WaitSuite: Productive Use of Diverse Waiting Moments

CARRIE J. CAI, MIT CSAIL
ANJI REN, MIT CSAIL
ROBERT C. MILLER, MIT CSAIL

The busyness of daily life makes it difficult to find time for informal learning. Yet, learning requires significant time and effort, with repeated exposures to educational content on a recurring basis. Despite the struggle to find time, there are numerous moments in a day that are typically wasted due to waiting, such as while waiting for the elevator to arrive, wifi to connect, or an instant message to arrive. We introduce the concept of *wait-learning*: automatically detecting wait time and inviting people to learn while waiting. Our approach is to design seamless interactions that augment existing wait time with productive opportunities. Combining wait time with productive work opens up a new class of software systems that overcome the problem of limited time.

In this paper, we establish a design space for wait-learning and explore this design space by creating WaitSuite, a suite of five different wait-learning apps that each uses a different kind of waiting. For one of these apps, we conducted a feasibility study to evaluate learning and to understand how exercises should be timed during waiting periods. Subsequently, we evaluated multiple kinds of wait-learning in a two-week field study of WaitSuite with 25 people. We present design implications for wait-learning, and a theoretical framework that describes how wait time, ease of accessing the learning task, and competing demands impact the effectiveness of wait-learning in different waiting scenarios. These findings provide insight into how wait-learning can be designed to minimize interruption to ongoing tasks and maximize engagement with learning.

Categories and Subject Descriptors: H.5.m. [**Information Interfaces and Presentation (e.g., HCI)**]: Miscellaneous

General Terms: User Interfaces; Design

Additional Key Words and Phrases: Micro-learning; wait-learning; microtasks; attention management; productivity

1. INTRODUCTION

Competing priorities in daily life make it difficult for those with a casual interest in learning to find time for practice. For those who would like to learn a foreign language or study for a standardized exam, the busyness of daily life makes it challenging to schedule regular time for these activities. Despite having the initial desire or *choice motivation* to learn, learners often lack the *executive motivation* to actually practice vocabulary and concepts on a repeated basis [Dornyei and Ottó 1998]. Drop out rates in adult education are as high as 70% [Park and Choi 2009], despite an initially high starting motivation.

Despite the difficulty of finding time for learning, there are numerous moments in a day spent waiting, which are often wasted because these moments are not perceived to be useful for doing meaningful work. Recent work on *micro-learning* has explored ways to distribute language study into micro moments throughout a person's daily life, such as at moments when users return to their computers [Gassler et al. 2004]. In our work, we extend micro-learning and introduce *wait-learning*, a particular kind of micro-learning that targets waiting moments as timely triggers for learning. 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 [Fogg 2009]. Whereas micro-learning traditionally occurs during *live* time, or moments when a different task (the *primary task*) could conceivably continue, *wait* time is particularly promising because it takes place when the primary task is temporarily blocked, a situation in which users may be more receptive to a secondary task [Jin and Dabbish 2009]. Using wait time as a trigger for micro-learning, we hope to engage users in learning at times when they are more likely to be available.

Because waiting occurs within the context of existing activities, a core challenge of wait-learning is designing interactions in a way that minimizes interruption to the 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. 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 in our design process. We establish a design space for wait-learning, charting design dimensions that characterize both the waiting moment and the learning interaction.

To understand which factors make wait-learning more effective, we created WaitSuite, a multi-app infrastructure that supports wait-learning across a diversity of waiting moments. 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). We selected five kinds of waiting that span a range within the design space, then designed and implemented an app for each one to be integrated into WaitSuite:

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

The research questions we seek to answer are:

— How can wait-learning systems be designed to maximize learning engagement and minimize disruption to ongoing tasks?
— To what extent can people learn during waiting moments?

We first conducted a feasibility study on WaitChatter. 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. Second, we evaluated the design space of wait-learning in a two-week, in-situ study of WaitSuite. 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. Taken together, our 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.

This paper makes the following contributions:

— A design space for wait-learning and a suite of five different wait-learning apps (WaitSuite) that explores the design space.
— An in-situ evaluation of WaitSuite.
— Design implications and a theoretical framework for wait-learning.

2. RELATED WORK

Our work on wait-learning draws on existing research related to micro-learning. It is also informed by a large body of work on attention and interruption management, motivations behind multi-tasking, and the waiting experience.

2.1. Micro-Learning

A rich thread of research on *micro-learning* [Gassler et al. 2004] aims to distribute learning into small units throughout a person's day-to-day life. Micro-learning is motivated by strong evidence that retention of new content is enhanced when that content is presented in a spaced [Dempster 1987] and repeated [Webb 2007] manner. Given that humans exhibit a negatively exponential forgetting curve [Ebbinghaus 1913], content should be presented in increasingly spaced intervals, so that it is encountered just before it is likely to be forgotten. Several approaches, such as the Pimsleur method [Pimsleur 1967] and the Leitner algorithm [Godwin-Jones 2010], automate the scheduling of flashcards based on the history of prior exposures.

Existing systems for micro-learning have clustered around methods for teaching foreign language vocabulary that is relevant to the learner's context. Mobile applications such as MicroMandarin [Edge et al. 2011] and Vocabulary Wallpaper [Dearman and Truong 2012] present location-related vocabulary so that users can learn words in context while on the go. Other mobile applications also teach vocabulary that is related to nearby objects [Beaudin et al. 2007]. These systems for contextual learning have tended to focus more on *what* and *where* the user is learning rather than *when* to present these learning opportunities. Moreover, because these systems typically displayed exercises passively, users still needed to make a self-motivated, conscious decision to learn, a level of activation energy that is often too high. For example, in MicroMandarin, 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 [Edge et al. 2011]. Given that learners may have a high initial motivation to learn, but lack the executive motivation to practice on a repeated basis [Dornyei and Ottó 1998], the timing of learning opportunities may be critical to a user's ultimate engagement with learning.

To address the issue of motivation, several systems have embedded learning into daily routines to help users form a regular habit. For example, Lernschoner [Gassler et al. 2004] 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 [Gassler et al. 2004]. ALOE translates words within web articles a user is reading and displays them in a foreign language [Trusty and Truong 2011]. FeedLearn augments Facebook newsfeeds with inserted flashcard exercises, so that users can learn while casually browsing social media [Kovacs 2015]. However, in these micro-learning systems, engagement with learning was sometimes dampened because learning delayed higher-priority tasks. For example, some users disabled in-place translation of webpages because they felt it decreased the speed at which they could perform tasks on the Web [Trusty and Truong 2011]. 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 primary task could conceivably continue, wait-learning instead encourages users to learn during times when they would otherwise be waiting for a primary task to resume.

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. Peripheral Interaction. Peripheral interaction aims to enable everyday interactions that can meaningfully and effortlessly blend into daily routines, by supporting shifts between the periphery and the center of attention [Bakker et al. 2015; Edge and Blackwell 2016]. 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 [Pohl and Murray-Smith 2013], or *implicit human computer interaction*, by automatically sensing a user's situational context and adapting to it [Schmidt 2000].

2.2.2. 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 [Kahneman 1973]. 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 [Wickens 1984].

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 [Broadbent 1958], 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 [Broadbent 1958], while others show that the meaning of the input is also a factor [Deutsch and Deutsch 1963; Treisman 1960]. For example, overhearing one's own name during a cocktail party can turn one's attention toward input that was initially unintended [Conway et al. 2001].

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 [Bailey and Iqbal 2008; Adamczyk and Bailey 2004]. Many studies have shown that mental workload decreases at the boundaries between subtasks [Miyata and Norman 1986], and decreases more between larger chunks of a task [Bailey and Iqbal 2008]. 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 [Bailey and Iqbal 2008], 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 [Salvucci and Taatgen 2008].

2.2.3. 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 [Monsell 2003; Wylie and Allport 2000]. 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.

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 [Borst et al. 2015]. 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 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 [Anderson 2005; Borst et al. 2010]. 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 [Lang

2006; Wang and Tchernev 2012]. 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 [Jin and Dabbish 2009]. 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 [Mark et al. 2015; Neuberg et al. 2011]. 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 [Wickens et al. 2015].

2.3.2. The Waiting Experience. There is strong evidence that waiting has a measurable effect on user satisfaction, mood, and level of frustration [Dellaert and Kahn 1999]. The longer the wait time relative to the expected waiting time, the lower the customer satisfaction [Dellaert and Kahn 1999; Maister 1984]. 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 [Lin and Raghavendra 1996], 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 [Maister 1984], a large number of approaches have been used to help improve the waiting experience. These range from filling wait time with music [Hul et al. 1997], content [Alt et al. 2012; Katz et al. 1991; Niida et al. 2011], and wait time estimates [Katz et al. 1991], to redesigning progress bars that alter the perception of time [Harrison et al. 2007].

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 [Maister 1984; Nah 2004], 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 [Jin and Dabbish 2009]. 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. 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 the interruption cost is higher if the ongoing task utilizes substantial attentional resources or demands memory of a problem state at the moment the second task is introduced. Furthermore, 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. In the following sections, we enumerate the design space of wait-learning, the systems we built to explore the design space of wait-learning, and our evaluations of those systems.

3. 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.

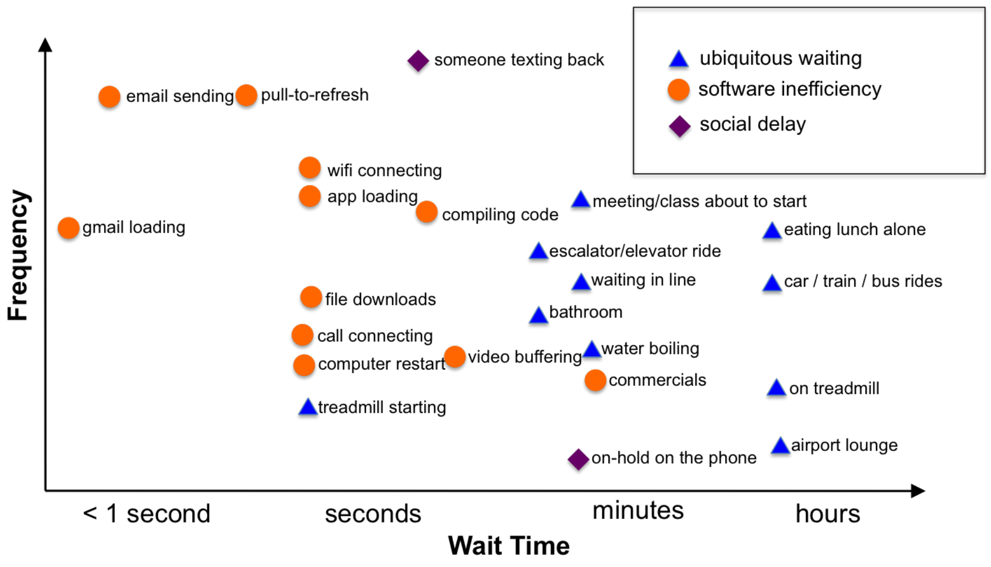


Fig. 1: A diverse set of brainstormed waiting moments, displayed by wait time and frequency. Some waiting moments occur in ubiquitous contexts, while others are due to software inefficiencies or delays in social communication.

3.1. Waiting Moment

To chart the space of waiting options, we first brainstormed a list of 20-30 waiting opportunities (Figure 1). 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 waiting moments and explain why we focused on particular subsets within each dimension.

3.1.1. Wait Time. Wait times can range from a few seconds to several 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 ubiquitous 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 ubiquitous 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 during ubiquitous activities. We do not explore hour-long waits, as these tend to be waits that are activities in and of themselves (e.g. eating lunch).

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.

3.1.2. 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 [Dempster 1987]. 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.

3.1.3. Competing Demands. As described in related work (Section 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.

3.1.4. 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 (Section 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 [Monsell 2003; Wylie and Allport 2000; Altmann and Trafton 2004; Borst et al. 2015]. 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 [Csikszentmihalyi 1990] 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 [Nardi et al. 2000]. 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 [Jin and Dabbish 2009], 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 [Böhmer et al. 2014], so that people can continue to view and monitor the main task even while completing the learning task.

3.2. Wait-Learning Interaction

3.2.1. 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 instructional research, instructional elements with few interdependent parts are easier to learn in isolation [Sweller et al. 1998]. For example, the study of vocabulary, special terminology, and chemical symbols impose a lower cognitive load in comparison to learning algebra, which has more interdependent parts [Sweller et al. 1998]. Furthermore, as described in related work (Section 2), interruption costs are lower when the secondary task does not require maintaining an intermediate problem state [Borst et al. 2015]. Therefore, while more complex concepts could in theory be learned during longer wait times, in this paper we focus on the task of vocabulary learning, which has low problem state because each word can reasonably be learned in isolation [Sweller et al. 1998].

3.2.2. Manner of Appearance. Peripheral interactions should be designed so that they are easy-to-initiate and easy-to-discard [Bakker et al. 2015]. Thus, the learning task should appear subtly at the periphery of the user's attention, and should also require little effort to ignore.

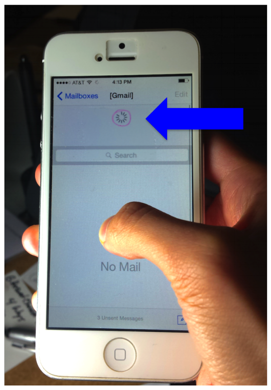
A learning exercise could be displayed as a hard notification, soft notification, or static notification. Prior work defined soft notifications as ones that appear gradually [Wilson 2006]. In this work we distinguish these categories more concretely as follows: hard notifications involve both animated 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 [Bailey et al. 2001], and can result in a severe loss of productivity [Bailey and Iqbal 2008]. On the other hand, a completely static notification may result in the user becoming habituated to the display over time [Jordan and Tipper 1998], leading to low engagement and potential abandonment.

After feedback from pilot studies, 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.

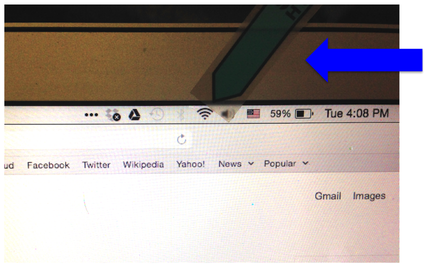
To maximize the ease of both initiating the learning task and resuming the primary task, the learning component should be positioned near the most likely locus of attention, such as below the last line of chat, or in dead space near a loading icon. If requiring manual interaction, it should also be physically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

3.3. 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



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.



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.

Fig. 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.

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.

3.3.1. 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 two rounds of data collection. Ten users (round 1) and eight users (round 2) 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). The participants were students and faculty at a university.

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, such as over the progress icon during a pull-to-refresh, as shown in Figure 2. Users took a timestamped screenshot to tally their waiting moment in a single action without leaving their current task.

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.

4. 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, Pul-

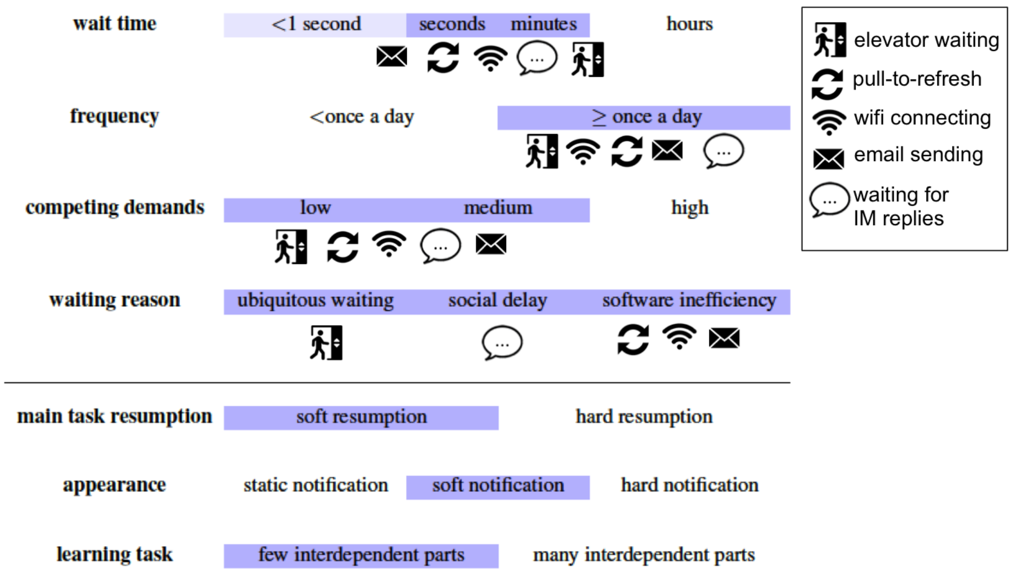
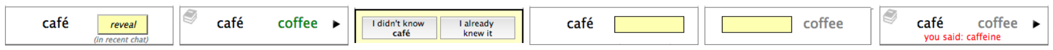


Fig. 3: The design space of wait-learning applications. Our design choices are highlighted in blue. The five waiting scenarios we selected vary in wait time, frequency, competing demands, and waiting reason. For the dimensions listed below the line, we required all five scenarios to meet the requirement 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.

lLearner, WifiLearner, EmailLearner, and WaitChatter. Figure 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.

4.1. 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 ubiquitous 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 directly a result of ubiquitous waiting, pull-to-refresh might also occur in ubiquitous settings while mobile. 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.



a) In Study Mode, user clicks Reveal to show the translation.
b) After clicking Reveal, the translation is shown.
c) User indicates whether or not 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.

Fig. 4: Components of WaitSuite vocabulary exercises.

4.2. 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 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.

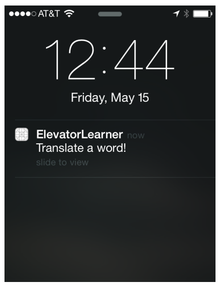
4.3. Vocabulary Exercises

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 4a) and the user presses Reveal to see the L1 (native language) translation target (Figure 4b). After revealing the new word, the user indicates whether they already knew that word (Figure 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 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 4e). On desktop interfaces, users can submit a blank response if they don't remember the word.

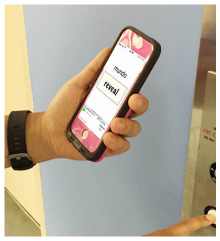
After the user completes an *initial* exercise, a *follow-up* one can be fetched by clicking an arrow or hitting the Enter key (Figure 4f). Fetching follow-up exercises can lead to chains of consecutive exercises.

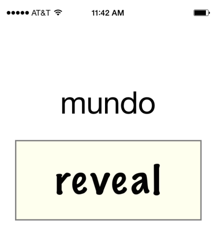
4.4. ElevatorLearner

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 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 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 Section4.3 above, the user sees the vocabulary prompt in L2 at the easy level, and L1 at the difficult level (Figure 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 5d). After self-grading, the next flashcard is shown, which the user can either engage with or ignore by leaving the app.

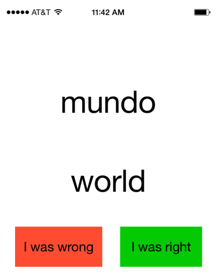


a) When the user walks into an elevator area, a bluetooth iBeacon triggers a mobile notification.


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.

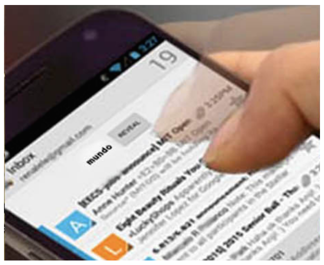


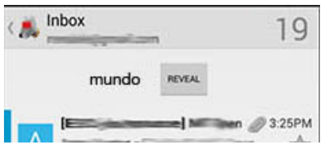
d) After the user pushes Reveal, the translation is shown. The user self grades by pushing Right or Wrong.

Fig. 5: ElevatorLearner user interface.

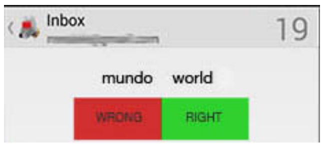
4.5. PullLearner

PullLearner was built on top of K9Mail, 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 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 [Ren 2015]. Similar to ElevatorLearner, the user presses Reveal to see the translation (Figure 6b), then presses right or wrong to self grade (Figure 6c), after which the learning panel retracts. The user can optionally fetch a follow-up exercise by pulling again.


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.

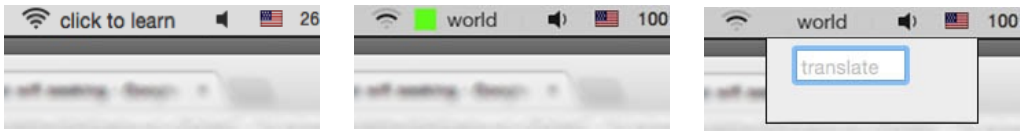


c) After the user pushes Reveal, the translation is shown. The user self grades by pushing Right or Wrong.

Fig. 6: PullLearner user interface.

4.6. 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 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 7b). If the user clicks to start an exercise, a learning panel appears directly below, where the user can type the translation (Figure 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.

Fig. 7: WifiLearner user interface.

4.7. EmailLearner

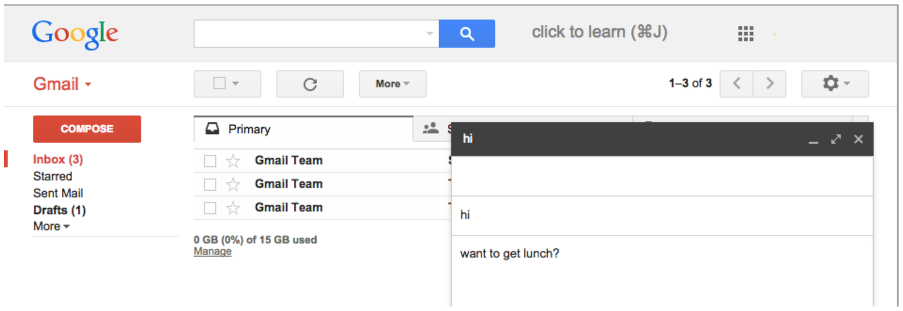
EmailLearner is a Gmail Chrome extension that appears when the user hits Send on an email. In many email clients, a status label indicates the state of the email being sent. In Gmail, it appears in the form of "Sending..." followed by "Your message has been sent," which users often watch to make sure 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 8a). Once the user sends an email, the panel displays the learning exercise (Figure 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.

4.8. WaitChatter

We developed WaitChatter as a Chrome extension of Google Chat that runs in the Web browser when a Gmail page is in view.[1] 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, he can fetch a *follow-up* one by clicking the right arrow (Figure 4f) or hitting the Enter key on the keyboard. This functionality allows the user to continuously learn more words during longer wait times.

Because IM conversations provide text that can be used for in-context learning, WaitChatter not only draws words from a built-in word list, but also extracts words on-the-fly from the IM conversation. To indicate that a word is contextual to the conversation, WaitChatter displays "in recent chat" directly below the exercise (Figure 9a). To prevent user concern over whether the other person can view WaitChatter content, we keep learning exercises within the learning panel, and highlight

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



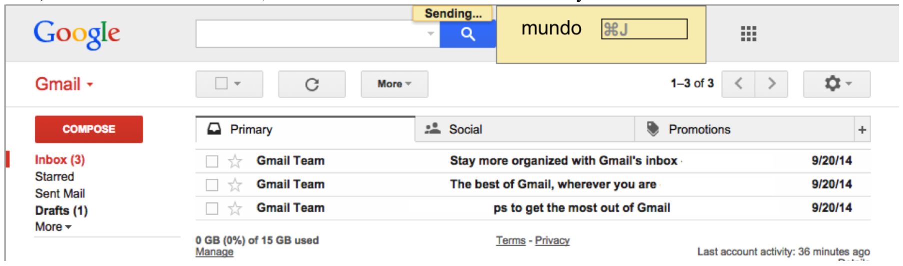
☆	Gmail Team
☆	Gmail Team
☆	Gmail Team

hi

hi

want to get lunch?

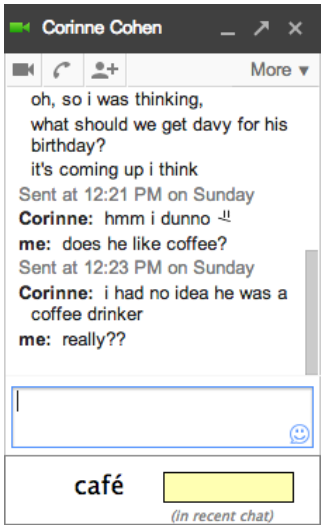
a) The user writes an email, then hits the send button or other keyboard shortcuts to send the email.



☆	Gmail Team	Stay more organized with Gmail's inbox	9/20/14
☆	Gmail Team	The best of Gmail, wherever you are	9/20/14
☆	Gmail Team	ps to get the most out of Gmail	9/20/14

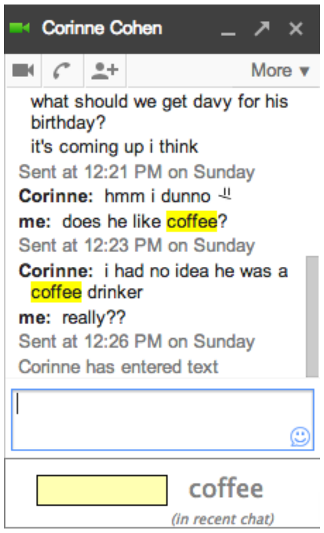
b) As soon as the email is triggered to be sent, EmailLearner displays an exercise next to the "Sending..." label.

Fig. 8: EmailLearner interface.



oh, so i was thinking,
what should we get davy for his birthday?
it's coming up i think
Sent at 12:21 PM on Sunday
Corinne: hmm i dunno
me: does he like coffee?
Sent at 12:23 PM on Sunday
Corinne: i had no idea he was a coffee drinker
me: really??

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



what should we get davy for his birthday?
it's coming up i think
Sent at 12:21 PM on Sunday
Corinne: hmm i dunno
me: does he like coffee?
Sent at 12:23 PM on Sunday
Corinne: i had no idea he was a coffee drinker
me: really??
Sent at 12:26 PM on Sunday
Corinne has entered text

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

Fig. 9: WaitChatter contextual interface.

a keyword inside the chat history if it is selected and presented for learning (Figure 9b). Details

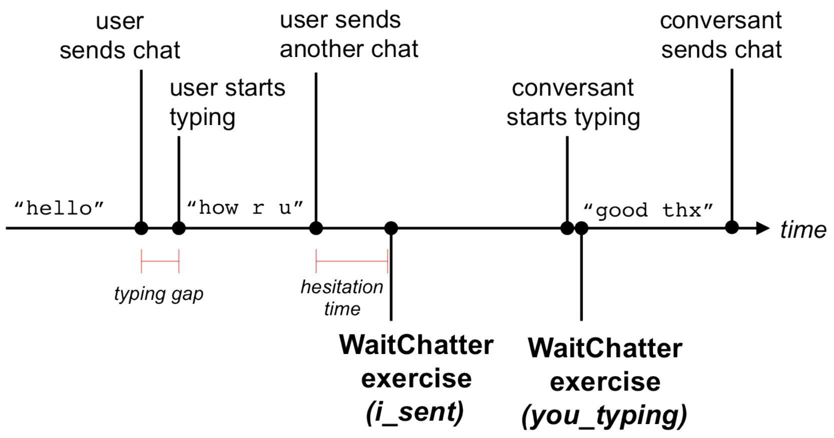


Fig. 10: Detection of waiting opportunities in WaitChatter.

about contextual word extraction and automatic translation can be found in a previous report on WaitChatter [Cai et al. 2015].

4.8.1. 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 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 [Isaacs et al. 2002; Ling and Baron 2007], 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 [Avrahami et al. 2008], 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.

5. 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.

5.1. Vocabulary Scheduling Algorithm

We use the Leitner schedule [Godwin-Jones 2010] for determining the order of learning exercises. The Leitner schedule is based on the principle of *spaced repetition* [Baddeley 1997]. Given that humans exhibit a negatively exponential forgetting curve [Ebbinghaus 1913], 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.

5.2. 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[2] 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.

5.3. 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.

6. STUDY 1: WAITCHATTER

To evaluate the extent to which people can learn while waiting, we first implemented WaitChatter. We ran a two-week field study in which participants used WaitChatter in Google Chat on their personal computers, during their normal instant messaging activities. We reported on WaitChatter in previous work [Cai et al. 2015], but here we summarize the study and our main findings.

The questions our study sought to answer were:

(1) Learning: To what extent can users learn vocabulary using WaitChatter?
(2) Timing: What is the best time to present learning exercises within a waiting period?

6.1. 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.[3] The words were translated to Spanish and French using Google Translate. Two native speakers manually reviewed the word list for inaccurate translations, and removed highly ambiguous words. The final word lists consisted of 446 words in each language. In addition to these word lists, contextual words were also automatically extracted from the conversation and translated on-the-fly, as described in Section 4.8.

6.2. Procedure

Each participant used WaitChatter for two weeks, after which they completed a post-study questionnaire, interview, and vocabulary quiz. Participants were asked to interact with WaitChatter exercises

[2]https://www.firebase.com/
[3]http://www.natcorp.ox.ac.uk/

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, then from L2 to L1.

6.2.1. 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 *detected_wait* version uses the *i_sent* and *you_typing* waiting opportunities as described above. The *random* version displays prompts at random whenever WaitChatter determines that a user is actively instant messaging.

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 interaction with 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 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.

6.3. 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 (mean=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.

6.4. 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.

6.4.1. 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. In post-study quizzes, users translated 57.1 words (66%) correctly to L2 and 80.2 words (92%) correctly to L1. 15% of wrong answers appeared to be spelling errors or near-misses. Thus, in two weeks of casual usage, 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.

6.4.2. 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 more likely to engage when the conversation was casual and when chatting was their main task, so that there was frequent back-and-forth waiting. High-usage participants said that they frequently used the "fetch more" feature (Figure 4f) to do a long sequence of exercises if the conversation was particularly sporadic.

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 11). 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 3.2.1.

6.4.3. 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 potential for wait-learning to entertain, stimulate, or amuse is a direction worth exploring, particularly given the well-known negative experience associated with waiting [Maister 1984].

6.4.4. Overcoming Limited Time. The most commonly reported benefit of WaitChatter was that it felt less time-consuming compared to existing learning channels because it offloaded the burden of setting aside time for learning. As one user stated, "The key thing is that I didn't feel like I was taking extra time out of my day to dedicate to learning vocabulary words. It was just sort of time that would be wasted otherwise." Users contrasted WaitChatter to language courses and other mobile apps, which they felt required a conscious effort to schedule time for learning: "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 in the past indicated that they eventually gave up, citing lack of time as a major factor.

6.4.5. 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*.

We found that engagement rate was highest for *i_sent* (49.1%), followed by *random* (44.5%) and *you_typing* (41.2%). A logistic mixed effects analysis with Bonferroni correction found that users were significantly more likely to respond when exercises appeared just after the user sent a chat mes-

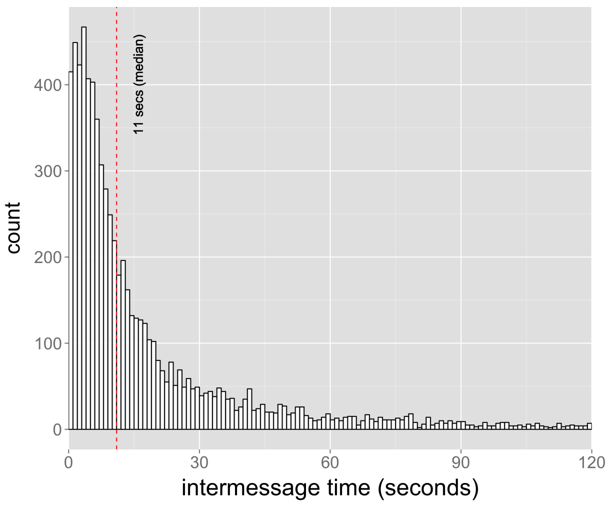


Fig. 11: Histogram of intermessage time: the time between the user sending a message and receiving a message from the conversant. Bin size is 1 second.

sage (*i_sent*) relative to the *you_typing* condition (p<0.01), with an odds ratio of 1.4. Furthermore, a repeated measures ANOVA with bonferroni correction found that users were significantly faster to respond in the *i_sent* condition (mean=3.63 sec, σ=0.64) than in the *random* condition (mean=4 sec, σ=0.79) (F(2,38)=4.00, p<0.05, $\eta_p^2 = 0.18$). Figure 12 shows response times by condition. The response time benefit of *i_sent* over *you_typing* was marginally significant (p=0.05) after Bonferroni correction.

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 [Miyata and Norman 1986]. 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 [Bailey and Iqbal 2008]. 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. Thus, the expected wait time becomes so short that it no longer justifies the switch cost.

At the beginning of a waiting period, users are also more likely to be in a state where peripheral interaction is better supported. In the *i_sent* condition, the exercise appears soon after the user sends a message, so the likelihood that the user is still visually and cognitively attending to the chatbox is high. Conversely, *you_typing* exercises appear further into the waiting period, when users may already be focused elsewhere. 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 [Bakker et al. 2015].

6.4.6. 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 activ-

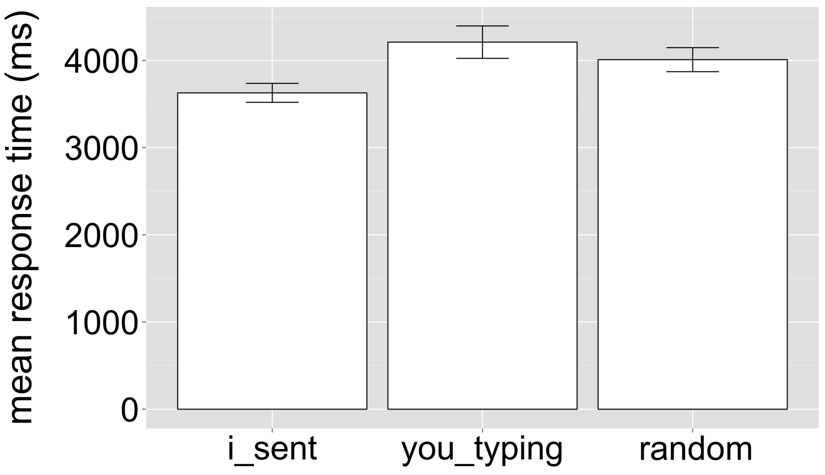


Fig. 12: Mean response times for *i_sent*, *you_typing*, and *random*. Error bars show SE of the mean.

ities" (mean=5.45, σ=1.05) and that they "enjoyed using WaitChatter today" (mean=5.61, σ=1.02). A Wilcoxon signed-rank test found no significant difference between user ratings on *detected_wait* versus *random* condition days, for either question (p=0.71 and p=0.57).

We posit that the use of soft notifications (static learning panel, dynamic text) was key to minimizing intrusiveness during poorly-timed exercises. 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."

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 [Böhmer et al. 2014]. 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. Since the version of WaitChatter tested in the study lacked this feature, we added it for Study 2.

In conclusion, results from this study show that bite-sized learning is feasible during wait time, and that a good time to present learning opportunities is at the start of the waiting period.

7. STUDY 2: WAITSUITE

In Study 2, we expand our analysis to the suite of five wait-learning apps and evaluate them in the context of the design dimensions we have described, i.e., wait time, frequency, competing demands. To assess WaitSuite in a real-world setting, we ran a field study in which 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 user engagement as a key metric in our evaluation. To keep exercises consistent across apps, all five apps used second language vocabulary exercises.

Study 2 explored 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?

7.1. Procedure

To answer these questions, we deployed WaitSuite on the participants' own devices for a two-week study. 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 Study 1. As before, the words were translated from high frequency English nouns taken from the British National Corpus. There were 446 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. 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 [Iqbal and Bailey 2005].

7.2. User Interface Modifications

Based on timing results from Study 1, we modified WaitChatter to automatically trigger exercises only after the user sends a chat message, the *i_sent* condition from Study 1. To be consistent with the other four apps, we also turned off WaitChatter's contextual feature which detects words within conversations.

Because users in Study 1 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).

7.3. 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. Participants were given a $40 gift card for their time.

The 25 participants (11 female) who completed the study were ages 19 to 47 (mean=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.

7.4. 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

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 I: 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.

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 I and II 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.
— *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 finishing 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 it appears.

7.5. Results: Engagement with System-Triggered Exercises (RQ1)

7.5.1. 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%, σ=12.1%), followed by ElevatorLearner (46.7%, σ=21.3%), EmailLearner (39.8%, σ=31%), WifiLearner (17.5%, σ=14.2%), and WaitChatter (13.4%, σ=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 ex-

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

Table II: For each app, the engagement rate (% engaged) was related to the ease of accessing the exercise (estimated as response time), the waiting duration (wait time), and the likelihood of competing demands. Engagement was highest when response time was lower then 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.

ercises 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 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. Figure 13 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.

7.5.2. ElevatorLearner. Relative to other apps, ElevatorLearner had a high engagement rate (46.7%, σ=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.

7.5.3. PullLearner. PullLearner also exhibited a high engagement rate (62.5%, σ=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.

7.5.4. 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

	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

Fig. 13: 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.

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.

7.5.5. 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 14, Pearson's

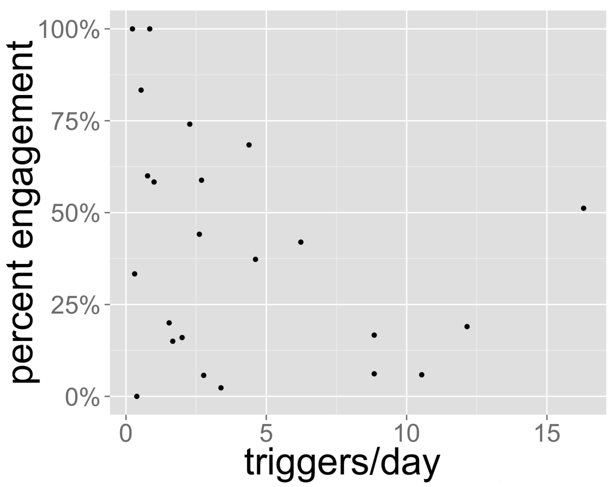


Fig. 14: On EmailLearner, there was a mild negative correlation between frequency of exposure to system-triggered exercises and likelihood of engaging with an exercise.

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%, σ=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 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.

7.5.6. 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.

7.5.7. 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 (mean=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).

7.6. 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.

7.7. 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.

7.7.1. Multi-App Usage. Analysis of our log data showed that a majority of users interleaved usage between apps, even within a single day (Figure 15). 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.

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 16). 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 17 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.

7.7.2. 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 unified." In contrast to previous single-app platforms they had used, users felt that Wait-

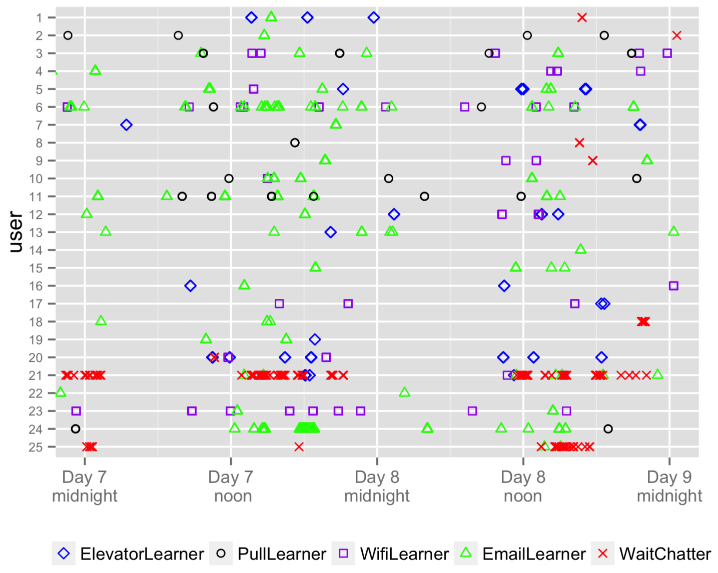


Fig. 15: A two-day snapshot of exercises submitted by each user. Most users (i.e. users 1-17) interleaved between different apps within a single day. We label these users *generalists*. The remaining users (i.e. users 18-25) could be considered *specialists*, submitting more than 75% of exercises in 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.

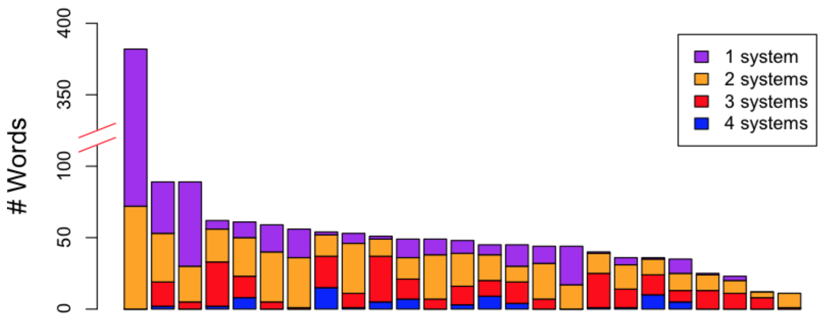


Fig. 16: For each user, the number of vocabulary words whose exercises appeared across 1, 2, 3, and 4 systems. Users are sorted from most (left) to least (right) number of exercises completed.

Suite 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.

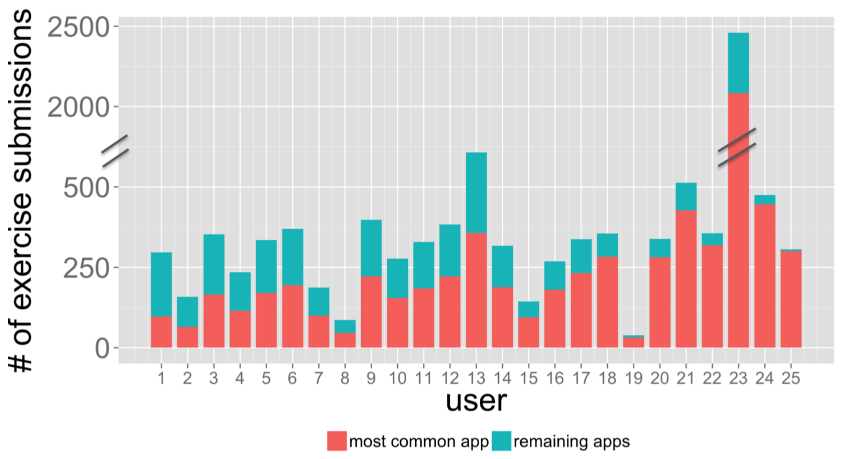


Fig. 17: For each user, the number of exercise submissions that were completed on the user's most common app, compared to the user's remaining apps. Users are ordered from the lowest (left) to highest (right) percentage of exercises completed on the most common app. Averaged across all users, 65% of exercises were completed on the most common app, and 35% were completed on the remaining apps.

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.

7.7.3. 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.

7.7.4. 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.”

7.7.5. Learning Across Apps. During the study, participants were exposed to 88 (σ=101) words on average, 61 (σ=70) of which they didn’t already know. After two weeks, participants translated 50 words (86%, σ=11%) correctly to L1 and 35 (60%, σ=18%) words to L2. User 23 (Figure 17) 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 [Edge et al. 2012; Pavlik and Anderson 2008], should be used in the future to improve long term retention.

In analyzing quiz results, we found that the average number of words retained in Study 2 (35) was not as high as that of Study 1 (57), even though in Study 2, participants used multiple apps. Whereas Study 1 recruited all regular users of Google Chat, Study 2 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 Study 1 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.

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.

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 18). 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 [Kahneman 1973; Monsell 2003], motivation [Lang 2006], and the waiting experience [Maister 1984] 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, in Study 2, we found

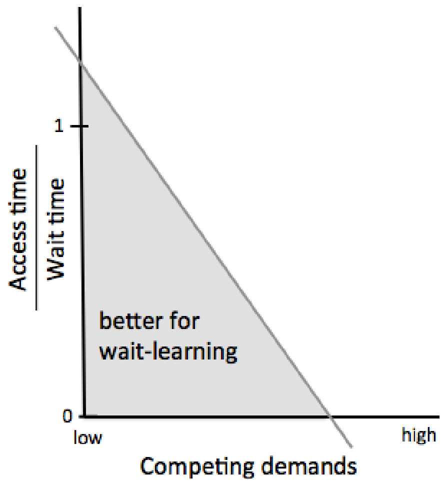


Fig. 18: Wait-learning is more engaging and less disruptive when the time taken to access an exercise is shorter than the wait time, and when competing mental demands are low. In situations where the access-to-wait ratio is high, competing demands ought to be even lower. This depiction assumes that the learning task is short and bite-sized.

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.

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.

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.

8.4. Customization

Although many participants used a combination of apps, several focused primarily on one app. Some waiting moments also coincided with very different levels of mental demand in different situations, e.g. instant messaging. Future implementations of wait-learning could consider a casual interaction design [Pohl and Murray-Smith 2013] that enables a spectrum of interactions, depending on how much the user is able to engage at the moment. Or, it could also 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 self-triggers exercises on Tuesdays at 2pm while walking to class. Given the day to day variations we observed during the study, these pattern suggestions should be made conservatively, to avoid unnecessarily reducing opportunities for learning.

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.

8.6. System-Triggers and Habit Formation

In Study 2, 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" (Wait-Chatter) 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 [Bentley and Tollmar 2013]. 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.

Secondly, user feedback in both studies support existing theories that well-timed triggers are key to sustaining a habit [Fogg 2009]. For example, 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.” In future systems, it would be useful to explore how system-triggers can better complement daily reminders, given variation of exposure to system-triggers.

8.7. Wait-learning and Well-being

A potential downside of wait-learning could be the inadvertent replacement of time spent reflecting or resting. Wait-learning is less appropriate for users who already make effective use of wait time, or who do not already engage in technology use while waiting. In our studies, however, most users had existing habits of engaging with technology use during wait time. Many described wait-learning as a timely replacement for their existing compulsive waiting habits, such as browsing social media, playing Candy Crush, or checking email repeatedly. Some also described WaitChatter as being playful and game-like, similar to other games they would play on their phones while waiting. Hence, for them, wait-learning did not actually fill unused time, but rather replaced existing digital activities with more productive ones.

Although we used vocabulary learning as one example of productivity, wait-learning could potentially be extended to other domains and applications, such as encouraging healthier habits. For example, wait time could be an opportune moment to provide timely reminders to stretch, plan, meditate, relax, or reflect. We envision the design dimensions and theoretical framework established in this paper to form the basis for future work to help people wait-learn healthier habits in general.

8.8. Limitations

We arrived at our conclusions not through a controlled study, but rather through an inductive analysis and iterative design process. 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. Thus, each participant installed only a subset of the apps. For example, no user installed both ElevatorLearner and PullLearner because one was implemented on iPhone to maximize compatibility with iBeacons, and the other on Android to make use of an open source email client. The statistical analyses used in our evaluation are intended to provide some grounding to our insights, but should not be interpreted in isolation.

Our results are also limited to the demographics of participants in our studies, which were primarily young adults in an academic setting. In the future, our design space could be expanded to include waiting scenarios commonly encountered by other demographics, such as adult professionals juggling work and caregiving, waiting for their children at soccer practice or for their older parents at medical appointments. Wait-learning should be evaluated within these broader contexts of use. Lastly, the users in our studies were primarily beginning and intermediate language learners. This meant that, while using WaitSuite, users typically marked only a minority of words as already known. 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. CONCLUSION

In this paper, we described a design space for wait-learning, and created five wait-learning applications targeting a variety of waiting scenarios, each with unique interaction constraints. In two evaluations, we found that wait-learning is effective for bite-sized learning, and that exercises timed at the start of waiting periods receive higher engagement and faster response time than alternative timing conditions. We also 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.

ACKNOWLEDGMENTS

We are very grateful to the entire User Interface Design and Haystack groups at MIT CSAIL for their help pilot testing WaitSuite, and for providing feedback on drafts. In particular, we would like to thank Elena Glassman, Katherine Fang, Max Goldman, Shamsi Iqbal, William Li, and Ramesh Sridharan for their help on this project, and Tom Buehler for help on the video. This research was funded by Quanta Computer and the National Science Foundation award SOCS-1111124.

REFERENCES

Piotr D Adamczyk and Brian P Bailey. 2004. If not now, when?: the effects of interruption at different moments within task execution. In *CHI '04*. ACM, 271–278.

Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, and Richard Atterer. 2012. Bridging waiting times on web pages. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. ACM, 305–308.

Erik M Altmann and J Gregory Trafton. 2004. *Task interruption: Resumption lag and the role of cues*. Technical Report. DTIC Document.

John R Anderson. 2005. Human symbol manipulation within an integrated cognitive architecture. *Cognitive science* 29, 3 (2005), 313–341.

Daniel Avrahami, Susan R Fussell, and Scott E Hudson. 2008. IM waiting: timing and responsiveness in semi-synchronous communication. In *CSCW '08*. ACM, 285–294.

Alan D Baddeley. 1997. *Human memory: Theory and practice*. Psychology Press.

Brian P Bailey and Shamsi T Iqbal. 2008. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *TOCHI* 14, 4 (2008), 21.

Brian P Bailey, Joseph A Konstan, and John V Carlis. 2001. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Proceedings of INTERACT*, Vol. 1. 593–601.

Saskia Bakker, Elise van den Hoven, and Berry Eggen. 2015. Peripheral interaction: characteristics and considerations. *Personal and Ubiquitous Computing* 19, 1 (2015), 239–254.

Jennifer S Beaudin, Stephen S Intille, Emmanuel Munguia Tapia, Randy Rockinson, and Margaret E Morris. 2007. Context-sensitive microlearning of foreign language vocabulary on a mobile device. In *Ambient Intelligence*. Springer, 55–72.

Frank Bentley and Konrad Tollmar. 2013. The power of mobile notifications to increase wellbeing logging behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1095–1098.

Matthias Böhmer, Christian Lander, Sven Gehring, Duncan P Brumby, and Antonio Krüger. 2014. Interrupted by a phone call: exploring designs for lowering the impact of call notifications for smartphone users. In *CHI '14*. ACM, 3045–3054.

Jelmer P Borst, Niels A Taatgen, and Hedderik van Rijn. 2010. The problem state: a cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, memory, and cognition* 36, 2 (2010), 363.

Jelmer P Borst, Niels A Taatgen, and Hedderik van Rijn. 2015. 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*. ACM, 2971–2980.

DE Broadbent. 1958. Perception mid communication. (1958).

Carrie J Cai, Philip J Guo, James Glass, and Robert C Miller. 2015. Wait-learning: leveraging wait time for education. In *CHI'14*. ACM.

Andrew RA Conway, Nelson Cowan, and Michael F Bunting. 2001. The cocktail party phenomenon revisited: The importance of working memory capacity. *Psychonomic bulletin & review* 8, 2 (2001), 331–335.

M Csikszentmihalyi. 1990. Flow: The psychology of optimal experience," New Yori: Harper and Row. (1990).

David Dearman and Khai Truong. 2012. Evaluating the implicit acquisition of second language vocabulary using a live wallpaper. In *CHI '12*. ACM, 1391–1400.

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

Frank N Dempster. 1987. Effects of variable encoding and spaced presentations on vocabulary learning. *Journal of Educational Psychology* 79, 2 (1987), 162.

J Anthony Deutsch and Diana Deutsch. 1963. Attention: Some theoretical considerations. *Psychological review* 70, 1 (1963), 80.

Zoltan Dornyei and István Ottó. 1998. Motivation in action: A process model of L2 motivation. *Working Papers in Applied Linguistics* (1998), 43–69.

Hermann Ebbinghaus. 1913. *Memory: A contribution to experimental psychology*. Number 3. Teachers college, Columbia university.

Darren Edge and Alan F Blackwell. 2016. Peripheral Tangible Interaction. In *Peripheral Interaction*. Springer, 65–93.

Darren Edge, Stephen Fitchett, Michael Whitney, and James Landay. 2012. MemReflex: adaptive flashcards for mobile microlearning. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. ACM, 431–440.

Darren Edge, Elly Searle, Kevin Chiu, Jing Zhao, and James A Landay. 2011. MicroMandarin: mobile language learning in context. In *CHI '11*. ACM, 3169–3178.

Brian J Fogg. 2009. A behavior model for persuasive design. In *Proceedings of the 4th international Conference on Persuasive Technology*. ACM, 40.

Gerhard Gassler, Theo Hug, and Christian Glahn. 2004. Integrated Micro Learning–An outline of the basic method and first results. *Interactive Computer Aided Learning* 4 (2004).

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

Chris Harrison, Brian Amento, Stacey Kuznetsov, and Robert Bell. 2007. Rethinking the progress bar. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 115–118.

Michael K Hul, Laurette Dube, and Jean-Charles Chebat. 1997. The impact of music on consumers' reactions to waiting for services. *Journal of Retailing* 73, 1 (1997), 87–104.

Shamsi T Iqbal and Brian P Bailey. 2005. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI'05 Extended Abstracts*. ACM, 1489–1492.

Ellen Isaacs, Alan Walendowski, Steve Whittaker, Diane J Schiano, and Candace Kamm. 2002. The character, functions, and styles of instant messaging in the workplace. In *CSCW '02*. ACM, 11–20.

Jing Jin and Laura A Dabbish. 2009. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1799–1808.

Heather Jordan and Steven P Tipper. 1998. Object-based inhibition of return in static displays. *Psychonomic Bulletin & Review* 5, 3 (1998), 504–509.

Daniel Kahneman. 1973. *Attention and effort*. Citeseer.

Karen L Katz, Blaire M Larson, and Richard C Larson. 1991. Prescription for the waiting-in-line blues: Entertain, enlighten, and engage. *MIT Sloan Management Review* 32, 2 (1991), 44.

Geza Kovacs. 2015. FeedLearn: Using Facebook Feeds for Microlearning. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1461–1466.

Annie Lang. 2006. Using the limited capacity model of motivated mediated message processing to design effective cancer communication messages. *Journal of Communication* 56, s1 (2006), S57–S80.

Hwa-Chun Lin and CS Raghavendra. 1996. An approximate analysis of the join the shortest queue (JSQ) policy. *Parallel and Distributed Systems, IEEE Transactions on* 7, 3 (1996), 301–307.

Rich Ling and Naomi S Baron. 2007. Text messaging and IM linguistic comparison of American college data. *Journal of Language and Social Psychology* 26, 3 (2007), 291–298.

David H Maister. 1984. *The psychology of waiting lines*. Harvard Business School.

Gloria Mark, Shamsi Iqbal, Mary Czerwinski, and Paul Johns. 2015. 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*. ACM, 903–916.

Yoshiro Miyata and Donald A Norman. 1986. Psychological issues in support of multiple activities. *User centered system design: New perspectives on human-computer interaction* (1986), 265–284.

Stephen Monsell. 2003. Task switching. *Trends in cognitive sciences* 7, 3 (2003), 134–140.

Fiona Fui-Hoon Nah. 2004. A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology* 23, 3 (2004), 153–163.

Bonnie A Nardi, Steve Whittaker, and Erin Bradner. 2000. Interaction and outeraction: instant messaging in action. In *CSCW '00*. ACM, 79–88.

Steven L Neuberg, Douglas T Kenrick, and Mark Schaller. 2011. Human threat management systems: Self-protection and disease avoidance. *Neuroscience & Biobehavioral Reviews* 35, 4 (2011), 1042–1051.

Sumaru Niida, Satoshi Uemura, Hajime Nakamura, and Etsuko Harada. 2011. 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*. ACM, 177–180.

Ji-Hye Park and Hee Jun Choi. 2009. Factors influencing adult learners' decision to drop out or persist in online learning. *Journal of Educational Technology & Society* 12, 4 (2009), 207–217.

Philip I Pavlik and John R Anderson. 2008. Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied* 14, 2 (2008), 101.

Paul Pimsleur. 1967. A memory schedule. *Modern Language Journal* (1967), 73–75.

Henning Pohl and Roderick Murray-Smith. 2013. Focused and casual interactions: allowing users to vary their level of engagement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2223–2232.

Anji Ren. 2015. Pull-To-Refresh and Learn: Leveraging Mobile Email Load Time for Education. In *CHI'15 Extended Abstracts*. ACM.

Dario D Salvucci and Niels A Taatgen. 2008. Threaded cognition: an integrated theory of concurrent multitasking. *Psychological review* 115, 1 (2008), 101.

Albrecht Schmidt. 2000. Implicit human computer interaction through context. *Personal technologies* 4, 2-3 (2000), 191–199.

John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paas. 1998. Cognitive architecture and instructional design. *Educational psychology review* 10, 3 (1998), 251–296.

Anne M Treisman. 1960. Contextual cues in selective listening. *Quarterly Journal of Experimental Psychology* 12, 4 (1960), 242–248.

Andrew Trusty and Khai N Truong. 2011. Augmenting the web for second language vocabulary learning. In *CHI '11*. ACM, 3179–3188.

Zheng Wang and John M Tchernev. 2012. The myth of media multitasking: Reciprocal dynamics of media multitasking, personal needs, and gratifications. *Journal of Communication* 62, 3 (2012), 493–513.

Stuart Webb. 2007. The effects of repetition on vocabulary knowledge. *Applied Linguistics* 28, 1 (2007), 46–65.

Christopher Wickens. 1984. D.(1984). Processing resources in attention. *Varieties of attention* (1984), 63–102.

Christopher D Wickens, Robert S Gutzwiller, and Amy Santamaria. 2015. Discrete task switching in overload: A meta-analyses and a model. *International Journal of Human-Computer Studies* 79 (2015), 79–84.

Thomas Blake Wilson. 2006. *Gradual awareness notification for the desktop environment*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Glenn Wylie and Alan Allport. 2000. Task switching and the measurement of switch costs. *Psychological research* 63, 3-4 (2000), 212–233.