complexity groups of microtasks.

In this section, we describe how we selected the operations and content so as to control the amount of variation in each study, and how we measured microtask complexity.

7.3 Microtask Operations

For the studies presented in this paper, it was necessary to identify sets of easy and hard operations that were considerably different from each other in complexity, but similar within each set. We first selected a number of common operations (see Table 7.1) along basic rhetorical dimensions of writing: mechanics and semantics [127, 140]. Tasks in the mechanics category include checking for technical errors such as spelling and punctuation. In contrast, semantic tasks involve more in-depth consideration of meaning, such as shortening or rephrasing a sentence. We also included several tasks that are often implicitly executed by the writer, but explicitly completed in many crowdsourcing workflows: Awkcheck (identify whether a sentence sounds awkward), WordChoice (provide a better word to replace a given word), and SelectBest (select the best word to replace an existing word). These may be meaning-rich, but still fast to complete when supported by a workflow. For example, the Find-Fix-Verify [21] workflow could be analogously expressed using Awkcheck (Find), WordChoice (Fix), and SelectBest (Verify). After several iterations, we identified a list of eleven microtasks, shown in Table 7.1.

Note that the operations we selected are just a subsample of the potential operations. Our goal was not to be exhaustive, but rather identify an interesting range for study, with several operations at similar complexity levels. Pilot studies suggested these were a reasonable set to pursue.

7.4 Microtask Content

The operations we identified were designed to be performed on sentences. We gathered sentences from CHI 2015 paper abstracts to use in our studies. We chose this source because abstracts are prominent and complex, condensing a body of work into a single paragraph. We targeted sentences that were comprehensible on their own but required considerable mental processing to understand. Starting with 3701 sentences, we removed those with technical terminology that might make the sentence difficult to understand without external knowledge. Of the remaining 3115 sentences, we kept only those that were relatively complex by calculating the Automated Readability Index [136] and selecting 300 sentences at around the 75th percentile, with an Automated Readability Index between 18 and 23. Several example sentences are shown in Table 7.2.

7.5 Microtask Complexity

To identify the complexity of each operation applied to this content, we conducted a between-subject study on Amazon Mechanical Turk. We analyzed how microtasks compared along the following dimensions of complexity: semantic processing, time, mental demand, interest, meaningfulness, and open-endedness. We then used these results to select sets of high and low complexity microtasks for further study.

7.5.1 Complexity Metrics

We examine semantic processing, or the extent to which meaning is processed during a task, as a key measure of microtask complexity. This is based on evidence that semantic processing is associated with the depth and elaborateness of mental processing [37]. We also measure time spent on a microtask and perceived mental demand. Because complexity is sometimes associated with involvement on a task, we also include questions on interest

and meaningfulness, based on existing research on motivation and involvement [102, 156]. Lastly, we include a question on open-endedness because we found in pilot studies that it can be a key component of writing difficulties.

7.5.2 Method

Each participant was randomly assigned to an operation, and completed that microtask operation on a particular sentence. After the microtask, the participant answered subjective questions about the microtask, followed by a multiple-choice quiz about the meaning of the sentence. The subjective questions included the mental demand portion of the NASA-TLX [67], and Likert scale items about interest, meaningfulness, and open-endedness. The multiple-choice quiz was modeled after a common question on the TOEFL exam, which asks the person to select which of three sentences is most similar in meaning to the original sentence. We used accuracy on the quiz as a measure of semantic processing during the microtask. In addition to the eleven microtasks, we also included a baseline condition (quiz-only), where only the quiz was completed without any preceding microtask. The original sentence was hidden during the quiz for all aside from the quiz-only condition.

In early iterations, we found that objective measures on writing tasks, such as task completion time and semantic processing, fluctuate considerably depending on the nuances of the sentence, even if those sentences are of similar readability. To eliminate confounds resulting from sentence variations, in each study of this paper we hold the sentence constant on any microtasks being compared, but randomly sampled from the corpus otherwise. We restricted participants to those on Amazon Mechanical Turk residing in the United States with at least a 95% approval rating. Each person was compensated $0.30 and repeat participation was disallowed. To avoid selection bias, the same HIT preview page was displayed to all participants regardless of which condition they were placed in. 264 participated in the study. Given that semantics-level microtasks involve more in-depth consideration of meaning than mechanics-level microtasks, we expect these tasks to exhibit greater semantic processing (higher quiz performance), impose more mental demand, take longer time to complete, and

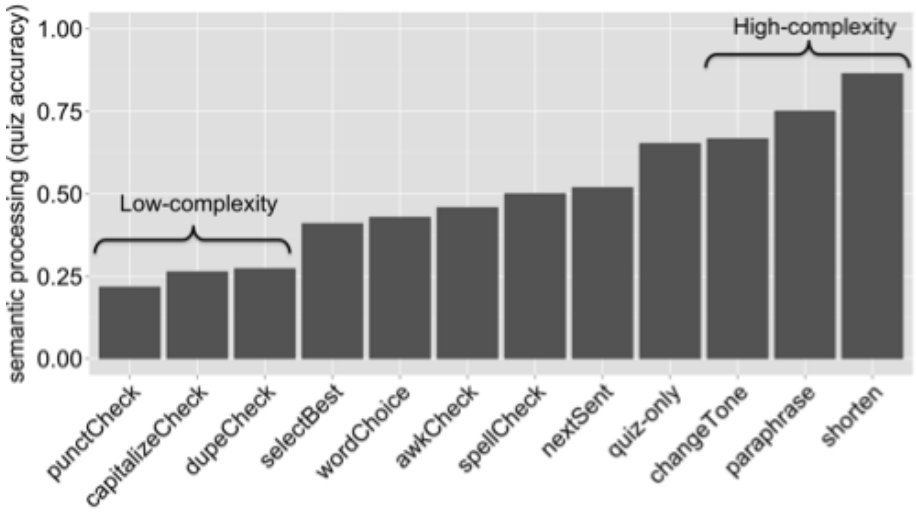


Figure 7-1: The level of semantic processing (quiz accuracy) by operation. In all conditions except the control (labeled quiz-only), the original sentence was hidden during the quiz. On average, 22 participants completed each operation. Based on these results, we selected low, medium, and high-complexity operations for further study.

be considered more open-ended than mechanics-level tasks.

7.5.3 Analysis

In all studies, we tracked any loss of browser window focus, and excluded timing data for those who were away for more than fifteen seconds. Extreme outliers were also removed. For any timing data that was not normally distributed, but was log-normal, we applied a log-transformation before running significance tests. Accuracy metrics (e.g., quiz performance) were evaluated using a logistic regression. For Likert scale items, we report on ANOVA results, but non-parametric tests yielded empirically similar results.

7.5.4 Results

Using this approach, we were able to identify microtasks of different complexities, as shown in Figure 7-1. The task of shortening a sentence (shorten) led to the best performance on

the quiz, whereas the task of checking for punctuation error (punctCheck) led to the worst performance. Interestingly, mechanics tasks led to below random quiz performance. Because the wrong quiz answers typically contained words from the original sentence, whereas the correct answer modified words while kept the original meaning intact, it is possible that those in the mechanics conditions simply guessed by selecting answers containing words they had seen.

As expected, we found significant differences between all pairs of mechanics microtasks and semantics microtasks ($p < 0.05$), with the exception of spellCheck and nextSent. Semantics microtasks had significantly higher semantic processing and longer task time, were perceived to be more open-ended and imposed greater mental demand than mechanics microtasks. SpellCheck and nextSent did not align well with these categories, possibly because spellcheck sometimes activates a moderate level of semantic processing [62], and nextSent may be open-ended enough to be completed without fully understanding the original sentence. No differences were found within each set of mechanics tasks and semantics tasks.

In analyzing the crowd-based microtasks (awkCheck, wordChoice, selectBest), we find that even though all three microtasks performed similarly on semantic processing, wordChoice took longer to complete, likely because it involved generating new content. Excluding wordChoice (as well as spellCheck and nextSent for reasons stated above), we compared how the crowd-based microtasks awkCheck, selectBest compared to mechanics and semantics operations. While awkCheck and selectBest had lower semantic processing, mental demand, and completion time than semantics microtasks (all $p < 0.05$), they were also more interesting and open-ended than mechanics microtasks (all $p < 0.05$), and more meaningful than mechanics with marginal significance ($p = 0.06$) (Figure 7-2).

Based on the results, we chose a subset of these operations for further study. Our goal was to select sets of operations at opposite ends of the complexity spectrum, such that the sets are substantially different in complexity, but with similar enough operations within each set to be interchangeably used in our studies. We thus select capitalizeCheck, punctuationCheck,

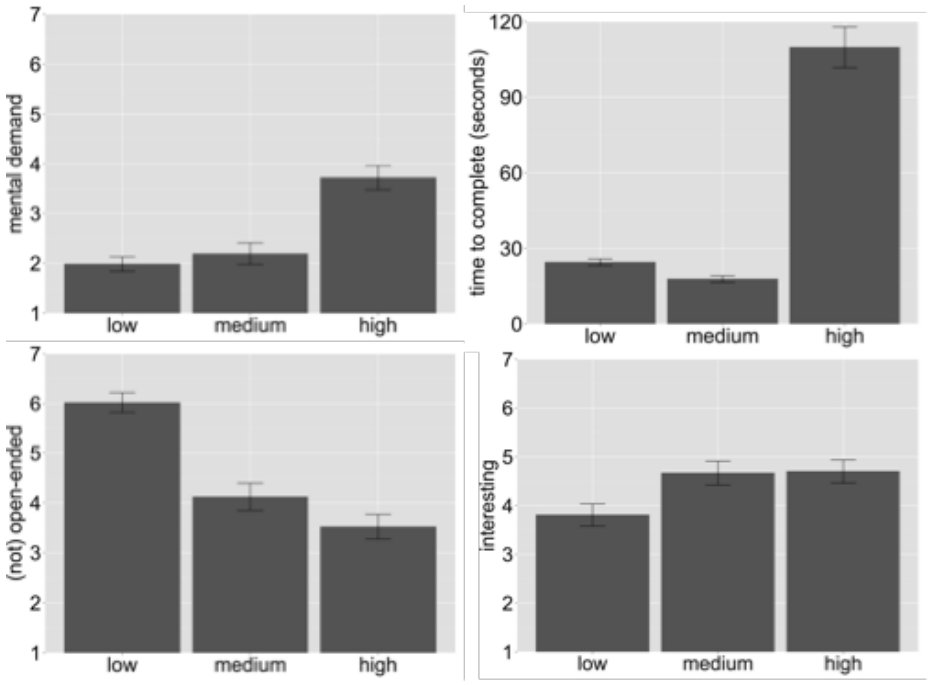


Figure 7-2: Medium-complexity microtasks are similar to low-complexity microtasks on mental demand, task time, and semantic processing, but similar to high-complexity microtasks on open-endedness, interest, and meaningfulness.

duplicateCheck to be low-complexity microtasks (L), and paraphrase, toneChange, shorten to be high-complexity microtasks (H). We also select awk-Check, selectBest to be medium-complexity microtasks (M), since they are more interesting than low-complexity tasks, but easier than high-complexity tasks.

7.6 Microtask Continuity

Using the microtasks identified in the previous section, we evaluate how ordering affects microtask chains through a series of three studies, spaced across several weeks. We first assess microtask continuity: how task chaining affects a chain of microtasks with the same complexity. We asked participants to perform a series of microtasks, and evaluated their experience on the final microtask. We varied whether preceding microtasks had shared the same operation or the same content as the final microtask. In addition, we examined whether

these effects depend on the complexity-level of the microtask.

7.6.1 Method

The study followed a 2 (complexity) x 2 (chain type) between-subject design, where complexity was either high (H) or low (L), and chain type was either same-operation or same-content. In all conditions, the chain consisted of three consecutive microtasks. In the same-operation conditions, participants did the same task (e.g., paraphrase, paraphrase, paraphrase) on three different sentences. In same-content conditions, participants did three different tasks (e.g., changeTone, shorten, paraphrase) on one single sentence. We used three tasks per chain because we had three operations per complexity level to work with. The study used a between-subject design because our goal was specifically to evaluate the effects of one microtask on the next, which could be potentially confounded in a within-subject design containing consecutive conditions.

In all microtasks aside from the final one in each chain, sentences were randomly sampled from the sentence corpus, and the order of sentences and task types was randomized for each participant. For the purpose of comparing performance on the final task, the sentence and task type of the final task was held constant. The final H task in a chain of high-complexity tasks was always paraphrase, and the final L task in a chain of low-complexity tasks was always duplicateCheck. We chose these particular microtasks because, compared to other microtasks of the same complexity, they demonstrated the most consistent performance across the complexity metrics previously described. Participants were compensated $1.00 for the study. 183 people completed the study.

Measures

We used measures of time, quality, mental demand, helpfulness and enjoyment to understand continuity.

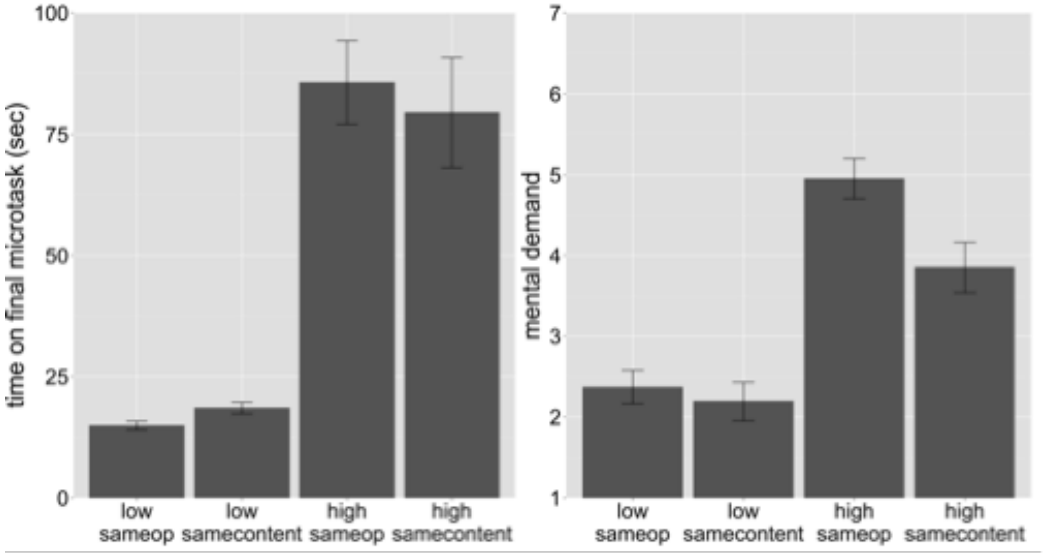


Figure 7-3: In low-complexity chains, the final microtask was completed faster in the same-operation condition. In high-complexity chains, participants found the final microtask less mentally demanding in the same-content condition.

- *Time.* We measured time spent on the final task.
- *Quality.* In L chains, we computed whether the participant correctly fixed the mechanics error on the final task. In H chains, the sentences produced by participants on the final task were each rated on a 5-point Likert scale by three different workers on Mechanical Turk.
- *Mental demand.* Participants completed the mental demand portion of the NASA TLX about the final task.
- *Helpfulness.* Participants rated on a 7-point Likert scale to what extent the first two tasks helped them do the final task.
- *Enjoyment.* Participants rated on a 7-point Likert scale to what extent they enjoyed the full microtask chains.

Because L microtasks are easier and encourage speed, we expected these microtasks to benefit more from same-operation chains which enable people to perform a series of similar microtasks in a row. Conversely, because H tasks are more cognitive and semantically rich, we hypothesized they would benefit from same-content chains, which allow people to focus and build on similar content across multiple microtasks.

7.6.2 Results

Overall, we found that tasks were affected by the preceding tasks, and that complexity-level was an important factor in mediating the effects of chain type. Low-complexity microtasks took less time when preceded by same-operation microtasks, but high-complexity microtasks were perceived to be less demanding when preceded by same-content microtasks (Figure 7-3). Specifically, we found a significant interaction effect between complexity and chain type on task completion time (F(1,149)=6.6, $p < 0.05$). In low-complexity chains, participants completed the final microtask faster when it was preceded by same-operation tasks ($\mu = 14.91$ sec) than by same-content tasks ($\mu = 18.49$ sec), a difference of 3.58 seconds ($p < 0.05$). However, no significant time difference was found on high-complexity conditions. In addition, no difference in quality was found.

In analyzing mental demand, we observed a marginally significant interaction effect between complexity and chain type (F(1,179)=3.35, $p = 0.06$) (Figure 7-3). We therefore examined high-complexity and low-complexity conditions separately. On high-complexity chains, those in the same-content condition found the final task significantly less mentally demanding ($\mu = 3.85$) than those in the same-operation condition ($\mu = 4.95$, $p < 0.01$). This suggests that the semantic meaning extracted during a H task builds up a mental state [25] about the sentence that is then utilized on subsequent tasks requiring access to the same state.

Lastly, we found that initial microtasks were perceived to be significantly more helpful to the final microtask in high-complexity chains ($\mu = 5.21$) than in low-complexity chains ($\mu = 4.46$, F(1,179)=9.09, $p < 0.005$). However, overall enjoyment was greater on low-complexity chains ($\mu = 5.98$) than high-complexity chains ($\mu = 5.40$, F(1,179)=7.43, $p < 0.05$). This is consistent with our earlier finding that low-complexity microtasks can be completed without as much scaffolding or semantic processing.

In summary, low-complexity microtasks were completed faster when preceded by same-operation microtasks. However, high-complexity microtasks felt less mentally demanding

when preceded by same-content microtasks.

7.7 Microtask Transitions

In the previous study we found that a person's experience with a microtask was affected by whether preceding tasks were of the same *operation* or the same *content*, given a constant complexity-level. However, microtask complexity might also vary within a chain of tasks. In writing, one might perform diverse modifications on the same section of content, with some that are more cognitively complex than others. In this study, we investigate the effects of transitioning between microtasks of the *same* complexity or *different* complexity, when the content is the same.

7.7.1 Method

The study followed a 2 (complexity) x 3 (chain type) between-subject design, with all microtasks performed on the same content. The complexity of the final microtask was either high or low, and the chain type was either same-complexity, different-complexity, or a control which had no microtasks before the final microtask. Specifically, the six conditions were: LLL (same-complexity), HHL (different-complexity), L-only (control), HHH (same-complexity), LLH (different-complexity), and H-only (control). We included L-only and H-only as control conditions to understand how performing a microtask with no preceding tasks compares to one that transitions from other microtasks, since these complexity shifts could impose a mental switch cost [153].

Similar to the previous study, microtask order was randomized for all except for the final microtask. The same content was held constant throughout the chains, and each participant did three different operations within each chain, where the last H microtask was paraphrase and last L microtask was duplicateCheck. The measures were the same as those described under the Microtask Continuity section, and participants were compensated $1.00 for the

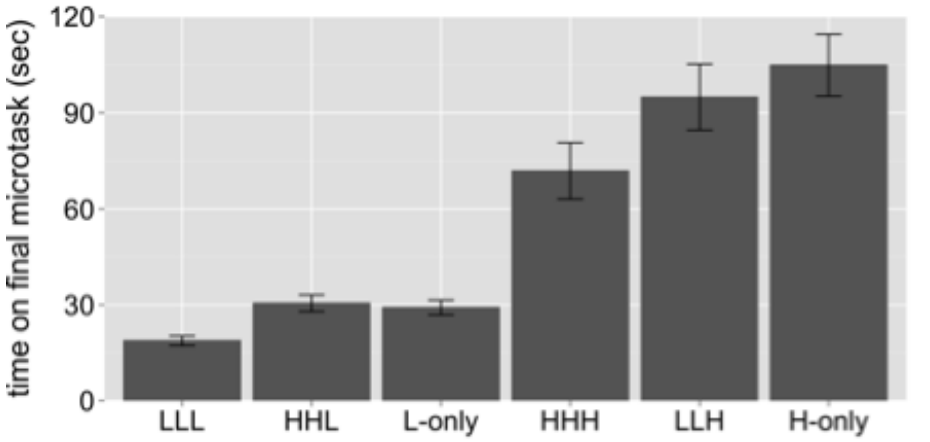


Figure 7-4: Same-complexity microtasks led to significantly faster completion time on the final microtask, compared to different-complexity microtasks and control conditions.

study. A total of 173 people participated in the experiment.

Since L microtasks demand relatively little semantic processing, we do not expect performance to be affected by whether the task is preceded by H tasks (HHL) or L tasks (LLL). Furthermore, if transitioning from H to L imposes an additional switch cost, HHL might perform worse than L-only. In contrast, because H tasks are more cognitive in nature and require semantic processing, we expect lead-up H tasks (HHH) to serve an advantage over lead-up L tasks (LLH). If transitioning from L to H imposes an additional switch cost, LLH might also perform worse than H-only.

7.7.2 Results

Overall, we found that the complexity of initial microtasks mattered, and their helpfulness was also perceived differently depending on the complexity of the final microtask. Same-complexity chains led to faster completion times on the final microtask ($\mu = 46.46$ sec), compared to control ($\mu = 64.60$ sec) and different-complexity ($\mu = 61.47$ sec) chains (Figure 7-4). A two-way ANOVA found a significant main effect of chain type on task time (F(2,139)=12.09, $p < 0.005$), and post-hoc Tukey tests found the completion time for same-complexity chains to be significantly faster than different-complexity and control

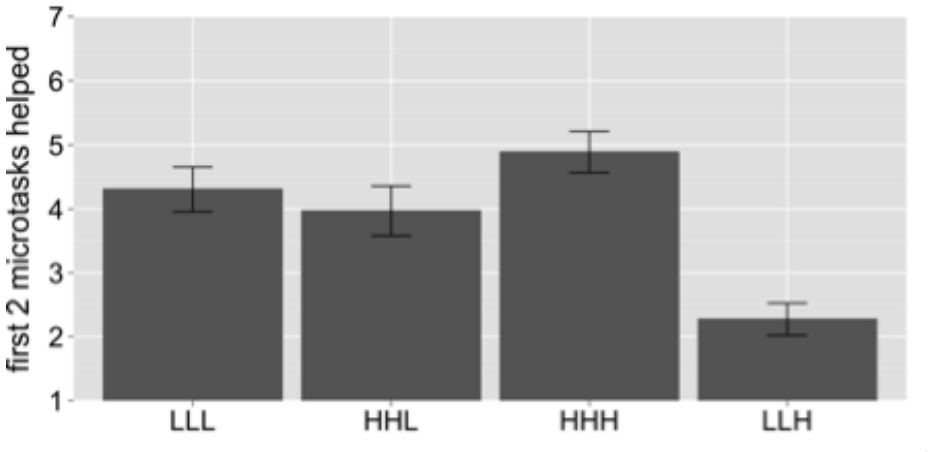


Figure 7-5: H microtasks were perceived to be significantly more helpful than L microtasks for completing final H microtask when chaining across complexity.

($p < 0.05$). No differences in quality were found.

Furthermore, we saw a significant interaction effect of complexity and chain type on perceived helpfulness (F(1,116)=11.49, $p < 0.0005$) (Figure 7-5). For chains ending in a high-complexity microtask, those in the same-complexity condition (HHH) found the lead-up tasks to be significantly more helpful (mean=4.89) than those in the different-complexity condition (LLH) (mean=2.27, $p < 0.0005$). However, no difference was found between chains ending on a low-complexity microtask. Hence, for chains ending on complex microtasks, same-complexity chains were perceived to offer a cognitive benefit over different-complexity chains.

Lastly, H-only tasks were significantly less enjoyable ($\mu = 4.08$) than L-only tasks ($\mu = 5.71$, $p < 0.005$). Even though H tasks may be valuable to future H tasks, they demand an upfront ramp-up of meaning and focus that demands effort. On average, those in the H-only condition spent 33 seconds longer completing the task than those who first completed other H tasks (HHH condition). The equivalent time lost was only 11 seconds in the L-only condition.

We found no evidence that transitioning between complexities dampens performance compared to starting off immediately with the final microtask. No significant difference was

found on either time or quality between different-complexity and control conditions. We hypothesize that starting immediately with the final microtask is itself a task switch, since it still demands mentally transitioning from the user's previous activity.

In summary, same-complexity microtasks led to faster completion times and were perceived to be more helpful when the final microtask was complex. Low-complexity microtasks were completed slower when preceded by high-complexity microtasks than when preceded by low-complexity microtasks, possibly due to a depletion of mental resources [135]. However, we found no evidence that complexity-switching is worse than starting immediately on the microtask, possibly because the latter still incurs a switch cost. Completing an initial high-complexity microtask may also be particularly arduous, even if that effort pays off on future high-complexity tasks. These findings raise the question of how microtasks could be better chained to ease people into meaning-rich microtasks.

7.8 Microtask Ease In

In the previous study, we found that initial H tasks were valuable to the completion of further H tasks of similar content, but they were also cognitively demanding and less enjoyable to do compared to L tasks. In this study, we examine whether performing lower-complexity microtasks can help people start a high-complexity microtask sooner, while simultaneously accomplishing a unit of work itself.

7.8.1 Method

To understand the impact of leading up with various kinds of simpler microtasks, we conducted a between-subject study with a 2 (complexity) x 2 (content) design, where the complexity of the first two microtasks was either low (L) or medium (M), and the content was either same or different from the final microtask. We also include a H-only condition with no lead-up microtasks to serve as a control. All conditions ended on a high-complexity

microtask. We include M microtasks as a potentially interesting class of lead-up tasks because they were perceived to be more interesting and open-ended than L microtasks, yet still faster and easier to complete than H microtasks (see Selecting Writing Microtasks). We also compare same-content to different-content lead-up tasks because same-content was previously found to help build context toward a high-complexity task (see Microtask Continuity).

The four conditions were: LLH_same, MMH_same, LLH_diff, MMH_diff, and H-only. Similar to previous studies in this paper, the final (H) microtask was a paraphrase task across conditions, the order of the first two task operations was randomized, and the first two sentences were also randomly sampled in the different-content conditions. Participants were compensated $0.50 for the study. 201 people participated in the study.

Measures

As with the previous studies we used measures of time, quality, and mental demand. We also looked at the amount of time it took to start a task and measures of participant-reported momentum and warm-up.

- *Time, Quality, Mental Demand* on the final task, as before.
- *FirstTypeTime.* We measured the duration between the final task appearing and the person starting to type.
- *Momentum.* Participants rated on a 7-point Likert scale to what extent they felt momentum going into the final task.
- *Warm-up.* Participants rated on a 7-point Likert scale how mentally warmed up they felt going into the final task.

To prevent the questionnaire itself from confounding the experience of completing a task chain, all Likert-scale questions were asked after the final task.

Because tasks chained on the same content could help build awareness of meaning, we hypothesize same-content lead-up tasks to yield stronger benefits than different-content

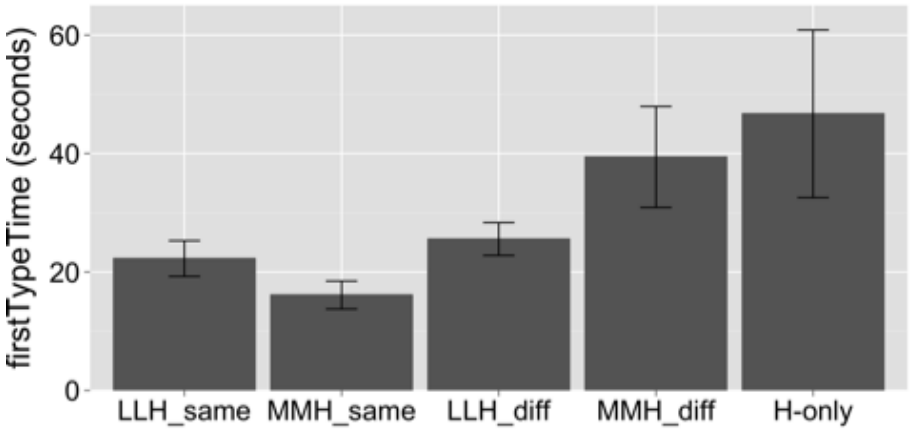


Figure 7-6: Typing started significantly sooner when an H microtask was preceded by M microtasks on the same content.

lead-up tasks. M lead-up tasks may be more helpful than L lead-up tasks, due to greater meaning extracted.

7.8.2 Results

Lead-up microtasks had an effect on the final microtask, and these effects varied depending on their properties. Participants initiated typing sooner in the MMH_same condition ($\mu = 16.11$ sec) than the H-only condition ($\mu = 46.71$ sec), a difference of 30.6 seconds. We found a significant effect of condition on FirstTypeTime ($F(4,195)=5.4$, $p<0.0005$), and a Tukey's post-hoc test found the FirstTypeTime of MMH_same to be significantly faster than that of H-only. While typing does not necessarily mean faster engagement, it does suggest they were able to start transforming thoughts into text sooner, possibly due to context gained from the lead-up tasks. Whereas 43% of MMH_same participants started typing within 10 seconds of the H task being displayed, only 11% started typing in the H-only condition, and 38% in the LLH_same condition. The difference between LLH_same and H-only was marginally significant after post-hoc correction ($p = 0.08$). No difference in quality was found.

Additionally, participants in the LLH_same condition reported a significantly greater sense of

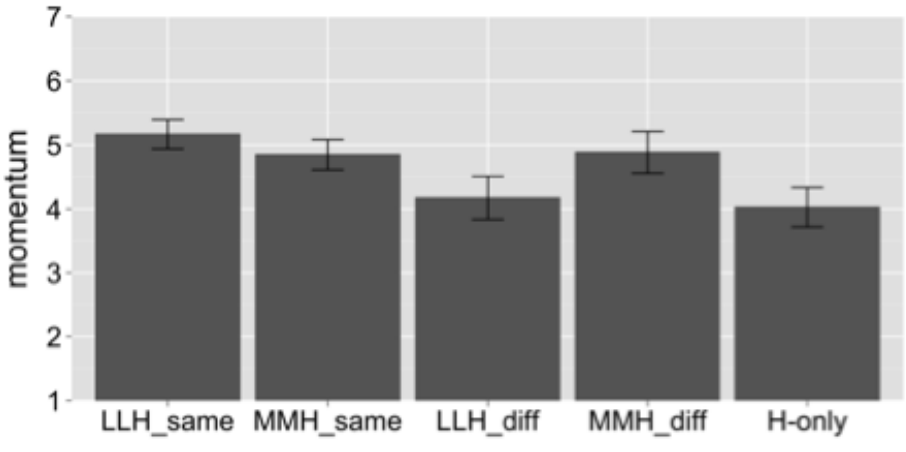


Figure 7-7: Participants felt they had significantly greater momentum going into an H microtask when it was first preceded by L microtasks on the same content.

momentum going into the final task ($\mu = 5.17$), compared to H-only participants ($\mu = 4.03$, p<0.05) (Figure 7-6). The concrete, self-contained nature of low-complexity tasks may have helped build initial rhythm. Lastly, users in the LLH_same condition felt significantly more mentally warmed up ($\mu = 5.25$) than those in the LLH_diff condition ($\mu = 3.87$, $p < 0.05$), suggesting that some awareness of content could be developed even during low-level mechanics tasks. However, given that participants did not perceive L microtasks to help with H microtasks in the prior study (see Microtask Transition results), these benefits may be subtle, and may be less obvious to the average user.

Although same-content lead-up tasks contributed to greater momentum and earlier action taken, we found no significant difference in task completion time on the final H task. Compared to L microtasks, the open-ended nature of H microtasks may mean that task time is not only attributed to efficiency, but also immersion in the task. Our post-hoc evaluations found that those in the MMH_same condition made a relatively large number of deletions during the H task ($\mu = 5.9$), suggesting a certain level of involvement (Figure 7-8) that may contribute to longer task times.

In summary, high-complexity microtasks were initiated sooner when preceded by medium-complexity microtasks, and were perceived to have greater momentum when pre-ceded by low-complexity microtasks. We observed these effects only in same-content chains,

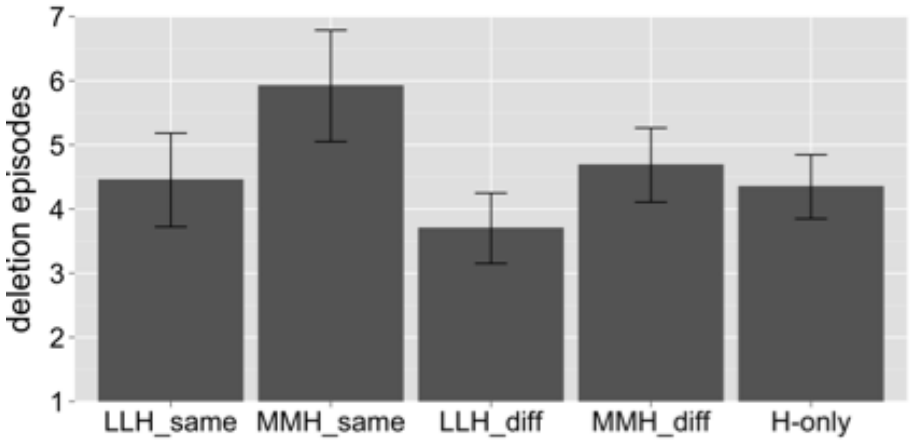


Figure 7-8: After performing M microtasks on the same content, participants made many deletes during the H microtask.

suggesting that some content awareness is developed even on low-complexity microtasks. These findings are consistent with ease-in strategies described by participants in our pilot interviews, and lend support to prior work that shows concrete, short-term goals can help people get started on larger tasks [1, 4].

7.9 Design Implications

We have seen that microtasks have carryover effects on subsequent microtasks, both in chains that continued with the same complexity and that transitioned across complexities. Table 7.3 summarizes our results. Given that the overall time taken to complete a chain of tasks was short (median=132 seconds), the ordering effects we observed may have far-reaching cumulative effects. In this section, we discuss how our findings can be used by researchers and practitioners to improve microtask chains for writing.

7.9.1 Build momentum using same-operation L microtasks.

At the start of a microtask chain, when users may be less focused, performing a series of low-complexity microtasks on the same operation could help build speed and rhythm. On low-complexity tasks, we found that same-operation chains led to higher efficiency compared to same-content chains (see Microtask Continuity). Preserving the same type of operation while switching on content may be particularly apt at the start of a chain, a time when users have relatively little context and are still acquainting to the task.

7.9.2 Transition using L and M microtasks on similar content.

As high-complexity microtasks may be painstaking to do at the start of a chain (see Microtask Transition), low-complexity microtasks could help build momentum and awareness of meaning, while simultaneously completing a unit of work. In our study (see Microtask Ease-In), M microtasks enabled people to start typing sooner on a subsequent H microtask, and L microtasks were perceived to build momentum, but only in cases where content was similar to that of the final H task.

7.9.3 Avoid switching content during a series of H microtasks.

Because the rich meaning extracted during a H task builds up semantic understanding about the content, interleaving content during H tasks may disrupt the mental state [25] associated with the task. Instead, staying on the same content could enhance experience on subsequent H tasks requiring access to the same state, building continuity in the task at hand. Participants rated H microtasks less mentally demanding when preceded by same-content microtasks than when preceded by same-operation microtasks (see Microtask Continuity). In crowd work, where several operations may be needed for multiple pieces of content, presenting similar content consecutively (e.g., adjacent audio segments) could help preserve the context gained in earlier tasks. In personal tasks, organizing tasks by region of content (e.g., email

thread or lecture topic) could alleviate cognitive effort amidst competing priorities.

7.9.4 Consider switching content on transitions from H to L.

Although transitions into H microtasks are supported by tasks with similar content, transitions into L microtasks demand less context and awareness of meaning. In fact, performance was slower when low-complexity tasks were preceded by high-complexity tasks on the same content, than when they were preceded by low-complexity tasks on the same content (see Microtask Transition). After completing a meaning-rich H task, shifting the content could potentially give a renewed perspective or "fresh eyes" to a low-level task such as proofreading, though without further investigation, this implication remains preliminary.

7.10 Discussion

Our findings demonstrate the substantial impact that small changes in microtask ordering could have on users. While same-operation chains aid in the efficiency of simple microtasks, same-content chains may help alleviate mental burden on more complex microtasks. Ordering low-level microtasks first may help people start meaning-rich tasks on the same content.

These transitions pave the way to richer forms of wait-learning. For example, micro-learning tasks could be strategically re-ordered in a way so that difficult concepts can be gradually uncovered during longer periods of wait time. For example, a microtask chain could start with a few vocabulary translation exercises (same operation) to engage the learner, then gradually ease into phrase-level translations by re-using some of those words in phrases (same content), once the learner has become more immersed. A wait-learning system might also take into consideration a user's cumulative experience *across* consecutive microtask chains if they are close together in time. For instance, a seemingly complex concept could be made easier if it builds on a simpler concept they have recently seen.

Although our studies focused on the domain of writing and editing, the general classes of measures that were investigated (operation, content, and complexity) can be found in diverse domains, and provide a baseline for future work in micro-productivity. For example, task chains are also common in systems for rehabilitation [117] and micro-volunteer work [147]. As more tasks are transformed into micro-components, systems should consider the user's cumulative experience and memory when ordering microtasks.

Given that individuals often engage in simple tasks as a way of starting more complex tasks, effective chaining could aid not only transitions between microtasks, but also transitions into larger tasks. One could imagine leveraging different ease-in chains for different purposes. For example, since medium-complexity microtasks led to faster action initiation, they could be used in mobile scenarios, when a user may otherwise be unmotivated to start a meaning-rich task. Alternatively, low-complexity microtasks can help build initial momentum in a desktop scenario, where the user simply needs support re-gaining focus.

In a complex system, microtask ordering may be subject to interdependencies and global constraints. Our findings could be combined with those additional constraints to inform which microtask ought to be presented next, given a history of recently completed microtasks and their properties. Although we used sentence-level units, the microtask properties described could be applied to longer text, such as keeping same-content chains within the same region of a document. We leave document-level tasks to future work.

Lastly, the task chains in our studies were relatively short, and were evaluated on text that was not originally authored by the participants. We held constant the chain length and text across conditions in order to study ordering effects alone. Future work should investigate ordering effects in longer chains, and explore situations that leverage an individual's long-term familiarity with the content.

7.11 Conclusion

In this chapter, we showed that microtasks have non-negligible effects on other microtasks within the same chain, and demonstrated how key properties of a microtask (operation, content, and complexity) mediate the efficiency, mental workload, and activation of future microtasks. Through a series of studies focusing on writing, we found that small changes in task ordering can significantly affect microtask continuity, transitions, and ease-in. Taken together, these findings have important implications for designing effective microtask chains.

	Operation	Example Prompt
Mechanics	punctuationCheck *	Fix the punctuation error in this sentence, if one exists.
	duplicateCheck *	Fix the spot that has the same word twice in a row, if one exists.
	capitalizationCheck *	Fix the capitalization error in this sentence, if one exists.
	spellCheck	Fix the spelling error in this sentence, if one exists.
Crowd	awkCheck *	How awkward does this sentence sound? (Rate on 5-point scale)
	selectBest *	Select the best re-wording of this sentence. The word to replace is in brackets.'
	wordChoice	Replace the word in brackets to improve this sentence.
Semantics	changeTone *	Rewrite this sentence in an emotional tone.
	paraphrase *	Paraphrase this sentence as if you were saying it to a five-year-old.
	shorten *	Without changing it meaning, shorten this sentence so that it is at most {2/3 of original length} words long.
	nextSent	Write a sentence that could make sense if it came after this sentence.

Table 7.1: Eleven microtasks commonly performed in writing. After we analyzed the complexity of these microtasks, those with asterisks were selected for final studies.

Example sentences from CHI 2015 paper abstracts
Selective exposure, the preferential seeking of confirmatory information, can potentially exacerbate fragmentation of online opinions and lead to biased decisions.
This paper describes a study of algorithmic living with Trace, a mobile mapping application that generates walking routes based on digital sketches people create and annotate without a map.
We found that collective action publics tread a precariously narrow path between the twin perils of stalling and friction, balancing with each step between losing momentum and flaring into acrimony.

Table 7.2: Examples from the 300-sentence corpus used in our studies. Sentences in the corpus are at the 75th percentile of reading difficulty among CHI 2015 paper abstracts, determined using the Automated Readability Index.

	SUMMARY OF FINDINGS
Microtask Continuity	**RQ 1:** When chains have the same complexity across microtasks, what is the effect of continuing on the same *operation* vs. same *content*?
	- In low-complexity chains, same-operation resulted in faster completion time on the final microtask than same-content.
	- In high-complexity chains, same-content resulted in lower mental demand on the final microtask than same-operation.
Microtask Transitions	**RQ 2:** When chains use the same content across microtasks, what is the effect of transitioning from microtasks of the *same* complexity vs. *different* complexity?
	- Same-complexity resulted in faster time on the final microtask than different-complexity and control conditions.
	- Same-complexity was perceived to be more helpful than different-complexity to a final high-complexity microtask.
	- Starting directly with a high-complexity microtask was less enjoyable than starting directly with a low-complexity microtask.
Microtask Ease-In	**RQ 3:** How is a person's experience on a *complex* microtask affected when it is preceded by *simpler* microtasks?
	- People started typing sooner if they first completed medium-complexity microtasks on the same content.
	- People perceived greater momentum if they first completed low-complexity microtasks on the same content.
	- People felt more mentally warmed up if they first completed low-complexity microtasks on the same content, compared to low-complexity microtasks on different content.

Table 7.3: Summary of findings from studies on Microtask Continuity, Microtask Transitions, and Microtask Ease In.

Chapter 8

Deadspace Finder

In previous chapters, we focused primarily on the *timing* aspect of microtasks, because timing is closely tied to a person's mental and physical availability. However, as we have seen in WaitSuite, *space* also affects the noticeability and disruptiveness of microtasks, and can often complement timing. In contrast to popups, which animate into view and occlude existing tasks, WaitSuite displayed flashcard microtasks within nearbly static panels, so that they were simultaneously within reach and effortless to ignore. Yet, embedding these non-occlusive panels within each WaitSuite system required substantial customization and knowledge about the existing interface.

Automating this spatial embedding process could substantially reduce the level of application-specific customization required, and scale up opportunities for microproductivity. On graphical user interfaces, however, it is challenging to automate such a process for several reasons. First, limited screen space and competing tasks make it difficult to find new screen space that is both within the user's field of view and non-disruptive to ongoing user activity. Furthermore, there are a wide diversity of applications and unexpected changes that could occur in an application after content is placed. In contrast, peripheral interfaces are already prevalent in the physical world: For example, in the physical world, vocabulary flashcards left on one's desk can be glanced at and picked up if desired, but otherwise do not break a user's flow. In the digital world, ambient flatscreen displays showing a weather report in hallways

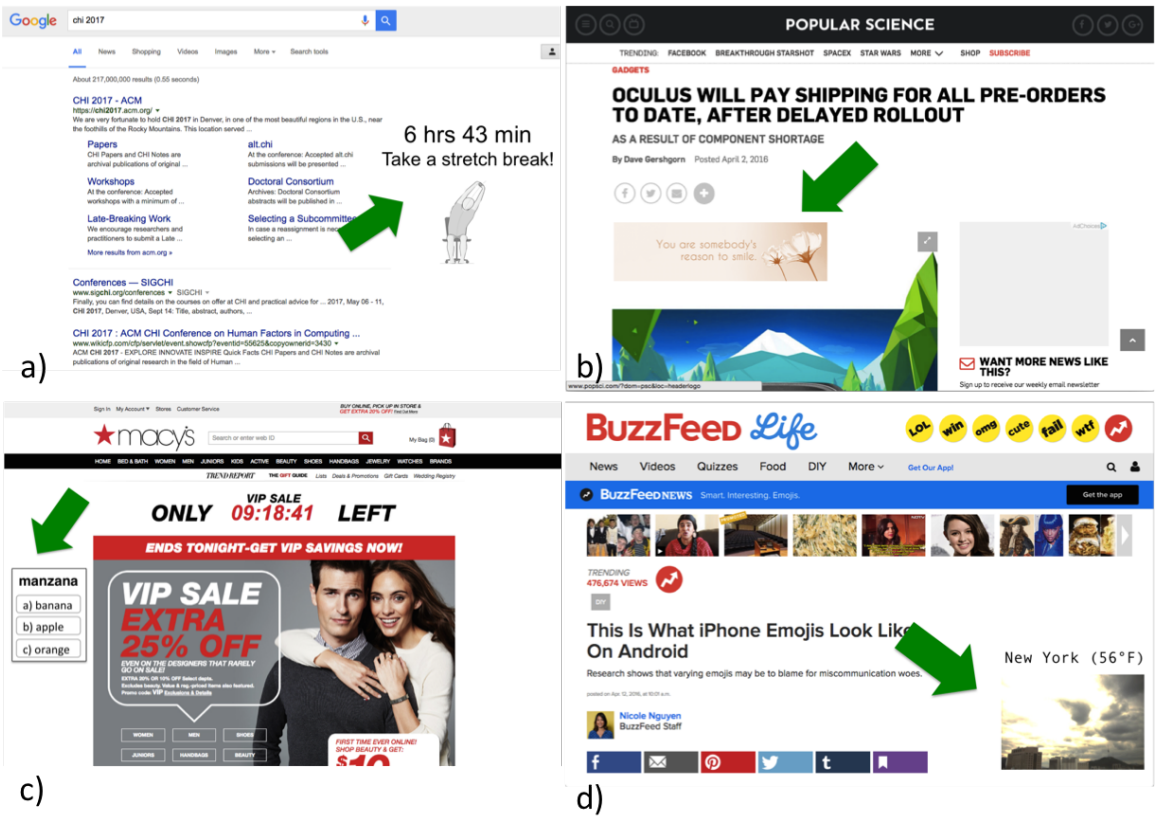


Figure 8-1: Examples of Deadspace Finder applications for purposes such as: a) computer break reminders, b) affective interfaces (e.g. an uplifting message), c) vocabulary flashcards, and d) ambient awareness of outside weather.

passively draw attention without interfering with existing tasks. Although such peripheral interfaces have been used for a wide range of applications intended to provide awareness of non-critical information [79, 110, 130], it remains difficult to do so on graphical user interfaces due to limited screen space, competing tasks, and lack of automation. In online advertising, advertisers are able to embed content of various formats into the extra space on webpages [27, 32], but these spaces are pre-determined and populated by webpage owners and ad providers. Users thus have little control over the content they see.

In this chapter, we introduce a novel approach for automatically finding and using the existing *deadspace* in any webpage a user is browsing for the passive display of secondary content. We created *Deadspace Finder*, a browser-based system that automatically detects deadspace, places content into deadspace while the user is focused elsewhere, and automatically removes

the content if it collides as a result of the page changing. Rather than popping up or animating into appearance, content is instead subtly placed in the unused space of backgrounded or idle tabs, so that it is already present by the time the user encounters it, and so that it does not create any detectable movement in the user's visual field. In this way, deadspace content can invite the user to engage if it happens to be noticed, but can otherwise be ignored without additional action.

Deadspace Finder puts greater control in the hands of users, by offering an automatic technique for non-occlusively embedding user-desired content into any webpage via a browser plugin. For example, in a typical day, it may be desirable to receive triggers that help us maintain better posture, practice vocabulary flashcards, or stay aware of the outside weather. Currently, users must either install standalone applications, which demand a substantial overhead to switch to, or use browser plugins that are not customizeable and tailored to specific webpages. Deadspace Finder could enable any microtask to be automatically embedded into any webpage with unused space, giving users the freedom to tailor the placement of these tasks to their own desired contexts (Figure 8-1).

We evaluated Deadspace Finder in two parts. First, we sought to measure whether Deadspace Finder can accurately detect deadspace, and how frequently deadspace occurs. Second, it is important to understand the dynamics of Deadspace Finder during real web browsing, and how it is perceived by end users. To answer these questions, we conducted an evaluation of 500 webpages, as well as a 10-day field deployment in which participants saw images delivered in both deadspace and popups during regular computer usage. We found that at least 80% of webpages contain deadspace large enough to fit 125x125 pixels, and that Deadspace Finder can accurately identify this deadspace and resolve collisions on-the-fly. Deadspace content was also perceived to be less disruptive than popup content, while still being noticed 73% of the times it appeared. However, some participants found deadspace content hard to ignore if it was too visually salient. These findings shed light on how deadspace can be automatically detected and used for secondary content.

The main contributions of this chapter are:

- The idea of using *deadspace* for the display of occlusion-free secondary content.
- Deadspace Finder, an approach for automatically discovering deadspace, placing content, and resolving collisions.
- An evaluation of Deadspace Finder on a corpus of 500 webpages, as well as a field study with 20 users, showing that deadspace is feasible and effective at delivering secondary content.

8.1 User Interface

Information systems can provide information at different *notification levels* (ignore, change blind, make aware, interrupt, or demand attention) using different display transitions [101, 126]. In contrast to systems that *interrupt* or *demand attention* from users, the vast majority of peripheral and ambient interfaces aim to provide information less obtrusively, in the *change blind* or *make aware* categories. Along these lines, Deadspace Finder seeks to minimize disruption by changing the display subtly. Rather than blinking, popping up, or animating into appearance, deadspace content is subtly placed into backgrounded or idle tabs while the user's attention is elsewhere, so that the content is already present by the time the user encounters it. Because the change does not occur while the user is watching, the user may not notice the change (change blind), or they may notice the change or eventually stumble upon the new content (make aware).

8.1.1 Example Applications

Figure 8-1 shows several examples of deadspace content serving various purposes: **a) Computer break reminder**: Users see the amount of time they have spent on the computer, along with a picture of a stretch they could do to take a break. **b) Affective interfaces**: Users see an uplifting image or quote during the day, which could elevate one's mood or trigger reflection. **c) Micro-learning**: At intermittent intervals during the day, users are presented an interface where they can answer a flashcard. **d) Ambient awareness**: During

the work day, users encounter a picture of the current weather outside, providing awareness of spaces beyond their current location.

8.1.2 Dynamic Behavior

Given a particular piece of content that needs to be displayed, Deadspace Finder embeds it into tabs that are currently backgrounded (not in view), to potentially be seen when the user switches tabs. If the computer becomes idle, Deadspace Finder inserts the content into the front-most tab to be encountered when the user returns. Since it is possible for a tab to be in view even while the computer is idle, the content fades in slowly over the course of 10 seconds, rather than appearing immediately. Hence, users can either encounter the content after switching to a tab, or when returning to their computer after it becomes idle.

For certain applications, a user may wish to receive content at a certain frequency, or within a certain time delay. Deadspace Finder triggers content to be inserted periodically, based on a default or user-customized *appearance interval*. Because it is uncertain which tab a user might switch to next, Deadspace Finder attempts to insert the content into any tabs that are currently backgrounded, but removes it from all other tabs once it has appeared.

A user's likelihood of encountering deadspace content depends on individual behaviors such as tab-switching or computer idling. In order to attain a particular appearance interval, Deadspace Finder must insert the content at a more frequent *insertion interval*. In early iterations, we found that the appearance interval is approximately double to triple the insertion interval. Therefore, Deadspace Finder sets a default insertion interval to be one third the desired appearance interval, but more intelligent variants could adapt the insertion interval to the user's browsing patterns over time.

In some situations, content that has already been placed can get in the way of an existing webpage due to changes on the page. We tested several alternate interfaces for conveying a collision to the user. One option was to visibly shake the content as if it were colliding. However, users found that this unnecessarily drew their attention to the colliding content

Figure 8-2: If deadspace content gets in the way due to changes on the page (left), Deadspace Finder detects the collision. The content turns into an apology message with a transparent background (right) and fades away, but the user could click “show anyway" to bring it back.

when their attention was elsewhere. Instead, it should communicate a collision only if the user happens to already be looking at it. Hence, in the final iteration, the content dismisses itself by transforming into a transparent “Oops, I’m in the way" apology message and fades away (Figure 8-2). However, the user has the option to click a “show anyway" button to bring back the content if desired.

8.1.3 Alignment with Existing Page

Deadspace Finder aims to create opportunities for peripheral interaction while minimizing disruption to the existing usability and flow of the page. In early testing, we found that deadspace content was more likely to flow visually with the page if it was positioned at the center of a section of deadspace. If content was positioned off-center within the deadspace, it was more likely to appear ajar or misplaced. Furthermore, small sections of deadspace sometimes helped visually separate sections of page content, and large sections of deadspace were more likely to already be aligned with page content. Thus, given a set of deadspace options, deadspace content is placed at the center of the largest one.

8.2 Implementation

In this section, we describe Deadspace Finder (Figure 8-3), our approach for opportunistically discovering deadspace, placing content, and dynamically removing the content when it gets in the way due to changes on the page. It is implemented as a Chrome extension, which end users can install to apply it to any pages they visit.

8.2.1 Deadspace Detection

To find potential locations for placing content, Deadspace Finder needs to identify sections of deadspace on the page. We considered using DOM (Document Object Model) based methods that leverage the structure of the webpage, but it can be difficult to determine how contents in DOM elements are perceived to the human eye. For example, an image element may appear either blank or full of content. Instead, we took a pixel-based approach to detect more closely what a user is actually seeing.

First, Deadspace Finder captures a screenshot of the visible viewport of a webpage. To find deadspace, Deadspace Finder identifies the background color of the page. Because webpages can contain multiple background colors, such as pages with several major sections, Deadspace Finder scans the image to determine the top-k most frequent colors, which it assigns to be candidate background colors. While it could scan every pixel, we found that scanning at 100 pixel increments reliably identified the background colors and was more efficient. We used $k = 5$ for our experiments.

Once the background colors are determined, Deadspace Finder identifies the r largest rectangles that consist entirely of pixels of each background color. We solve this problem using the maximal rectangle algorithm [108], where each cell of the image represents a pixel, and contains an indicator for whether it contains the background color or not. An efficient maximal rectangle algorithm is a greedy algorithm that runs in $O(mn)$ time on an $m \times n$ image. Using the greedy algorithm, we found the r largest rectangles by removing the

Figure 8-3: Deadspace Finder automatically detects deadspace and selects a location for placing deadspace content without occlusions.

maximum rectangle found after each iteration, and re-running the algorithm with the updated matrix. We found that setting r to 10 produced a reasonable set of candidate rectangles within which to place content.

8.2.2 Deadspace Selection

Given r largest rectangles and a minimum size to fit desired content, Deadspace Finder excludes rectangles that are too small, then determines the best rectangle within which to place content. It identifies the best candidate rectangle based on a set of adjustable parameters, such as area, aspect ratio, and location. For example, content could be placed closer to the center or periphery of the page depending on how urgent or salient it needs to be, or could be placed in a rectangle that could fit a particular aspect ratio, depending on the content. Given the usability considerations described above, our specific implementation selects the largest rectangle by default.

In order for the content to embed into the webpage and scroll naturally with the rest of the page, Deadspace Finder computes the rectangle positions relative to the top left corner of the webpage. Since a user could be browsing anywhere on the page at the time a screenshot is taken, the screenshot may be offset relative to the top left corner. To account for this

offset, Deadspace Finder determines the browser viewport's left and top positions when a screenshot is taken, and shifts each rectangle's coordinates by those values to find its absolute position. In some cases, if a user is scrolling quickly, the viewport may shift during the moment a screenshot is being taken. Thus, Deadspace Finder compares the pre-screenshot viewport position with its post-screenshot position, and only considers a screenshot to be valid if the viewport position has not changed.

Given the absolute position of each rectangle, Deadspace Finder filters out rectangles that are inappropriate for content placement. In early testing, we found that these tended to be spots that might look blank, but are not in fact unused space. The most common kinds of inappropriate rectangles include: 1) large textboxes, 2) rectangles inside iframes such as videos or animated displays, and 3) rectangles in elements that are independently scrollable. To filter out rectangles intersecting textboxes or iframes, Deadspace Finder finds the coordinates of all text input, content-editable, or iframe elements on the page, and excludes rectangles whose positions would intersect with any of those coordinates. To determine whether a rectangle intersects a scrollable element, Deadspace Finder identifies the element deepest in the DOM tree at the rectangle's position (using document.elementFromPoint()), then traverses its ancestors until it either finds an ancestor that is scrollable, or reaches the root of the DOM.

8.2.3 Dynamic Adaptation

In many situations, a webpage may change dynamically as a result of user interaction or asynchronous Javascript running on the page. For example, a user may click on an image to view a modal gallery of related images, or a page may load in additional features long after the primary content has loaded. In these cases, a screenshot that is taken before these changes have occurred may lead to a collision between the deadspace content and the surrounding content. In this section, we describe how Deadspace Finder monitors changes and handles collisions as they arise.

Collision Detection

To monitor potential collisions with elements on the page, Deadspace Finder takes a screenshot at regular intervals to check for changes, and simultaneously determines the deadspace content's position relative to the screenshot by inspecting the DOM. Using the screenshot, Deadspace Finder checks the 1-pixel perimeter surrounding the deadspace content to determine whether the color still matches the previously identified background color. Deadspace Finder determines that a collision has occurred if any pixel on the content's perimeter no longer matches the background color. Although changes could also be detected using DOM mutation listeners, we found that DOM additions and deletions alone did not capture all changes, and detecting changes on all DOM attributes led to excessive computation. We found that our pixel-based approach of checking the perimeter was more reliable, and could be done efficiently because the content is typically small in size.

Deadspace Finder uses a combination of time-based and event-based triggers to detect collisions in a timely manner without using excessive computation. In addition to checking every 15 seconds for a collision, it also checks as soon as the user returns to an old tab and when a zoom occurs on the page, which are times when the page may have changed.

Page Zoom

Some webpages are not displayed at a 100% zoom level because the user has set a browser-wide zoom level, or because the user has zoomed in or out on a particular page. In these cases, the screenshot taken will be stretched or compressed relatively to the viewport. Furthermore, screenshots have higher pixel densities on retina displays than on regular displays. On Mac retina displays, the pixel density is twice as high. Deadspace Finder determines the zoom level and retina setting on the fly by comparing the screenshot dimensions to the browser viewport dimensions, and scaling the screenshot accordingly.

Page Bounce

A user scrolling quickly near the edge of the page may cause the webpage to bounce. If a screenshot is taken at the moment the page bounces, the position computed for the screenshot will be inaccurate. For example, if a page bounces at the top of the page due to a user scrolling up, the viewport position will appear to be zero even though the pixel at the zero position is partway down the screenshot. Deadspace Finder detects a page bounce by inserting a div that is one-pixel tall at the zero position, with a color that is imperceptibly different from the page's border color. If the screenshot does not have this color at the edge, Deadspace Finder determines that a bounce is occurring. We found page bounces to be infrequent due to their transient nature. Thus, at the time a screenshot is taken, if a page bounce is detected, the screenshot is ignored.

8.3 Study 1

To measure the availability and characteristics of deadspace on the web, we conducted an analysis of deadspace on 500 webpages. The webpages were collected from three sources: a website aggregator (Popurls), a dataset of publicly shared web browsing histories (Eyebrowse), and the Alexa top 50. Within each source, we ensured that any given domain name appeared only once, with the purpose of collecting diverse styles and layouts. Furthermore, we disabled ad-block while running the study to establish a conservative estimate of the amount of deadspace available.

For each webpage, we ran Deadspace Finder and recorded the sizes and positions of the deadspace found. To ensure that the deadspace could fit a reasonable unit of content, Deadspace Finder was parameterized to exclude any rectangles whose areas were smaller than 100×100 pixels. Furthermore, because page layout and deadspace may vary depending on browser dimensions, each webpage was loaded at nine different browser viewport sizes: in addition to an average viewport dimension of 1366×784[1], we also set viewport widths of

[1] https://css-tricks.com/screen-resolution-notequalto-browser-window/

800 pixels, 1200 pixels, 1600 pixels, and 2000 pixels. For each width, the viewport height was set to two different heights, with width-to-height ratios of 1.5 and 1.7. After loading each page and applying Deadspace Finder, we manually labeled each result to evaluate the appropriateness of the deadspace found, as well as missed opportunities.

8.4 Study 1 Results

Of the 500 webpages total, we excluded 16 of them (3%) which failed to load, leaving 484 pages for analysis.

The median time taken to run our Javascript implementation of Deadspace Finder on the average viewport was 1.7 seconds ($\sigma = 0.48$), with one background color considered. The largest portion of this time was spent executing the maximal rectangle algorithm (1.1 second), with relatively less time spent taking the screenshot (0.48 second), determining the background colors (0.005 second), and finding pixels containing the background color (0.05 second). Each additional background color considered added an extra 1.1 seconds.

8.4.1 Deadspace Characteristics

As our goal is to enable secondary content to appear in deadspaces, we examined the likelihood that deadspace could fit different kinds of secondary content (Figure 8-4). In addition to the minimum size (100×100), we examined common dimensions found in secondary tasks or peripheral displays, including: web icons (125×125), web images (120×240), computer notifications (284×144), and Chrome panels (240×290). Chrome panels are typically used for secondary activities such as instant messaging. At the average viewport setting, we found that 88% of webpages contained deadspace large enough to fit the smallest 100×100 display, 80% could fit a web icon, 66% could fit a web image, 43% could fit a computer notification, and 27% could fit a Chrome panel. The likelihood of fitting larger content increases markedly between a viewport width of 1200 to 1600, likely due to

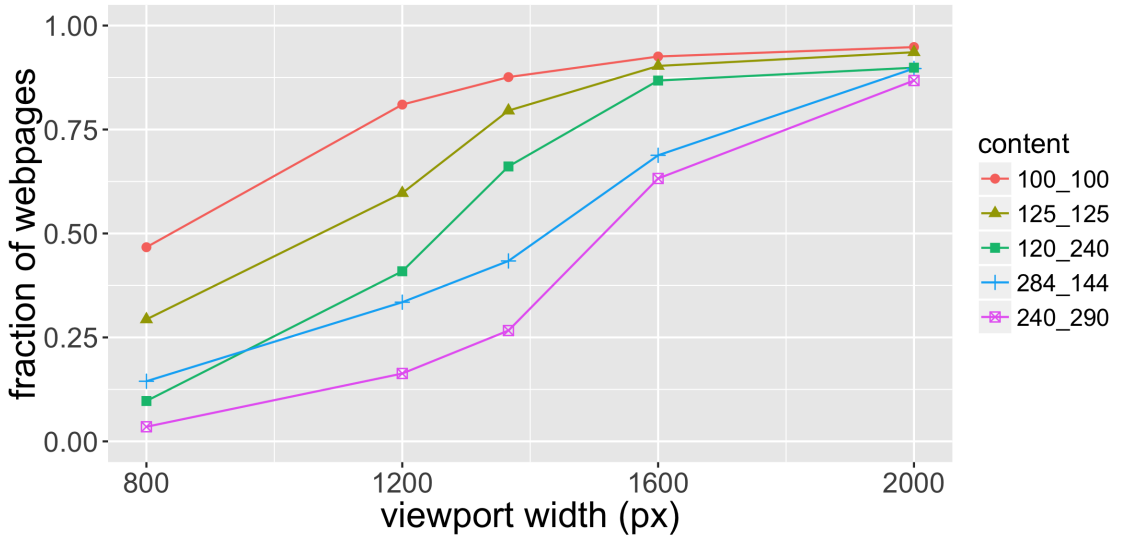


Figure 8-4: The percentage of pages that could fit different types of content within its deadspace, at viewport widths of 800, 1200, 1366 (average), 1600, and 2000. We show only results for viewports with an aspect ratio of 1.7, because a ratio of 1.5 yielded very similar results.

space gained in vertical margins.

Figure 8-5 shows a heatmap of deadspace rectangles at the average viewport dimension (1366×784). Only rectangles with areas greater than 100×100 pixels were considered deadspace. We observe the highest occurrence of deadspace in the bottom-right side of pages and within vertical margins. This is consistent with the observation that web content tends to fill from the top left of the page, leaving leftover space in the bottom and right sides. The top border appears least likely to contain deadspace, likely because it is valuable real estate for placing menu bars and navigational content.

We further computed the likelihood of webpages to contain a deadspace rectangle adjacent to one of the four margins, as well as deadspace in non-margins (Figure 8-6). While a majority of webpages contained deadspace bordering the left, right, and bottom margins (72%, 77%, and 67%, assuming the average viewport size), substantially fewer had deadspace in the top margin (30%). Furthermore, as viewport size increases, the likelihood of deadspace in the top margin does not increase nearly as much as the other margins, suggesting that the top of the page is reserved for high-priority items even as vertical margins get wider.

While there was a high occurrence of deadspace in the vertical margins (Figure 8-7a), 56% of webpages contained deadspace that did not border any margin. For example, deadspace also tends to appear in spaces next to large titles (Figure 8-7d), or on the right side of webpages where there is often less content (Figure 8-7e,f). Smaller areas of deadspace may appear between sections (Figure 8-7c), within spaces leftover from text wrapping (Figure 8-7c), in the area under a table of contents (Figure 8-7b), or due to neighboring regions jutting to different extents.

Finally, we examine how the consideration of multiple background colors affects the likelihood of finding deadspace (Figure 8-8). While considering two background colors increased the likelihood of finding deadspace over one background color, little was gained in considering more than two background colors. It could be the case that most webpages contain one or two dominant sections of solid background, with additional colors forming smaller

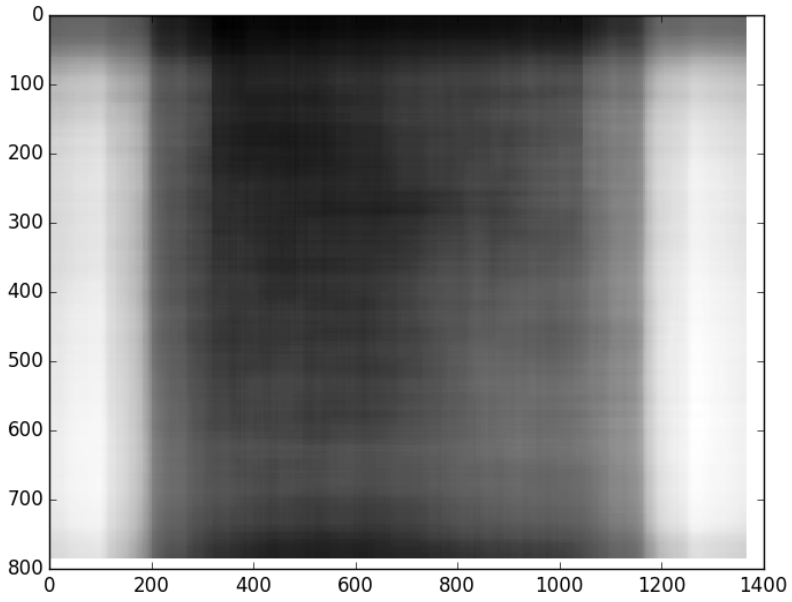


Figure 8-5: Heatmap of deadspace found across 484 webpages in a 1366×784 browser viewport, created by overlapping rectangular deadspace larger than 100×100 pixels. Whiter areas indicate higher occurrence of deadspace. We observe that left and right margins are frequent deadspace, and that the lower-right of a page has more deadspace than the upper left.

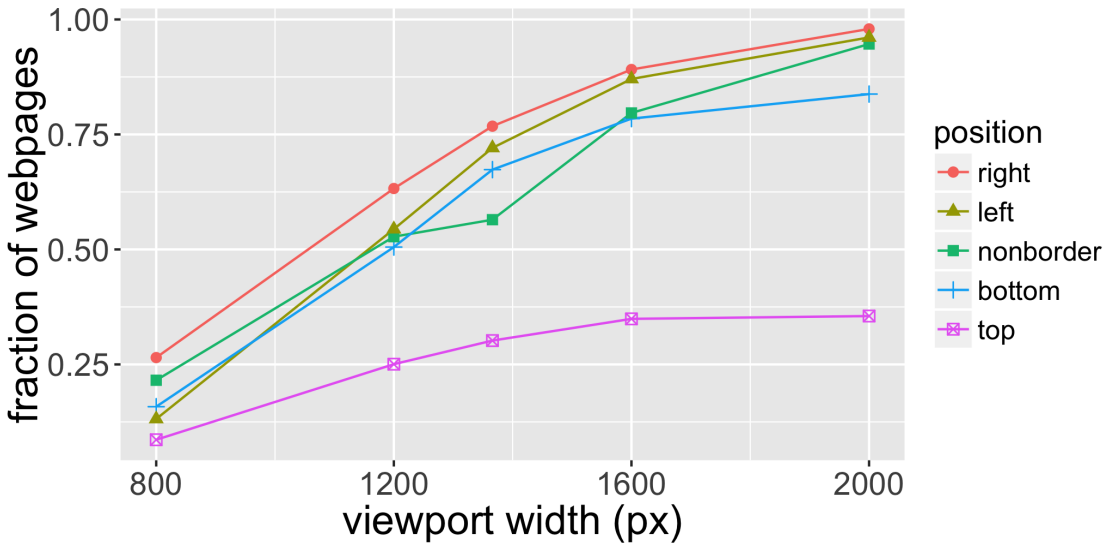


Figure 8-6: The percentage of webpages containing deadspace at each margin and at non-margins, at a browser viewport size of 1366×784. We show only results for viewport aspect ratio of 1.7, because a ratio of 1.5 yielded very similar results.

sections or foreground.

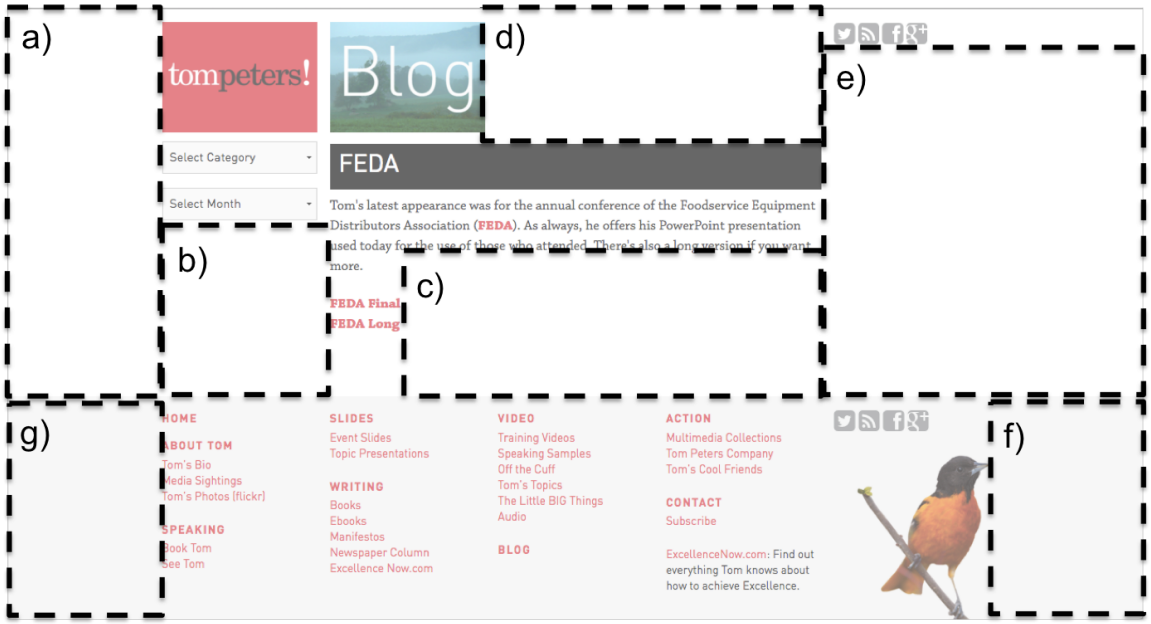


Figure 8-7: An example of different kinds of deadspace.

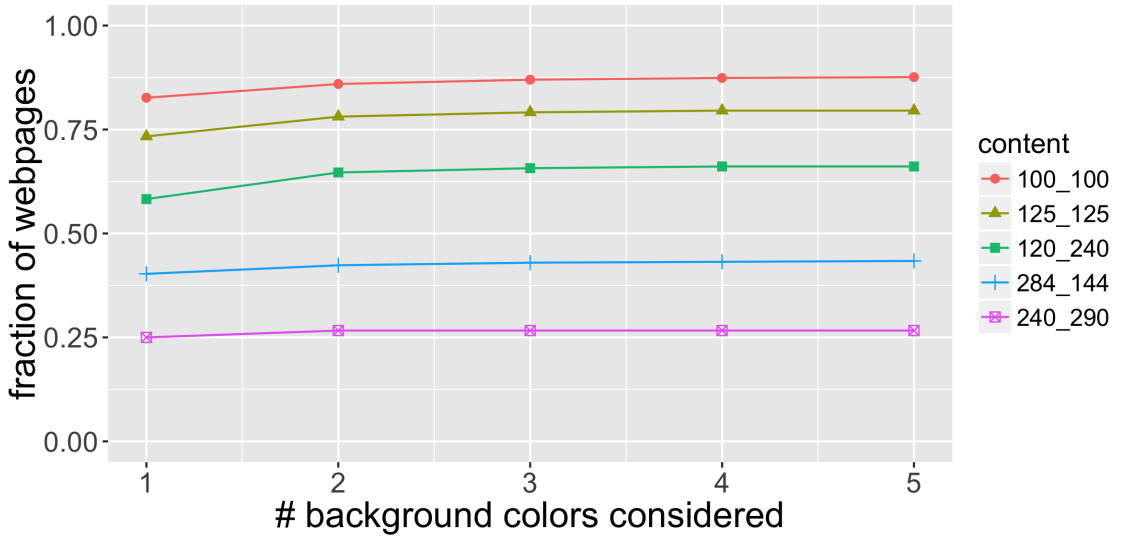


Figure 8-8: The percentage of pages that could fit deadspace of different dimensions, when Deadspace Finder considered different numbers of background colors. Deadspace was found on more pages when the number of colors was increased from 1 to 2, but little was gained beyond 2 background colors.

8.4.2 Deadspace Finder Accuracy

To characterize the accuracy of Deadspace Finder, we manually examined the 484 webpages, at the average browser setting. For each webpage, we labeled any *missed* deadspace and, for each deadspace rectangle found by Deadspace Finder, whether it was *faulty* deadspace. Figure 8-9 shows a few examples of missed deadspace and faulty deadspace from our corpus. In ambiguous cases, such as space within nearly-solid parts of a large image background, we judged whether secondary content would be appropriate to place in that location.

Deadspace Finder did not detect deadspace in 12% of webpages. 4% indeed did not contain deadspace, whereas 8% were missed opportunities. Pages that did not contain deadspace sometimes had large, colorful image backgrounds, or contained many smaller sections filled with content, such as a Pinterest gallery. Hence, any remaining space on those pages was smaller than 100×100 pixels. Among pages containing *missed* deadspace, 32% were due to Deadspace Finder's greedy search algorithm for identifying maximum rectangles, and 78% missed deadspace due to the presence of a subtle non-solid background. These

include nearly-solid image backgrounds (27%), textured backgrounds (24%), striped or grid backgrounds (17%), or subtle color gradients (10%). Deadspace Finder did not handle these cases since it searched for solid blocks of the same color. Lastly, 7% contained a rectangle that was falsely rejected because it overlapped an iframe but the iframe did not contain content, or it intersected a textbox but the textbox was in fact invisible. However, Deadspace Finder accurately rejected the majority of these cases (95%).

In evaluating *faulty* rectangles, we found that 2% of the rectangles found (4% of webpages) were not in fact situated within unused space. Faulty deadspace tended to coincide with semantically meaningful blocks of solid color. For example, one rectangle was part of a black shirt (Figure 8-9), and another was a grid within a picture of a game board. While they did not physically occlude any primary task, we consider these faulty deadspaces because the semantic contrast could be surprising or unnerving.

Rectangles containing the non-primary background color were more likely to be faulty. We observed the greatest percentage increase in faulty rectangles (from 0.5% to 1.8%) when the number of background colors considered increased from 1 to 2. While tertiary and quarternary background colors did not have very many rectangles large enough to be considered deadspace, secondary background colors often did include deadspace, but in

Figure 8-9: Examples of missed deadspace (left) and faulty deadspace (right). Missed deadspace often had textured backgrounds (left), and faulty deadspace coincided with meaningful content, such as inside a black shirt (right). Faulty deadspace tended to occur on a color that was not the primary background color on that page, as is the case in this example.

some cases also coincided with foregrounds.

8.4.3 Summary

In summary, we have demonstrated that Deadspace Finder can identify deadspace across a large number of pages with reasonable accuracy. We also showed where deadspace is likely to be found, what kinds of content the deadspace can fit, and how its location and availability vary across viewport sizes. Interestingly, deadspace occurs not only in margins, but also in a variety of non-margin spaces. While we disabled ad-block during the study, we expect more space to be available in browsers where ad-block is turned on. In the next study, we evaluate Deadspace Finder's behavior during real web browsing. To minimize faulty rectangles and optimize speed, we set Deadspace Finder to consider only the primary background color when identifying deadspace.

8.5 Study 2

To evaluate the dynamic behavior of Deadspace Finder and its effect on end users, we conducted a 10-day study in which participants used Deadspace Finder on their personal computers during their normal web browsing activities.

The questions our study sought to answer are:

- Dynamic behavior: How frequently are there opportunities to use deadspace during web browsing, and to what extent can Deadspace Finder resolve collisions?
- Subjective experience: How does deadspace content affect user experience, and how does it compare to popup content that appears at the edge of the screen?

Figure 8-10: Each participant was exposed to deadspace content (left) and popup content (right).

8.5.1 Content

The content was randomly chosen from an image application. We used images on the basis that they are often shown in persuasive content intended to achieve peripheral goals, such as for mood management (e.g. uplifting images), personal analytics (e.g. charts and graphs), and ambient awareness (e.g. lighting and weather displays). They could also be delivered in a controlled manner to all users. To keep the kinds of images consistent, we selected from a stock set of cat photographs.

To understand how deadspace content affect user experience compared to popup content, we exposed each participant to both *deadspace* and *popup* content (Figure 8-10). Deadspace content appeared in the deadspace of webpages. We also set Deadspace Finder to use an injection interval of 10 minutes, with the intent of having an appearance interval of approximately 30 minutes. We felt that 30 minutes was a reasonable upper bound given that the goal of Deadspace Finder is to show non-urgent rather than urgent content. Content that was triggered due to computer idleness was set to trigger after 60 seconds of computer idle time, with no keyboard or mouse activity.

Popup content resembled the behavior of typical computer and browser notifications. They were displayed in the form of a pop-up alert in the upper right corner of the computer screen. The deadspace version was implemented as a Chrome extension, and the popup version as a

Mac application. Both kinds of content had a dimension of 175×175 pixels, between the size of a small icon and regular web image. All users experienced the same set of stock images.

To capture in-the-moment reactions, each deadspace and popup content contained two buttons at the bottom, labeled “I see it" and “I felt interrupted." Participants were asked to click on one of these buttons each time they saw an appearance, to simultaneously indicate that they saw it, and to indicate whether they felt interrupted by it. The content disappears as soon as it has been clicked. If not clicked, popup content remains on the screen for 15 seconds, and deadspace content remains on the screen until the user switches away from the tab.

8.5.2 Procedure

Participants met with a researcher to install both the deadspace and popup versions on their personal computers. They were asked to use their computers as they normally would for 10 days, with the exception of responding to the content each time they saw one. At the end of the study, participants returned for an interview and post-study questionnaire, which consisted of NASA TLX questions about both kinds of content. During the study, screenshots were also recorded each time deadspace content appeared, so that during the post-study interview, users could examine the screenshots and reflect on their experience.

To ensure that the two kinds of content appeared at similar frequencies, participants were exposed to deadspace and popup content on alternating days. The desired frequency on a *popup* condition day was determined by calculating the total number of deadspace content shown on all previous *deadspace* condition days, divided by the total seconds of computer activity on those days. We define a user to be actively using their computer as long as the computer has not been idle for more than 60 seconds. This gives us the probability of showing content in a given second on a popup condition day.

8.5.3 Participants

20 participants were recruited by emails sent through university email lists. They included 11 males and 9 females, ages 18 to 37 (mean 22), and consisted of undergraduates, graduate students, and staff. We selected only users who would not be traveling during the study, and who reported using their computer for at least 7 hours a day, the duration of a typical work day. So that they could install the plugins, we enrolled only those who owned Mac computers and used Chrome as their primary browser.

8.6 Study 2 Results

In this section, we report on Deadspace Finder's dynamic behavior and users' subjective experiences during real web browsing.

8.6.1 Dynamic Behavior

Overall, we logged 1305 total hours of computer usage over the course of 10 days. On average, users spent 6.5 hours ($\sigma = 1.9$) on their computers per day, within which 4.6 hours ($\sigma = 1.7$) were spent in the web browser per day.

Frequency of Opportunities

Opportunities to place deadspace content depend on both the existence of deadspace as well as user behaviors such as tab switching and computer idling. Thus, we examined the extent that Deadspace Finder could place content during real web browsing. We observe that the number of appearances grew linearly with browser usage time, approximately twice per hour of browser usage (Figure 8-11). This appearance interval is approximately three times longer than the 10-minute insertion interval setting of Deadspace Finder, which is what we had intended. Content that appeared as a result of the computer idling were also

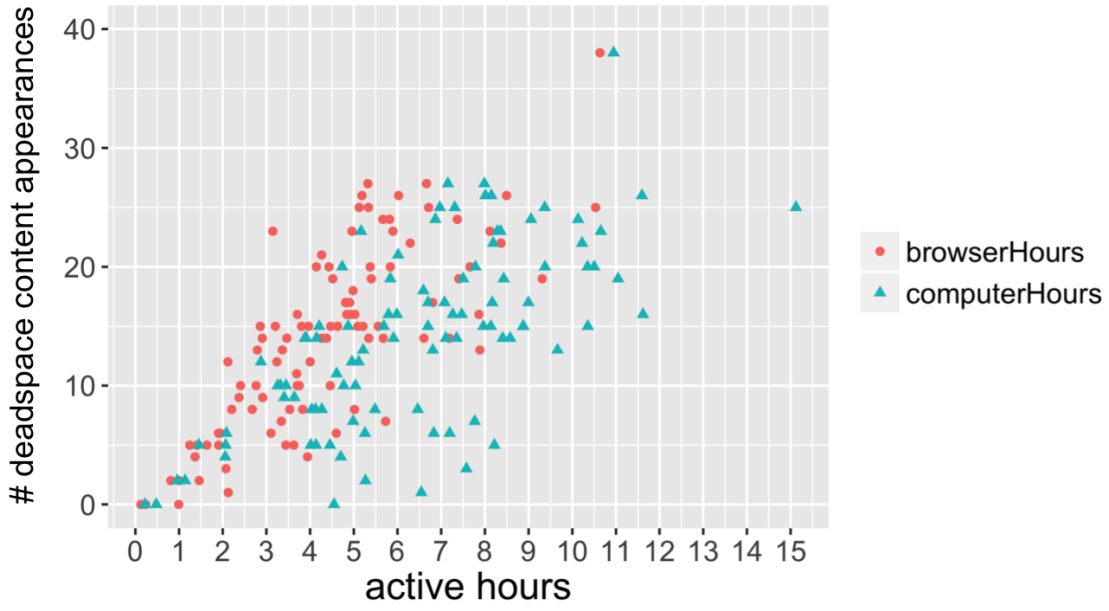


Figure 8-11: The number of browser hours and computer hours spent in a day, compared to the number of deadspace content appearances that day. Appearances include content that was clicked as well as those that were not clicked.

less frequent than those appearing during tab switching. This is not surprising given that computer idling tends to happen during breaks rather than while the user is actively on the web.

We also found that deadspace content appeared in diverse settings. Over the course of 10 days, deadspace content appeared in 1433 pages total, among which 877 were unique urls and 311 were unique domains. For each user, they appeared in 44 unique urls ($\sigma = 17$) and 23 unique domains ($\sigma = 9$), and the url with the most occurrences accounted for 12% of appearances ($\sigma = 6.5\%$).

Mechanics of Interaction

Although deadspace content was intended to be seen only during tab switches and after computer idling, there were several unintended consequences. Some users reported seeing deadspace content while deleting a series of tabs, and would realize they'd seen it only after

already deleting the tab. Others who use keyboard shortcuts for tab switching saw it in the midst of traversing a series of tabs, because some gestures require traversing all tabs between two tabs to get from one to the other. In these cases, users often overshot the tab, and needed to backtrack in order to access the deadspace content. To prevent this from happening in the future, Deadspace Finder could detect a series of rapid tab removals or tab switches.

A few users also encountered deadspace content even though they weren't in fact returning from idle. This occurred in cases where a user was reading a webpage without mouse movement for more than 60 seconds, the threshold at which Deadspace Finder assumes the computer is idle. Users found these situations more distracting, since they would see the content fading in. Lengthening the 60 second interval may decrease the likelihood of this occurring.

Tab Usage

Participants had on average 16 tabs open at any one time ($\sigma = 12$), across 4 windows on average. Among tabs in which deadspace content appeared as a result of tab switching, 70% were tabs whose most recent visit was within 10 tab switches, and 54% within 5 tab switches. Overall, switching to a recent tab was more common than switching to an old tab. Given this behavior, future versions of Deadspace Finder could conserve computation by running only on the most recent set of tabs encountered, and aborting once a tab becomes older than a threshold. This could prevent Deadspace Finder from being too computationally intensive for those who use a large number of tabs.

Collision Detection

Users felt that deadspace content almost always appeared in unused space. Deadspace Finder detected collisions in 7% of appearances on average ($\sigma = 5\%$), and was able to detect them promptly after they appeared (median=1.1 seconds, $\sigma = 5.3$). Several users described

the content as already disappearing by the time they noticed it blocking web content: *"It wasn't really annoying because typically it was fading already, so it never really got too distracting."* Some users said the collisions occurred soon after a tab switch, which suggests that that those collisions arose due to a screenshot being prematurely taken before the page had fully loaded, and were resolved quickly due to Deadspace Finder's event-based trigger.

8.6.2 Subjective Experience

Users were less likely to indicate that they felt interrupted by deadspace content, compared to popup content. Taking only the instances where users responded, 24.7% ($\sigma = 0.26$) of deadspace content received an "I felt interrupted" rating, compared to 30.7% ($\sigma = 0.31$) in the popup condition. We ran a mixed effects logistic regression, with the participant as a random effect and content type as the fixed effect. We found that deadspace content was significantly less likely to receive an "I felt interrupted" response than popup content ($p < 0.001$, $F = 35.7$), though the practical difference was small.

Deadspace content was also less likely to be perceived than popup content. This result is consistent with research on change blindness, which posits that people are less able to perceive changes if they occur during a visual saccade or between scene changes [129, 137], as is the case during a tab switch. While 79.8% ($\sigma = 0.14$) of popups received responses on average, 73.2% ($\sigma = 0.13$) of deadspace content received responses. A mixed effects logistic regression, with the participant as a random effect and content type as the fixed effect, found that popups were more likely to be perceived ($p < 0.01$, $F = 8.1$). Though the difference was statistically significant, the practical significance was relatively small, despite the change blindness effect. It's possible that the visual salience of the images used in the study, combined with having viewed the same tab recently, made it easier for users to notice the change even in the deadspace condition. Text content may have been less noticeable, and should be studied in the future. Overall, we find this difference in noticeability to be acceptable given that the goal of Deadspace Finder is to present non-critical content that would acceptable if sometimes missed.

Upon further analysis, we found that deadspace content was less likely to be perceived if it appeared farther to the right side of the page. For example, deadspace content appearing more than three-quarters to the right of the page were noticed 62% of the time, compared to 70% for all other page locations. Deadspace content was also less likely to be noticed if there had been more tab switches since the last visit of the same tab. For example, users noticed the deadspace content 72% of the time if there had been fewer than 4 tabs visited, compared to only 62% if there had been more than 4 tabs visited. A mixed effects logistic regression found these differences to be statistically significant ($p < 0.05$ and $p < 0.001$, respectively). These results are consistent with users being more attuned to the left side of the page where content tends to exist [28], as well as visual perception theories indicating that scene changes contribute to change blindness.

In the post-study NASA TLX questionnaire, no significant differences were found between deadspace content and popup content. In the following sections, we elaborate on the subjective feedback received from participants.

Manner of Appearance

Many users indicated that deadspace content was less disruptive because it was already in place by the time it was seen, rather than visibly moving into view. For example, one participant said that *"I wouldn't see the [deadspace content] visibly appearing, whereas the [popups] I would always see them as they appeared, which was more annoying. If I'm doing something else, and it pops up, it can be distracting."* Because they felt less obtrusive, one participant described deadspace content as filling a space *"between turning off [the content] completely and having a thing that comes up all the time."*

Several also indicated that deadspace content was less disruptive than popups because they did not occlude their view or block access to tabs and extension icons: *"The [popup] was bad because it would cover some of my screen content, which was especially annoying when it covered some of my tabs."* Another felt that *"The [deadspace version] was more respectful. It would find a random unused space, whereas popups sometimes have something behind*

it." One user mentioned that typical popup notifications can stack up vertically if there are several being received at once, which can block even more screen space. He appreciated that Deadspace Finder *"puts it in your line of view, but doesn't disrupt what you're doing at the time."*

Relation to Surrounding Content

Although deadspace content was placed in unused space, some found it distracting if it was too visually salient against the background color of the page, making it more difficult to ignore. One user explained that when deadspace content was colorful against a blank, solid background, it was more likely to *"draw my eyes away from whatever I was doing."* These users wished the deadspace content blended in more with surrounding content or were placed on pages that were more cluttered: *"If the design of the webpage is very cluttered, then the [deadspace content] sort of blended in, and felt more normal."* Some also preferred content to blend in more for privacy reasons, so that onlookers would be less likely to notice.

Deadspace content was also disconcerting when it was semantically confusing in the context of adjacent content, or when mistaken for page content. For example, one said that *"sometimes they appeared under a chunk of text, making me confused as to whether they were part of the website."* Another encountered a deadspace image next to an image search for rabbits.

Desired Use Cases

When asked what kinds of applications they would use Deadspace Finder for, users described content that they would like to see regularly, but that they wouldn't mind missing, such as a word-of-the-day, uplifting image, or news headline – content that would otherwise take too much effort to go to on their own. Some felt that the varied location of deadspace content made it feel more playful and surprising, and thus more likely to be persuasive if noticed. In contrast, they preferred popups for high-priority items they would not want to miss (e.g. text messages), so that they would know to always expect them in the same place. These reports

are consistent with our goal of using deadspace for non-critical content that is acceptable if sometimes missed, but beneficial if noticed.

8.7 Design Implications

Results from these studies indicate that Deadspace Finder is able to detect deadspace and resolve collisions, and that the visual and behavioral properties of its content affect user experience. We discuss implications related to the automatic detection and use of deadspace for peripheral interaction.

8.7.1 Deadspace Detection

In Study 1, we found that *faulty* deadspace tended to contain the non-primary background color. Limiting the deadspace detection algorithm to consider only one background color could help minimize faulty detection. Conversely, *missed* deadspace tended to occur on textured backgrounds. Deadspace Finder could be modified to handle these backgrounds, such as by detecting repeated patterns or by mapping similar shades of a color to the same background color. Given that Deadspace Finder is intended for low-priority, optional interactions, Deadspace Finder could err on the side of caution by only considering rectangles with solid backgrounds containing the primary background color, to minimize the faulty detection of deadspace at the expense of finding every possible deadspace.

8.7.2 Content Placement

Although deadspace content was noticed frequently, it was less likely to be noticed than popup content. This is not surprising given that we had purposefully designed deadspace content to appear without movement. However, to maximize the chances that deadspace content will eventually be noticed, Deadspace Finder could avoiding presenting content on

the farthest right side of the page, or on tabs that are very old. Furthermore, because some users saw deadspace content just as they were deleting a series of tabs, Deadspace Finder could detect this behavior and avoid placing deadspace content in these situations.

Our finding that deadspace occurs frequently and in diverse locations also opens up opportunities for selective filtering. For example, using website categorization APIs or user whitelists, it could prioritize pages intended for leisure (e.g. BuzzFeed), while avoiding pages that are work-related. Some users described seeing deadspace content come up consistently in a spot they particularly liked, on a page they browse frequently. Allowing users to upvote a desired hotspot could help provide on-the-fly feedback to the algorithm. Finally, it is unclear what the benefits and costs are of placing deadspace content in varied locations opportunistically, rather than keeping it in a consistent on-screen location. While the varied placement felt more playful and surprising to users, and could thus increase engagement and combat desensitization, it is also less predictable and could thus contribute to a higher mental workload.

8.7.3 Visual Appearance

Users preferred when deadspace content blended in visually with the rest of the page, so that it could be stumbled upon without drawing as much attention. Future iterations of Deadspace Finder could tailor the visual salience of deadspace content to the surrounding page. For example, colorful content could initially be shown greyscale or nearly transparent, and only become opaque when a user interacts with it. It could also be shown with greater salience on cluttered pages and with less salience on sparse pages, using existing clutter detection algorithms [131, 132], or using the size of the containing deadspace rectangle as a heuristic.

Although users preferred deadspace content to blend in with the background page, some found it confusing when it initially appeared to be part of the original page. To blend in with the page while still separating itself semantically, all deadspace content could share a

uniform feature (e.g. dotted borders or a distinguishing icon) so that it can be identifiable as non-page content after it is noticed. While we did not modify the visual appearance of content in our study, we leave the exploration of these properties to future work.

8.7.4 Privacy

For some users, the proximity of deadspace content to page content brought up privacy concerns, particularly when peers could see their computer. Some ambient interfaces help ensure privacy by encoding information abstractly [35], such as displaying more flowers the more a person has exercised. While this dimension of abstraction was outside the scope of this work, the input content to Deadspace Finder could in theory be tailored to show sensitive information more symbolically. In the future, deadspace content should also be collapsible so that users have control over its visibility.

Furthermore, because Deadspace Finder's was able to intelligent detect collisions with page content, some users wondered whether if Deadspace Finder was aware of the content they were browsing. To prevent these kinds of privacy concerns, systems like Deadspace Finder should do their best to be transparent about the algorithm used, and expose their mechanics to users.

8.7.5 Limitations

We created Deadspace Finder for desktop web browsers, due to the availability of current browser APIs and screen real estate. However, deadspace also exists on other platforms where users juggle multiple tasks. Our approach of finding deadspace and resolving collisions relies primarily on a screenshot, which could be applied to many settings. Future work could explore how to detect issues such as textboxes and scrolling with less understanding of the underlying objects, by using pixel-based techniques [46] or accessibility APIs.

Furthermore, Study 2 uses stock images rather than content that is personally meaningful or

useful to the participant. We used stock images because they may be desired by users, but still non-critical and thus acceptable if missed. They also allowed us to avoid the potential confounding factor of users reacting to personally critical incidents during one condition but not the other. However, future work should explore the use of deadspace for personally useful content (e.g. flashcards, weather forecasts), to reflect a more realistic context of use.

8.8 Conclusion

In this chapter, we introduced deadspace as a way of complementing wait time, and introduced an approach for automatically discovering deadspace on any webpage, placing desired content in deadspace, and resolving collisions on the fly. Through two evaluations of Deadspace Finder, we found that deadspace is a feasible and effective way to display secondary content, and provided implications for how to automatically detect and use deadspace more effectively. These findings inform future uses of deadspace to enable awareness while minimizing interruption.

Chapter 9

Discussion and Future Work

9.1 Design Guidelines for Wait-learning

Designing for wait-learning involves both identifying appropriate waiting scenarios, and designing interactions that balance between enabling awareness and minimizing interruption. Our cumulative findings suggest the following design guidelines for wait-learning:

1. Consider the switch cost given the ongoing activity surrounding the waiting period: switch cost is high if the complexity of information that must be retrieved at the end of the waiting period is high. Switch cost can be decreased if the activity's intermediate state can be externalized (e.g. chat log), so that intermediate state does not need to be maintained in memory.
2. To avoid interrupting the wait-time task, use waiting periods that tend to end softly rather than abruptly.
3. Select waiting scenarios in which the time taken to access the wait-time task is short, relatively to the waiting duration. In some waiting situations, it may not currently be possible to embed a wait-time task near the ongoing activity due to security protections or technical limitations (e.g. mobile texting applications). If switch time is high, consider whether there are mitigating factors such as a long waiting period or

low mental workload.

4. Embed wait time tasks within pre-allocated space or unused deadspace so that they are less likely to occlude or interfere with existing activities.
5. Use wait-time tasks that have low complexity, with few interdependent parts. Although this thesis did not explore highly complex wait-time tasks, it would be reasonable to assume that the shorter the waiting period and the more complex the ongoing activity, the simpler the wait-time task should be. Design the modality (e.g. typing, button pressing, speaking) of the wait-time task so that it complements the modality of the existing activity, and so that it is socially appropriate given the context in which waiting is likely to occur.
6. Design the learning task to appear at the *start* of the waiting period rather than in the middle of the waiting period.
7. Allow users to optionally continue learning during longer waiting periods by enabling them to fetch more wait-time tasks. The next task can be optimized based on the previous task's content, operation, and complexity to best encourage efficiency and flow.

9.2 Customization

Some waiting moments coincided with very different levels of mental demand in different situations, e.g. instant messaging and wifi waiting. Future implementations of wait-learning could consider a casual interaction design [123] that enables a spectrum of interactions, depending on how much the user is able to engage at the moment.

One possibility is to allow a user to customize or toggle apps based on personal circumstances. However, this introduces the risk that users may forget to turn an app back on after turning it off, and also adds an additional layer of management that could ultimately diminish engagement with learning. Alternatively, a contextually aware system could attempt to infer availability using sensors, or detect user patterns over time coupled with probes for

the user to acknowledge if a certain pattern is true. For example, it may detect that a user never engages in wait-learning while instant messaging with a particular person, or always responds to wait-learning triggers on Tuesdays at 2pm while walking to class. Given the day to day variations we observed during the study, these pattern-based suggestions should be made conservatively, to avoid unnecessarily reducing opportunities for learning.

Because the users in our studies were primarily beginning and intermediate language learners, users typically marked only a minority of words as already known when using WaitSuite. To avoid filling wait time with a large quantity of known words, WaitSuite could be modified for more advanced learners, by automatically filtering out vocabulary according to the user's language level or pre-study vocabulary quiz.

9.3 Automating Microtask Creation

Although microtasks can help people strengthen a skill, piece of knowledge, or habit through repeated practice, a key hurdle is the time-consuming process of creating the microtasks themselves. What if complex work tasks could be automatically decomposed into smaller chunks, for easier completion? For example, what if engineers could offload repetitive parts of their work to microtasks, to be completed through automation, or done while waiting for longer tasks to finish running? Similarly, what if educational applications or text editors could automatically detect a user's most common slip-ups or mistakes, and use them to auto-populate microtasks for practice? Future work could push the boundaries of microproductivity by leveraging context about a user's existing workflows to automatically generate microtasks.

9.4 Wait-learning as a Way to Maintain Focus

A potential risk of wait-learning is that it could distract a user from his or her existing activities. Surprisingly, however, we found in Chapter 6 that the opposite may be true. For

users who have existing habits of switching tasks while waiting, well-timed microtasks could help keep users attuned to their existing activities, such that those activities can be promptly resumed once waiting ends. In order to be effective, the microtask needs to be more attractive than alternative wait time activities. This is more likely to be the case if the microtask is easier to access, easier to discard (once waiting ends), and demands less decisional energy to initiate relative to competing wait time activities.

Such a possibility opens up a broad range of future designs for wait-learning and attention management. Wait-learning tasks could be designed to complement the main task or to encourage focus. For example, a wait time microtask could contain information relevant to the main task, encourage users to reflect on their ongoing task, or help them relax from a stressful situation. Furthermore, wait time tasks could be strategically placed to keep users visually attuned to the main task. For instance, augmented reality could help situate wait-learning tasks as close as possible to the wait time progress indicator, such as near a red light at an intersection, next to the elevator buttons, or even on the person directly ahead of the user in line. Of course, such designs ought to consider their impact on the wait-time task as well. More broadly speaking, microtasks could play an active role in helping users *manage* their own time and attention, keeping them focused and alert without disrupting their flow.

9.5 Wait-learning and Well-being

In our studies, most users had existing habits of engaging with digital activities during wait time. Many described wait-learning as a more meaningful way of spending time compared to their compulsive waiting habits. Some also described WaitChatter as being playful and game-like, similar to other games they would play on their phones while waiting. Hence, for these demographics, wait-learning did not actually fill unused time, but rather replaced existing habits with more meaningful ones.

However, wait-learning is less appropriate for those who already make effective use of wait

time, or who do not already engage in task switching and technology use while waiting. A potential downside of wait-learning could be the inadvertent replacement of time spent reflecting or resting, components that are essential to a healthy lifestyle. For example, wait time could be an opportune moment to daydream, reflect, notice something in one's environment (e.g. posters in elevators), or even engage in social activities. Although we addressed this problem through design, by making wait-learning tasks easy-to-ignore and non-intrusive, future systems could leverage more intelligent bio-sensing techniques and machine learning to detect or predict what a user is actually experiencing while waiting. For instance, an intelligent system could leverage a combination of bio-sensors, accelerometers, and location detection to infer whether it is truly a good time to intervene.

Although this thesis focused on two specific domains, wait-learning could potentially be used to encourage healthier habits in the future. For example, wait time could be an opportune moment to provide timely reminders to stretch, plan, meditate, relax, or reflect. In particular, the short, repetitive nature of microtasks could be capitalized to provide meditative qualities. As one user described in Chapter 6, the repetitive nature of flashcards felt similar to crocheting, a meditative activity she does in spare time. The design dimensions and theoretical framework established in this paper help form the basis for future work to help people wait-learn healthier habits in general.

9.6 Wait-learning Modalities and Devices

For practical purposes and for ease of user recruitment, we implemented wait-learning on smartphones and computers as graphical user interfaces. However, new technologies such as wearables and tangible user interfaces open up a broad space of future possibilities for wait-learning. For example, smartwatches and augmented reality glasses could reduce the effort required to switch between the ongoing and wait-learning tasks, creating truly blended and seamless experiences. Tangible interfaces could empower people to practice skills in domains that involve physical manipulation, like juggling or music rehearsal. Haptics and

voice interaction could enable learners to complete a flashcard without having to glance at a screen, allowing them to keep their eyes on their environment. Although the design principles outlined in this thesis were based on smartphone and desktop interactions, they provide a baseline for future work using a variety of modalities and devices.

9.7 Ethics and Privacy

While the current applications of wait-learning have been discretionary, allowing users to decide on their own volition when to learn and when to ignore, it remains an open question whether wait-learning ought to be made available to children and those who are not fully capable of making their own decisions. On the one hand, wait-learning could serve as a powerful supplement to classroom learning, for helping students study for standardized exams, or even for enriching the learning process of restless toddlers during a critical period of their development. On the other hand, because wait-learning is intended to be optional, and only for those who wish to change their existing habits and behaviors, it could have an unpredictable impact on developing minds.

In addition, the effectiveness of wait-learning lies in how seamlessly it can embed itself into existing activities and how reliably it can detect waiting, both of which involve insight into certain user activities and real-time user events. In deploying wait-learning systems, it is important to inform users of what data is and is not being collected, and what benefits they might expect, so that they can make an informed decision of whether to install. As more and more sensors and real-time technology are being made available to developers, creators of wait-learning systems should carefully consider privacy concerns at an early stage.

9.8 Wait-Learning Platform

In the future, wait-learning platforms could enable users to publish activities they would like to use as substrates for wait-learning, and for wait-learning developers to subscribe

to those activities. Third-party platform developers could furthermore be incentivized to open up their platforms for easier microtask embedding. Unlike advertising content, which aims to draw users to their landing page away from their ongoing activities, wait-learning interfaces are designed to facilitate continuity on the primary task. For instance, video-game platforms could be incentivized to allow wait-learning plugins to serve microtasks that keep users engaged while games load, or in between levels so that they can be reminded to take a healthy break, without leaving the platform altogether.

9.9 Limitations of Wait-Learning

9.9.1 Complexity of the Wait-learning Task

Because wait-learning takes place in the midst of existing activities, there is an upper bound on the complexity of the wait-time task. If the existing activity and wait-time task are both mentally demanding, the benefit of wait-learning may be offset by the high cost of switching back and forth. Furthermore, the user may be less likely to engage in the wait-time task if it feels substantially more complicated than alternative wait time activities, most of which tend to be lightweight.

One way to assist in the completion of more complex learning tasks is to externalize the task's intermediate state so that the user does not need to remember it. For example, if an algebra problem were to be broken up into multiple steps and spaced out over time, the previous step taken could be visualized to the user as a reminder. Secondly, if a user fetches a consecutive chain of microtasks and becomes increasingly immersed in learning, the wait-learning system could take advantage of this and present more and more complex problems at that time.

9.9.2 Digital Proficiency

The participants in our study were primarily young adults who were technologically adept. Some demographics may be less experienced with technology, limiting how quickly they can realistically switch to a digital wait-time task and back. Future work should explore how to best cater to their needs given varied technical skill levels, such as using tangible interfaces or wearable devices that are easier to access.

9.9.3 Future of Waiting

With the rapid advancement of technology and growing speed of computation, it is possible that current scenarios for waiting (e.g. waiting for wifi) may no longer exist in the near future. Despite this reality, I would argue that waiting will still exist, for two reasons: first, there is a human component to waiting that will always be present, such as waiting for a friend to arrive for lunch, or for an acquaintance to reply to a text message. Second, the emergence of new technologies will bring with them new challenges and accompanying delays. For example, the moment of hesitation introduced by pull-to-refresh did not exist until mobile phones enabled people to receive updated information on-the-go. One question to consider is whether people will become less tolerant of waiting as they expect shorter and shorter wait times. As indicated by existing research, wait-tolerance affects how quickly one abandons a task or switches to other activities, and could thus affect how soon the wait-learning task must appear before a user leaves in search of another activity.

Chapter 10

Conclusion

This thesis makes substantive contributions in the domains of productivity, learning science, peripheral interaction, and attention management. Specifically, the contributions are as follows:

- **Productivity**: 5 novel systems that enable micro-productivity during wait time, with demonstrated benefits over traditional reminders; a set of design guidelines for chaining together microtasks in a way that optimizes efficiency and mental effort.
- **Learning**: an expansion of micro-learning to the targeted use of wait time, demonstrating that existing principles of spaced repetition can be applied to daily waiting scenarios.
- **Peripheral interaction**: a pixel-based technique for enabling less intrusive peripheral interaction in the unused space of webpages; design guidelines for facilitating peripheral interaction during fleeting waiting moments.
- **Attention management**: a design space for wait-learning and a theoretical framework that extends existing work on attention management by incorporating dimensions such as wait time, ease of access, and mental demand for managing attention during wait time; empirical results indicating that mental workload is lower at the start of a waiting period compared to alternative timing conditions.

Taken together, these contributions open up a broad space for combining wait time with the pursuit of long-term goals, making meaningful use of time we never knew we had.

Appendix A

Materials for the Feasibility Study

Please note that WaitChatter was renamed ChatLearner in questionnaires to avoid biasing the user toward the notion of wait time.

A.1 Daily Survey

- ChatLearner exercises appeared at good moments within the flow of my daily activities.
- ChatLearner was disruptive to my daily activities.

A.2 Post-Study Questionnaire

- Please rate the frequency of ChatLearner exercises. (1: too infrequent, 7: too frequent)
- The non-contextual vocabulary that ChatLearner presented were words I wanted to learn. (1: strongly disagree, 7: strongly agree)
- I was satisfied with the variety of non-contextual vocabulary than ChatLearner presented. (1: strongly disagree, 7: strongly agree)
- The contextual vocabulary that ChatLearner presented were words I wanted to learn.

(1: strongly disagree, 7: strongly agree)

- I was satisfied with the variety of contextual vocabulary than ChatLearner presented. (1: strongly disagree, 7: strongly agree)
- Overall, I enjoyed using ChatLearner. (1: strongly disagree, 7: strongly agree)
- I would continue using ChatLearner if I could. (1: strongly disagree, 7: strongly agree)
- Using ChatLearner, I would engage in vocabulary practice more frequently than I would otherwise. (1: strongly disagree, 7: strongly agree)

A.3 Vocabulary List

A.3.1 Spanish

people, personas

year, año

day, día

man, hombre

world, mundo

work, trabajo

life, vida

children, niños

case, caso

thing, cosa

woman, mujer

money, dinero

fact, hecho

night, noche

area, zona

company, empresa

family, familia

business, negocios

side, lado

week, semana

country, país

council, consejo

room, habitación

member, miembro

problem, problema

car, coche

office, oficina

door, puerta

body, cuerpo

person, persona

month, mes

health, salud

law, ley

word, palabra

child, niño

society, sociedad

market, mercado

job, trabajo

process, proceso

effect, efecto

community, comunidad

evidence, evidencia

minister, ministro

morning, mañana

level, nivel

death, muerte

industry, industria

century, siglo

church, iglesia

history, historia

road, carretera

center, centro

food, comida

program, programa

result, resultado

hour, hora

committee, comité

team, equipo

course, curso

language, idioma

mind, mente

authority, autoridad

data, datos

class, clase

paper, papel

wife, esposa

city, ciudad

friend, amigo

price, precio

god, dios

town, pueblo

bed, cama

girl, chica

quality, calidad

music, música

game, juego

april, abril

student, estudiante

june, junio

hair, pelo

basis, base

series, serie

bank, banco

foot, pie

south, sur

west, oeste

secretary, secretario

security, seguridad

manager, gerente

heart, corazón

story, historia

letter, carta

chapter, capítulo

field, campo

movement, movimiento

success, éxito

analysis, análisis

news, noticias

evening, tarde

boy, chico

theory, teoría

approach, enfoque

growth, crecimiento

agreement, acuerdo

size, tamaño

son, hijo

space, espacio

property, propiedad

example, ejemplo

energy, energía

sir, señor

east, este

income, ingresos

buildings, edificios

treatment, tratamiento

july, julio

north, norte

loss, pérdida

activity, actividad

march, marcha

army, ejército

summer, verano

product, producto

wall, pared

teacher, maestro

unit, unidad

technology, tecnología

october, octubre

window, ventana

resource, recurso

sea, mar

september, septiembre

event, evento

january, enero

goods, bienes

blood, sangre

extent, grado

response, respuesta

majority, mayoría

degree, grado

economy, economía

glass, vidrio

glasses, gafas

street, calle

parliament, parlamento

labor, trabajo

species, especies

title, título

employment, empleo

daughter, hija

competition, competencia

november, noviembre

december, diciembre

sunday, domingo

purpose, propósito

task, tarea

ability, capacidad

method, método

future, futuro

equipment, equipo

disease, enfermedad

peace, paz

status, estado

variety, variedad

safety, seguridad

tea, té

weight, peso

sale, venta

afternoon, tarde

february, febrero

king, rey

saturday, sábado

university, universidad

kitchen, cocina

brother, hermano

principle, principio

pupil, alumno

duty, deber

county, condado

presence, presencia

truth, verdad

exchange, intercambio

august, augusto

marriage, matrimonio

failure, fracaso

career, carrera

horse, caballo

memory, memoria

skill, habilidad

politics, política

advantage, ventaja

officer, oficial

length, longitud

river, río

strength, fuerza

insurance, seguro

ball, bola

confidence, confianza

traffic, tráfico

sister, hermana

executive, ejecutivo

lady, dama

spirit, espíritu

relief, alivio

progress, progreso

earth, tierra

reality, realidad

tree, árbol

firm, firma

difficulty, dificultad

past, pasado

assessment, valoración

meaning, significado

path, camino

aircraft, aeronave

background, fondo

official, oficial

freedom, libertad

requirement, requisito

user, usuario

employee, empleado

weather, tiempo

sentence, frase

card, tarjeta

commitment, compromiso

victory, victoria

colleague, colega

expenditure, gasto

bird, pájaro

reaction, reacción

neck, cuello

thousand, mil

congress, congreso

appearance, apariencia

threat, amenaza

assembly, montaje

leg, pierna

player, jugador

membership, afiliación

payment, pago

institute, instituto

faith, fe

island, isla

driver, conductor

entry, entrada

breath, aliento

birth, nacimiento

wood, madera

belief, creencia

coal, carbón

gentleman, caballero

lip, labio

hell, infierno

newspaper, periódico

exhibition, exposición

treaty, tratado

engine, motor

sky, cielo

definition, definición

acid, ácido

atmosphere, ambiente

examination, examen

troop, tropa

debt, deuda

middle, medio

reduction, reducción

criticism, crítica

tooth, diente

visitor, visitante

creation, creación

christmas, navidad

forest, bosque

reader, lector

flight, vuelo

engineering, ingeniería

farmer, agricultor

museum, museo

factory, fábrica

injury, lesión

desk, escritorio

hole, agujero

noise, ruido

customer, cliente

wednesday, miércoles

youth, juventud

objective, objetivo

assistance, asistencia

nose, nariz

meal, comida

agency, agencia

beauty, belleza

enterprise, empresa

employer, empleador

roof, techo

grass, hierba

revolution, revolución

gold, oro

second, segundo

vehicle, vehículo

revenue, ingresos

artist, artista

plastic, plástico

housing, viviendas

location, ubicación

hundred, cien

song, canción

weapon, arma

identity, identidad

countryside, campo

autumn, otoño

criteria, criterios

character, carácter

theatre, teatro

framework, marco

chest, pecho

theme, tema

writer, escritor

wealth, riqueza

height, altura

decade, década

thursday, jueves

resistance, resistencia

reputation, reputación

democracy, democracia

offense, delito

wage, salario

recession, recesión

candidate, candidato

stair, escalera

shoe, zapato

outcome, resultado

foundation, fundación

delivery, entrega

ministry, ministerio

meat, carne

victim, víctima

parish, parroquia

resolution, resolución

hill, colina

furniture, muebles

efficiency, eficiencia

mountain, montaña

gun, pistola

kingdom, reino

regime, régimen

resident, residente

arrival, llegada

plane, plano

enemy, enemigo

minority, minoría

queen, reina

guest, invitado

adult, adulto

politician, político

defendant, demandado

captain, capitán

priority, prioridad

category, categoría

paintings, pinturas

premise, premisa

findings, hallazgos

expectation, expectativa

mood, humor

retirement, jubilación

citizen, ciudadano

darkness, oscuridad

sugar, azúcar

illness, enfermedad

throat, garganta

component, componente

professor, profesor

judgement, juicio

tool, herramienta

partnership, asociación

earnings, ganancias

neighbor, vecino

poverty, pobreza

mixture, mezcla

emergency, emergencia

entrance, entrada

currency, moneda

assumption, asunción

winner, ganador

sheep, oveja

territory, territorio

potential, potencial

ear, oído

core, núcleo

shareholder, accionista

depth, profundidad

savings, ahorros

soul, alma

servant, criado

symptom, síntoma

prisoner, preso

jacket, chaqueta

tendency, tendencia

mechanism, mecanismo

charity, caridad

economics, economía

landscape, paisaje

valley, valle

leisure, ocio

origin, origen

circle, círculo

column, columna

fee, cuota

childhood, infancia

climate, clima

cake, pastel

tank, tanque

frequency, frecuencia

disaster, desastre

evaluation, evaluación

wing, ala

guitar, guitarra

critic, crítico

poetry, poesía

laboratory, laboratorio

knife, cuchillo

republic, república

castle, castillo

bath, baño

universe, universo

certificate, certificado

buyer, comprador

paragraph, párrafo

acceptance, aceptación

gift, regalo

openings, aberturas

surgery, cirugía

shirt, camisa

knee, rodilla

cattle, ganado

anxiety, ansiedad

leather, cuero

tenant, inquilino

flesh, carne

percentage, porcentaje

summit, cumbre

egg, huevo

trend, tendencia

equation, ecuación

cheese, queso

topic, tema

soccer, fútbol

A.3.2 French

people, personnes

year, an

day, jour

man, homme

world, monde

work, travail

life, vie

children, enfants

case, cas

thing, chose

woman, femme

money, argent

fact, fait

night, nuit

area, région

company, entreprise

family, famille

business, entreprise

side, côté

week, semaine

country, pays

council, conseil

room, chambre

member, membre

problem, problème

car, voiture

office, bureau

door, porte

body, corps

person, personne

month, mois

health, santé

law, droit

word, mot

child, enfant

society, société

market, marché

job, emploi

process, processus

effect, effet

community, communauté

evidence, preuve

minister, ministre

morning, matin

level, niveau

death, décès

industry, industrie

century, siècle

church, église

history, histoire

road, route

center, centre

food, nourriture

program, programme

result, résultat

hour, heure

committee, comité

team, équipe

course, cours

language, langue

mind, esprit

authority, autorité

data, données

class, classe

paper, papier

wife, femme

city, ville

friend, ami

price, prix

god, dieu

town, ville

bed, lit

girl, fille

quality, qualité

music, musique

game, jeu

april, avril

student, étudiant

june, juin

hair, cheveux

basis, base

series, série

bank, banque

foot, pied

south, sud

west, ouest

secretary, secrétaire

security, sécurité

manager, directeur

heart, cœur

story, histoire

letter, lettre

chapter, chapitre

field, domaine

movement, mouvement

success, succès

analysis, analyse

news, nouvelles

evening, soirée

boy, garçon

theory, théorie

approach, approche

growth, croissance

agreement, accord

size, taille

son, fils

space, espace

property, propriété

example, exemple

energy, énergie

sir, monsieur

east, est

income, revenu

buildings, bâtiments

treatment, traitement

july, juillet

north, nord

loss, perte

activity, activité

march, mars

army, armée

summer, été

product, produit

wall, mur

teacher, professeur

unit, unité

technology, technologie

october, octobre

window, fenêtre

resource, ressource

sea, mer

september, septembre

event, événement

january, janvier

goods, biens

blood, sang

extent, ampleur

response, réponse

majority, majorité

degree, degré

economy, économie

glass, verre

glasses, lunettes

street, rue

parliament, parlement

labor, travail

species, espèce

title, titre

employment, emploi

daughter, fille

competition, concurrence

november, novembre

december, décembre

sunday, dimanche

purpose, but

task, tâche

ability, capacité

method, méthode

future, avenir

equipment, équipement

disease, maladie

peace, paix

status, statut

variety, variété

safety, sécurité

tea, thé

weight, poids

sale, vente

afternoon, après-midi

february, février

king, roi

saturday, samedi

university, université

kitchen, cuisine

brother, frère

principle, principe

pupil, élève

duty, devoir

county, comté

presence, présence

truth, vérité

exchange, échange

august, août

marriage, mariage

failure, échec

career, carrière

horse, cheval

memory, mémoire

skill, compétence

politics, politique

advantage, avantage

officer, officier

length, longueur

river, rivière

strength, force

insurance, assurance

ball, balle

confidence, confiance

traffic, trafic

sister, sœur

executive, exécutif

lady, dame

spirit, esprit

relief, soulagement

progress, progrès

earth, terre

reality, réalité

tree, arbre

firm, entreprise

difficulty, difficulté

past, passé

assessment, évaluation

meaning, sens

path, chemin

aircraft, avion

background, fond

official, officiel

freedom, liberté

requirement, exigence

user, utilisateur

employee, employé

weather, temps

sentence, phrase

card, carte

commitment, engagement

victory, victoire

colleague, collègue

expenditure, dépense

bird, oiseau

reaction, réaction

neck, cou

thousand, mille

congress, congrès

appearance, apparence

threat, menace

assembly, assemblage

leg, jambe

player, joueur

membership, adhésion

payment, paiement

institute, institut

faith, foi

island, île

driver, conducteur

entry, entrée

breath, souffle

birth, naissance

wood, bois

belief, croyance

coal, charbon

gentleman, gentilhomme

lip, lèvre

hell, enfer

newspaper, journal

exhibition, exposition

treaty, traité

engine, moteur

sky, ciel

definition, définition

acid, acide

atmosphere, atmosphère

examination, examen

troop, troupe

debt, dette

middle, milieu

reduction, réduction

criticism, critique

tooth, dent

visitor, visiteur

creation, création

christmas, noël

forest, forêt

reader, lecteur

flight, vol

engineering, ingénierie

farmer, agriculteur

museum, musée

factory, usine

injury, blessure

desk, bureau

hole, trou

noise, bruit

customer, client

wednesday, mercredi

youth, jeunes

objective, objectif

assistance, aide

nose, nez

meal, repas

agency, agence

beauty, beauté

enterprise, entreprise

employer, employeur

roof, toit

grass, herbe

revolution, révolution

gold, or

second, deuxième

vehicle, véhicule

revenue, revenu

artist, artiste

plastic, plastique

housing, logement

location, emplacement

hundred, cent

song, chanson

weapon, arme

identity, identité

countryside, campagne

autumn, automne

criteria, critères

character, caractère

theatre, théâtre

framework, cadre

chest, poitrine

theme, thème

writer, écrivain

wealth, richesse

height, hauteur

decade, décennie

thursday, jeudi

resistance, résistance

reputation, réputation

democracy, démocratie

offense, infraction

wage, salaire

recession, récession

candidate, candidat

stair, marche

shoe, chaussure

outcome, résultat

foundation, fondation

delivery, livraison

ministry, ministère

meat, viande

victim, victime

parish, paroisse

resolution, résolution

hill, colline

furniture, meubles

efficiency, efficacité

mountain, montagne

gun, pistolet

kingdom, royaume

regime, régime

resident, résident

arrival, arrivée

plane, avion

enemy, ennemi

minority, minorité

queen, reine

guest, invité

adult, adulte

politician, politicien

defendant, défendeur

captain, capitaine

priority, priorité

category, catégorie

paintings, peintures

premise, prémisse

findings, résultats

expectation, attente

mood, humeur

retirement, retraite

citizen, citoyen

darkness, obscurité

sugar, sucre

illness, maladie

throat, gorge

component, composant

professor, professeur

judgement, jugement

tool, outil

partnership, partenariat

earnings, bénéfices

neighbor, voisin

poverty, pauvreté

mixture, mélange

emergency, urgence

entrance, entrée

currency, monnaie

assumption, hypothèse

winner, gagnant

sheep, mouton

territory, territoire

potential, potentiel

ear, oreille

core, noyau

shareholder, actionnaire

depth, profondeur

savings, épargnes

soul, âme

servant, serviteur

symptom, symptôme

prisoner, prisonnier

jacket, veste

tendency, tendance

mechanism, mécanisme

charity, charité

economics, économie

landscape, paysage

valley, vallée

leisure, loisir

origin, origine

circle, cercle

column, colonne

fee, frais

childhood, enfance

climate, climat

cake, gâteau

tank, réservoir

frequency, fréquence

disaster, catastrophe

evaluation, évaluation

wing, aile

guitar, guitare

critic, critique

poetry, poésie

laboratory, laboratoire

knife, couteau

republic, république

castle, château

bath, bain

universe, univers

certificate, certificat

buyer, acheteur

paragraph, paragraphe

acceptance, acceptation

gift, cadeau

openings, ouvertures

surgery, chirurgie

shirt, chemise

knee, genou

cattle, bétail

anxiety, anxiété

leather, cuir

tenant, locataire

flesh, chair

percentage, pourcentage

summit, sommet

egg, œuf

trend, tendance

equation, équation

cheese, fromage

topic, sujet

soccer, football

Appendix B

Materials for the Multi-System Deployment

B.1 Pre-Study Questionnaire

Please answer these questions with respect to how you generally felt in the past 2 weeks while doing the following activities: (1: very low, 7: very high)

- How irritated, stressed, insecure, discouraged, or annoyed were you while waiting for and riding the elevator?
- How hurried or rushed was the pace of waiting for and riding the elevator?
- How hard did you have to work to accomplish your level of performance of waiting for and riding the elevator?
- How irritated, stressed, insecure, discouraged, or annoyed were you while waiting for wifi on your laptop?
- How hurried or rushed was the pace of waiting for wifi on your laptop?
- How hard did you have to work to accomplish your level of performance of waiting for wifi on your laptop?
- How irritated, stressed, insecure, discouraged, or annoyed were you while fetching

and reading email on your phone?

- How hurried or rushed was the pace of fetching and reading email on your phone?
- How hard did you have to work to accomplish your level of performance of fetching and reading email on your phone?
- How irritated, stressed, insecure, discouraged, or annoyed were you while sending email on your computer (at gmail.com)?
- How hurried or rushed was the pace of sending email on your computer (at gmail.com)?
- How hard did you have to work to accomplish your level of performance of sending email on your computer (at gmail.com)?
- How irritated, stressed, insecure, discouraged, or annoyed were you while instant messaging on GChat (at gmail.com)?
- How hurried or rushed was the pace of instant messaging on GChat (at gmail.com)?
- How hard did you have to work to accomplish your level of performance of instant messaging on GChat (at gmail.com)?

B.2 Post-Study Questionnaire

- How irritated, stressed, insecure, discouraged, or annoyed were you while waiting for and riding the elevator in combination with doing learning exercises on your iphone app?
- How hurried or rushed was the pace of waiting for and riding the elevator in combination with doing learning exercises on your iphone app?
- How hard did you have to work to accomplish your level of performance of waiting for and riding the elevator in combination with doing learning exercises on your iphone app?
- How respectful of your activity (waiting for and riding the elevator) were the learning exercises in your iphone app?
- How irritated, stressed, insecure, discouraged, or annoyed were you while waiting for wifi on your laptop in combination with doing learning exercses?

- How hurried or rushed was the pace of waiting for wifi on your laptop in combination with doing learning exercises?
- How hard did you have to work to accomplish your level of performance of waiting for wifi on your laptop in combination with doing learning exercises?
- How respectful of your activity (waiting for wifi on your laptop) were the learning exercises?
- How irritated, stressed, insecure, discouraged, or annoyed were you while fetching and reading email on your phone in combination with doing learning exercises in K9?
- How hurried or rushed was the pace of fetching and reading email on your phone in combination with doing learning exercises in K9?
- How hard did you have to work to accomplish your level of performance of fetching and reading email on your phone in combination with doing learning exercises in K9?
- How respectful of your activity (fetching and reading email on your phone) were the learning exercises in K9?
- How irritated, stressed, insecure, discouraged, or annoyed were you while sending email on your computer (at gmail.com) in combination with doing learning exercises?
- How hurried or rushed was the pace of sending email on your computer (at gmail.com) in combination with doing learning exercises?
- How hard did you have to work to accomplish your level of performance of sending email on your computer (at gmail.com) in combination with doing learning exercises?
- How respectful of your activity (sending email on your computer at gmail.com) were the learning exercises?
- How irritated, stressed, insecure, discouraged, or annoyed were you while instant messaging on GChat (at gmail.com) in combination with doing learning exercises?
- How hurried or rushed was the pace of instant messaging on GChat (at gmail.com) in combination with doing learning exercises?
- How hard did you have to work to accomplish your level of performance of instant messaging on GChat (at gmail.com) in combination with doing learning exercises?
- How respectful of your activity (instant messaging on GChat at gmail.com) were the learning exercises?

Appendix C

Materials for the Long-term Study

C.1 Weekly Mental Workload Survey

- How mentally demanding was GChatting? (1: very low, 7: very high)
- How irritated, stressed, and annoyed were you while GChatting? (1: very low, 7: very high)

C.2 Post-Study Questionnaire

- I enjoyed using in-chat flashcards. (1: strongly disagree, 7: strongly agree)
- I enjoyed using the flashcard website + email reminders. (1: strongly disagree, 7: strongly agree)
- I found it annoying when I got in-chat flashcards. (1: strongly disagree, 7: strongly agree)
- I found it annoying when I got flashcard website email reminders. (1: strongly disagree, 7: strongly agree)
- It was easy to find time to learn vocabulary using in-chat flashcards. (1: strongly disagree, 7: strongly agree)
- It was easy to find time to learn vocabulary using the flashcard website + email

reminders. (1: strongly disagree, 7: strongly agree)

- Using in-chat flashcards, I would engage in vocabulary practice more often than before the study. (1: strongly disagree, 7: strongly agree)
- Using the flashcard website + email reminders, I would engage in vocabulary practice more often than before the study. (1: strongly disagree, 7: strongly agree)
- In the future, I would like to continue using in-chat flashcards. (1: strongly disagree, 7: strongly agree)
- In the future, I would like to continue using the flashcard website + email reminders. (1: strongly disagree, 7: strongly agree)

Overall, which did you prefer?

- in-chat flashcards
- flashcard website email reminders
- no preference

Bibliography

[1] D. S. Ackerman and B. L. Gross. My instructor made me do it: Task characteristics of procrastination. *Journal of Marketing education*, 27(1):5–13, 2005. 46, 156

[2] P. D. Adamczyk and B. P. Bailey. If not now, when?: the effects of interruption at different moments within task execution. In *CHI '04*, pages 271–278. ACM, 2004. 39, 46

[3] P. S. Adler, B. Goldoftas, and D. I. Levine. Flexibility versus efficiency? a case study of model changeovers in the toyota production system. 2000. 46

[4] D. Allen. *Getting things done: The art of stress-free productivity*. Penguin, 2015. 45, 156

[5] F. Alt, A. Sahami Shirazi, A. Schmidt, and R. Atterer. Bridging waiting times on web pages. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 305–308. ACM, 2012. 42

[6] E. M. Altmann and J. G. Trafton. Task interruption: Resumption lag and the role of cues. Technical report, DTIC Document, 2004. 53

[7] J. R. Anderson. Human symbol manipulation within an integrated cognitive architecture. *Cognitive science*, 29(3):313–341, 2005. 41

[8] D. Avrahami and S. E. Hudson. Communication characteristics of instant messaging:

effects and predictions of interpersonal relationships. In *CSCW '06*, pages 505–514. ACM, 2006. 83

[9] D. Avrahami and S. E. Hudson. Responsiveness in instant messaging: predictive models supporting inter-personal communication. In *CHI '06*, pages 731–740. ACM, 2006. 118

[10] D. Avrahami, S. R. Fussell, and S. E. Hudson. Im waiting: timing and responsiveness in semi-synchronous communication. In *CSCW '08*, pages 285–294. ACM, 2008. 68, 73, 76, 80

[11] A. D. Baddeley. *Human memory: Theory and practice*. Psychology Press, 1997. 69

[12] H. P. Bahrick, L. E. Bahrick, A. S. Bahrick, and P. E. Bahrick. Maintenance of foreign language vocabulary and the spacing effect. *Psychological Science*, 4(5):316–321, 1993. 36

[13] B. P. Bailey and S. T. Iqbal. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *TOCHI*, 14(4):21, 2008. 39, 40, 55, 81

[14] B. P. Bailey, J. A. Konstan, and J. V. Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Proceedings of INTERACT*, volume 1, pages 593–601, 2001. 55

[15] S. Bakker, E. van den Hoven, and B. Eggen. Peripheral interaction: characteristics and considerations. *Personal and Ubiquitous Computing*, 19(1):239–254, 2015. 43, 55, 80

[16] A. Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Advances in Behaviour Research and Therapy*, 1(4):139–161, 1978. 46

[17] L. F. Barrett and D. J. Barrett. An introduction to computerized experience sampling in psychology. *Social Science Computer Review*, 19(2):175–185, 2001. 45

[18] J. S. Beaudin, S. S. Intille, E. M. Tapia, R. Rockinson, and M. E. Morris. Context-sensitive microlearning of foreign language vocabulary on a mobile device. In *Ambient Intelligence*, pages 55–72. Springer, 2007. 37

[19] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2001. 47

[20] F. Bentley and K. Tollmar. The power of mobile notifications to increase wellbeing logging behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1095–1098. ACM, 2013. 44, 111, 117

[21] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. *Communications of the ACM*, 58(8):85–94, 2015. 45, 46, 140

[22] J. P. Bigham. Making the web easier to see with opportunistic accessibility improvement. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 117–122. ACM, 2014. 47

[23] M. Böhmer, C. Lander, S. Gehring, D. P. Brumby, and A. Krüger. Interrupted by a phone call: exploring designs for lowering the impact of call notifications for smartphone users. In *CHI '14*, pages 3045–3054. ACM, 2014. 54, 85

[24] J. P. Borst, N. A. Taatgen, and H. van Rijn. The problem state: a cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, memory, and cognition*, 36(2):363, 2010. 41

[25] J. P. Borst, N. A. Taatgen, and H. van Rijn. What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2971–2980. ACM, 2015. 40, 53, 55, 148, 157

[26] D. Broadbent. Perception mid communication, 1958. 39

[27] K. S. Burns and R. J. Lutz. The function of format: Consumer responses to six on-line advertising formats. *Journal of Advertising*, 35(1):53–63, 2006. 166

[28] J. A. Bzostek and M. S. Wogalter. Measuring visual search time for a product warning label as a function of icon, color, column and vertical placement. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 43, pages 888–892. SAGE Publications, 1999. 189

[29] C. J. Cai, P. J. Guo, J. Glass, and R. C. Miller. Wait-learning: leveraging conversational dead time for second language education. In *CHI'14 Extended Abstracts*, pages 2239–2244. ACM, 2014. 55, 62

[30] C. J. Cai, P. J. Guo, J. Glass, and R. C. Miller. Wait-learning: leveraging wait time for education. In *CHI'14*. ACM, 2015. 137

[31] N. J. Cepeda, H. Pashler, E. Vul, J. T. Wixted, and D. Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132 (3):354, 2006. 36

[32] P. Chatterjee. Are unclicked ads wasted? enduring effects of banner and pop-up ad exposures on brand memory and attitudes. *Journal of electronic commerce Research*, 9(1):51, 2008. 166

[33] J. Cheng, J. Teevan, S. T. Iqbal, and M. S. Bernstein. Break it down: A comparison of macro-and microtasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4061–4064. ACM, 2015. 45

[34] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011. 67

[35] S. Consolvo, D. W. McDonald, and J. A. Landay. Theory-driven design strategies

for technologies that support behavior change in everyday life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 405–414. ACM, 2009. 43, 193

[36] A. R. Conway, N. Cowan, and M. F. Bunting. The cocktail party phenomenon revisited: The importance of working memory capacity. *Psychonomic bulletin & review*, 8(2):331–335, 2001. 39

[37] F. I. Craik and R. S. Lockhart. Levels of processing: A framework for memory research. *Journal of verbal learning and verbal behavior*, 11(6):671–684, 1972. 137, 141

[38] M. Csikszentmihalyi. Flow: The psychology of optimal experience," new yori: Harper and row. 1990. 54

[39] P. Dai, J. M. Rzeszotarski, P. Paritosh, and E. H. Chi. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 628–638. ACM, 2015. 46, 137

[40] D. Dearman and K. Truong. Evaluating the implicit acquisition of second language vocabulary using a live wallpaper. In *CHI '12*, pages 1391–1400. ACM, 2012. 37

[41] B. G. Dellaert and B. E. Kahn. How tolerable is delay?: Consumers? evaluations of internet web sites after waiting. *Journal of interactive marketing*, 13(1):41–54, 1999. 42

[42] F. N. Dempster. Effects of variable encoding and spaced presentations on vocabulary learning. *Journal of Educational Psychology*, 79(2):162, 1987. 36, 52

[43] F. N. Dempster. The spacing effect: A case study in the failure to apply the results of psychological research. *American Psychologist*, 43(8):627, 1988. 36

[44] F. N. Dempster. Distributing and managing the conditions of encoding and practice. *Memory*, 10:317–344, 1996. 36

[45] J. A. Deutsch and D. Deutsch. Attention: Some theoretical considerations. *Psychological review*, 70(1):80, 1963. 39

[46] M. Dixon and J. Fogarty. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1525–1534. ACM, 2010. 193

[47] Z. Dornyei and I. Ottó. Motivation in action: A process model of l2 motivation. *Working Papers in Applied Linguistics*, pages 43–69, 1998. 38, 44

[48] S. Dow, A. Kulkarni, S. Klemmer, and B. Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1013–1022. ACM, 2012. 45

[49] J. Dunlosky and T. O. Nelson. Does the sensitivity of judgments of learning (jols) to the effects of various study activities depend on when the jols occur? *Journal of Memory and Language*, 33(4):545, 1994. 36

[50] H. Ebbinghaus. *Memory: A contribution to experimental psychology*. Number 3. Teachers college, Columbia university, 1913. 36, 69

[51] D. Edge and A. F. Blackwell. Peripheral tangible interaction. In *Peripheral Interaction*, pages 65–93. Springer, 2016. 43

[52] D. Edge, E. Searle, K. Chiu, J. Zhao, and J. A. Landay. Micromandarin: mobile language learning in context. In *CHI '11*, pages 3169–3178. ACM, 2011. 37, 38, 67

[53] D. Edge, S. Fitchett, M. Whitney, and J. Landay. Memreflex: adaptive flashcards for mobile microlearning. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 431–440. ACM, 2012. 36, 69, 107, 117

[54] R. A. Emmons and M. E. McCullough. Counting blessings versus burdens: an experimental investigation of gratitude and subjective well-being in daily life. *Journal of personality and social psychology*, 84(2):377, 2003. 45

[55] J. R. Ferrari, J. L. Johnson, and W. G. McCown. Procrastination research. In *Procrastination and Task Avoidance*, pages 21–46. Springer, 1995. 46

[56] L. Flower. Writer-based prose: A cognitive basis for problems in writing. *College English*, 41(1):19–37, 1979. 46, 138

[57] B. J. Fogg. A behavior model for persuasive design. In *Proceedings of the 4th international Conference on Persuasive Technology*, page 40. ACM, 2009. 38, 44, 133

[58] J. Forlizzi, I. Li, and A. Dey. Ambient interfaces that motivate changes in human behavior. In *Workshop at Pervasive 2007 Designing and Evaluating Ambient Information Systems*, page 6. Citeseer, 2007. 43

[59] G. Gassler, T. Hug, and C. Glahn. Integrated micro learning–an outline of the basic method and first results. *Interactive Computer Aided Learning*, 4, 2004. 35, 38

[60] T. Gillie and D. Broadbent. What makes interruptions disruptive? a study of length, similarity, and complexity. *Psychological research*, 50(4):243–250, 1989. 40

[61] R. Godwin-Jones. Emerging technologies from memory palaces to spacing algorithms: approaches to secondlanguage vocabulary learning. *Language, Learning & Technology*, 14(2), 2010. 36, 69

[62] W. Graves, J. Binder, M. Seidenberg, and R. Desai. “neural correlates of semantic processing in reading aloud." in m. faust (ed.), handbook of the neuropsychology of language. malden, ma: Wiley-blackwell. 144

[63] S. W. Greenwald, M. Khan, C. D. Vazquez, and P. Maes. Tagalong: Informal learning

from a remote companion with mobile perspective sharing. *International Association for Development of the Information Society*, 2015. 37

[64] N. Greer, J. Teevan, and S. T. Iqbal. An introduction to technological support for writing. Technical report, Technical Report. Microsoft Research Tech Report MSR-TR-2016-001, 2016. 138

[65] A. Gupta, W. Thies, E. Cutrell, and R. Balakrishnan. mclerk: enabling mobile crowdsourcing in developing regions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1843–1852. ACM, 2012. 45

[66] C. Harrison, B. Amento, S. Kuznetsov, and R. Bell. Rethinking the progress bar. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 115–118. ACM, 2007. 42

[67] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988. 142

[68] S. Hidi and P. Boscolo. Motivation and writing. *Handbook of writing research*, 144: 157, 2006. 46, 138

[69] J. J. Horton and L. B. Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 209–218. ACM, 2010. 137

[70] M. K. Hul, L. Dube, and J.-C. Chebat. The impact of music on consumers' reactions to waiting for services. *Journal of Retailing*, 73(1):87–104, 1997. 42

[71] S. S. Intille. Change blind information display for ubiquitous computing environments. In *International Conference on Ubiquitous Computing*, pages 91–106. Springer, 2002. 43

[72] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010. 138

[73] S. T. Iqbal and B. P. Bailey. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI'05 Extended Abstracts*, pages 1489–1492. ACM, 2005. 88

[74] S. T. Iqbal and B. P. Bailey. Effects of intelligent notification management on users and their tasks. In *CHI '08*, pages 93–102. ACM, 2008. 46

[75] S. T. Iqbal and E. Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *CHI '07*, pages 677–686. ACM, 2007. 40, 129, 134, 137

[76] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm. The character, functions, and styles of instant messaging in the workplace. In *CSCW '02*, pages 11–20. ACM, 2002. 68, 118

[77] E. W. Ishak and S. K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 189–192. ACM, 2004. 47

[78] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 234–241. ACM, 1997. 43

[79] N. Jafarinaimi, J. Forlizzi, A. Hurst, and J. Zimmerman. Breakaway: an ambient display designed to change human behavior. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1945–1948. ACM, 2005. 43, 166

[80] J. Jin and L. A. Dabbish. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1799–1808. ACM, 2009. 38, 41, 42, 54

[81] H. Jordan and S. P. Tipper. Object-based inhibition of return in static displays. *Psychonomic Bulletin & Review*, 5(3):504–509, 1998. 55

[82] D. Kahneman. *Attention and effort*. Citeseer, 1973. 39, 109

[83] J. D. Karpicke and J. R. Blunt. Retrieval practice produces more learning than elaborative studying with concept mapping. *Science*, 331(6018):772–775, 2011. 37

[84] K. L. Katz, B. M. Larson, and R. C. Larson. Prescription for the waiting-in-line blues: Entertain, enlighten, and engage. *MIT Sloan Management Review*, 32(2):44, 1991. 42

[85] J. Kim, M. Dontcheva, W. Li, M. S. Bernstein, and D. Steinsapir. Motif: Supporting novice creativity through expert patterns. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1211–1220. ACM, 2015. 45

[86] N. Kornell. Optimising learning using flashcards: Spacing is more effective than cramming. *Applied Cognitive Psychology*, 23(9):1297–1317, 2009. 36

[87] N. Kornell and R. A. Bjork. Learning concepts and categories is spacing the "enemy of induction"? *Psychological science*, 19(6):585–592, 2008. 36

[88] G. Kovacs. Feedlearn: Using facebook feeds for microlearning. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1461–1466. ACM, 2015. 38

[89] A. Lang. Using the limited capacity model of motivated mediated message processing to design effective cancer communication messages. *Journal of Communication*, 56 (s1):S57–S80, 2006. 41, 109

[90] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham. Real-time captioning by groups of non-experts. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 23–34. ACM, 2012. 45

[91] W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein. Apparition: Crowdsourced user interfaces that come to life as you sketch them. In

Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pages 1925–1934. ACM, 2015. 45

[92] W. S. Lasecki, J. M. Rzeszotarski, A. Marcus, and J. P. Bigham. The effects of sequence and delay on crowd work. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1375–1378. ACM, 2015. 45, 137, 138

[93] J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991. 85

[94] H.-C. Lin and C. Raghavendra. An approximate analysis of the join the shortest queue (jsq) policy. *Parallel and Distributed Systems, IEEE Transactions on*, 7(3): 301–307, 1996. 42

[95] K. Lin, H. Spindell, S. Cambo, Y. Kim, and H. Zhang. Habitsourcing: Sensing the environment through immersive, habit-building experiences. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 639–650. ACM, 2016. 45

[96] R. Ling and N. S. Baron. Text messaging and im linguistic comparison of american college data. *Journal of Language and Social Psychology*, 26(3):291–298, 2007. 68, 81

[97] U. Lundberg, M. Granqvist, T. Hansson, M. Magnusson, and L. Wallin. Psychological and physiological stress responses during repetitive work at an assembly line. *Work & Stress*, 3(2):143–153, 1989. 46

[98] D. H. Maister. *The psychology of waiting lines*. Harvard Business School, 1984. 42, 77, 109

[99] G. Mark, D. Gudith, and U. Klocke. The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 107–110. ACM, 2008. 40

[100] G. Mark, S. Iqbal, M. Czerwinski, and P. Johns. Focused, aroused, but so distractible: Temporal perspectives on multitasking and communications. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 903–916. ACM, 2015. 41

[101] T. Matthews, T. Rattenbury, S. Carter, A. Dey, and J. Mankoff. A peripheral display toolkit. *University of California, Berkeley Technotes, UCB//CSD-03-1258*, 2003. 168

[102] A. A. Mitchell. Involvement: a potentially important mediator of consumer behavior. *NA-Advances in Consumer Research Volume 06*, 1979. 142

[103] Y. Miyata and D. A. Norman. Psychological issues in support of multiple activities. *User centered system design: New perspectives on human-computer interaction*, pages 265–284, 1986. 39, 81

[104] S. Monsell. Task switching. *Trends in cognitive sciences*, 7(3):134–140, 2003. 40, 53, 109

[105] R. R. Morris, M. Dontcheva, and E. M. Gerber. Priming for better performance in microtask crowdsourcing environments. *IEEE Internet Computing*, 16(5):13–19, 2012. 46

[106] H. Müller, M. Pielot, and R. de Oliveira. Towards ambient notifications. *Peripheral Interaction: Embedding HCI in Everyday Life*, 21, 2013. 43

[107] H. Müller, A. Kazakova, W. Heuten, and S. Boll. Supporting efficient task switching in a work environment with a pervasive display. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays*, pages 13–19. ACM, 2016. 43

[108] A. Naamad, D. Lee, and W.-L. Hsu. On the maximum empty rectangle problem. *Discrete Applied Mathematics*, 8(3):267–277, 1984. 171

[109] F. F.-H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004. 42

[110] T. Nakajima, V. Lehdonvirta, E. Tokunaga, and H. Kimura. Reflecting human behavior to motivate desirable lifestyle. In *Proceedings of the 7th ACM conference on Designing interactive systems*, pages 405–414. ACM, 2008. 166

[111] B. A. Nardi, S. Whittaker, and E. Bradner. Interaction and outeraction: instant messaging in action. In *CSCW '00*, pages 79–88. ACM, 2000. 54, 75

[112] P. Narula, P. Gutheim, D. Rolnitzky, A. Kulkarni, and B. Hartmann. Mobileworks: A mobile crowdsourcing platform for workers at the bottom of the pyramid. *Human Computation*, 11:11, 2011. 45

[113] M. Nebeling, A. To, A. Guo, A. A. de Freitas, J. Teevan, S. P. Dow, and J. P. Bigham. Wearwrite: Crowd-assisted writing from smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3834–3846. ACM, 2016. 45, 46

[114] S. L. Neuberg, D. T. Kenrick, and M. Schaller. Human threat management systems: Self-protection and disease avoidance. *Neuroscience & Biobehavioral Reviews*, 35 (4):1042–1051, 2011. 41

[115] C. M. Neuwirth, D. S. Kaufer, R. Chandhok, and J. H. Morris. Computer support for distributed collaborative writing: A coordination science perspective. *Coordination Theory and Collaboration Technology, ed. GM Olson, TW Malone, and JB Smith, Lawrence Erlbaum Associates, New Jersey*, 2001. 45, 46

[116] S. Niida, S. Uemura, H. Nakamura, and E. Harada. Field study of a waiting-time filler delivery system. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 177–180. ACM, 2011. 42

[117] M. C. Panyan and R. V. Hall. Effects of serial versus concurrent task sequencing on acquisition, maintenance, and generalization. *Journal of Applied Behavior Analysis*, 11(1):67–74, 1978. 45, 159

[118] J.-H. Park and H. J. Choi. Factors influencing adult learners' decision to drop out or persist in online learning. *Journal of Educational Technology & Society*, 12(4): 207–217, 2009. 25

[119] H. Pashler, D. Rohrer, N. J. Cepeda, and S. K. Carpenter. Enhancing learning and retarding forgetting: Choices and consequences. *Psychonomic Bulletin & Review*, 14 (2):187–193, 2007. 121

[120] N. Pattyn, X. Neyt, D. Henderickx, and E. Soetens. Psychophysiological investigation of vigilance decrement: boredom or cognitive fatigue? *Physiology & Behavior*, 93 (1):369–378, 2008. 137

[121] P. I. Pavlik and J. R. Anderson. Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, 14(2):101, 2008. 107

[122] P. Pimsleur. A memory schedule. *Modern Language Journal*, pages 73–75, 1967. 36

[123] H. Pohl and R. Murray-Smith. Focused and casual interactions: allowing users to vary their level of engagement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2223–2232. ACM, 2013. 43, 196

[124] T. Porter and T. Duff. Compositing digital images. In *ACM Siggraph Computer Graphics*, volume 18, pages 253–259. ACM, 1984. 47

[125] I. R. Posner and R. M. Baecker. How people write together (groupware). In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume 4, pages 127–138. IEEE, 1992. 45

[126] Z. Pousman and J. Stasko. A taxonomy of ambient information systems: four patterns of design. In *Proceedings of the working conference on Advanced visual interfaces*, pages 67–74. ACM, 2006. 42, 168

[127] B. A. Rafoth and D. L. Rubin. The impact of content and mechanics on judgments of writing quality. *Written Communication*, 1(4):446–458, 1984. 140

[128] A. Ren. Pull-to-refresh and learn: Leveraging mobile email load time for education. In *CHI'15 Extended Abstracts*. ACM, 2015. 63

[129] R. A. Rensink, J. Kevin O'Regan, and J. J. Clark. On the failure to detect changes in scenes across brief interruptions. *Visual cognition*, 7(1-3):127–145, 2000. 43, 188

[130] Y. Rogers, W. R. Hazlewood, P. Marshall, N. Dalton, and S. Hertrich. Ambient influence: Can twinkly lights lure and abstract representations trigger behavioral change? In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 261–270. ACM, 2010. 166

[131] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin. Feature congestion: a measure of display clutter. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 761–770. ACM, 2005. 192

[132] R. Rosenholtz, Y. Li, and L. Nakano. Measuring visual clutter. *Journal of vision*, 7 (2):17–17, 2007. 192

[133] D. D. Salvucci and N. A. Taatgen. Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review*, 115(1):101, 2008. 40

[134] A. Schmidt. Implicit human computer interaction through context. *Personal technologies*, 4(2-3):191–199, 2000. 43

[135] D. W. Schneider and J. R. Anderson. Asymmetric switch costs as sequential difficulty effects. *The Quarterly Journal of Experimental Psychology*, 63(10):1873–1894, 2010. 152

[136] R. Senter and E. A. Smith. Automated readability index. Technical report, DTIC Document, 1967. 141

[137] D. J. Simons and R. A. Rensink. Change blindness: Past, present, and future. *Trends in cognitive sciences*, 9(1):16–20, 2005. 43, 188

[138] J. Surowiecki. *The wisdom of crowds*. Anchor, 2005. 45

[139] J. Sweller, J. J. Van Merrienboer, and F. G. Paas. Cognitive architecture and instructional design. *Educational psychology review*, 10(3):251–296, 1998. 37, 54, 55

[140] G. Taylor and P. Nightingale. Not mechanics but meaning: Error in tertiary students' writing. *Higher Education Research and Development*, 9(2):161–176, 1990. 140

[141] J. Teevan, D. J. Liebling, and W. S. Lasecki. Selfsourcing personal tasks. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 2527–2532. ACM, 2014. 45

[142] J. Teevan, S. T. Iqbal, and C. von Veh. Supporting collaborative writing with microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2657–2668. ACM, 2016. 45, 46

[143] A. M. Treisman. Contextual cues in selective listening. *Quarterly Journal of Experimental Psychology*, 12(4):242–248, 1960. 39

[144] A. Trusty and K. N. Truong. Augmenting the web for second language vocabulary learning. In *CHI '11*, pages 3179–3188. ACM, 2011. 38

[145] C. C. Tsai, G. Lee, F. Raab, G. J. Norman, T. Sohn, W. G. Griswold, and K. Patrick. Usability and feasibility of pmeb: a mobile phone application for monitoring real time caloric balance. *Mobile networks and applications*, 12(2-3):173–184, 2007. 45

[146] E. Tulving. The effects of presentation and recall of material in free-recall learning. *Journal of verbal learning and verbal behavior*, 6(2):175–184, 1967. 36

[147] R. Vaish, K. Wyngarden, J. Chen, B. Cheung, and M. S. Bernstein. Twitch crowdsourcing: crowd contributions in short bursts of time. In *CHI '14*, pages 3645–3654. ACM, 2014. 45, 159

[148] Z. Wang and J. M. Tchernev. The ?myth? of media multitasking: Reciprocal dynamics

of media multitasking, personal needs, and gratifications. *Journal of Communication*, 62(3):493–513, 2012. 41

[149] S. Webb. The effects of repetition on vocabulary knowledge. *Applied Linguistics*, 28 (1):46–65, 2007. 36

[150] C. Wickens. D.(1984). processing resources in attention. *Varieties of attention*, pages 63–102, 1984. 39

[151] C. D. Wickens, R. S. Gutzwiller, and A. Santamaria. Discrete task switching in overload: A meta-analyses and a model. *International Journal of Human-Computer Studies*, 79:79–84, 2015. 41

[152] T. B. Wilson. *Gradual awareness notification for the desktop environment*. PhD thesis, Massachusetts Institute of Technology, 2006. 55

[153] G. Wylie and A. Allport. Task switching and the measurement of switch costs. *Psychological research*, 63(3-4):212–233, 2000. 40, 53, 149

[154] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner. mcrowd: a platform for mobile crowdsourcing. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 347–348. ACM, 2009. 45

[155] M. Yin, Y. Chen, and Y.-A. Sun. Monetary interventions in crowdsourcing task switching. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014. 46

[156] J. L. Zaichkowsky. The personal involvement inventory: Reduction, revision, and application to advertising. *Journal of advertising*, 23(4):59–70, 1994. 142