Tabs.do: Task-Centric Browser Tab Management

Joseph Chee Chang
josephcc@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Yongsung Kim
yongsung@cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Victor Miller
vamiller@tepper.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Michael Xieyang Liu
xieyangl@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Brad A. Myers
bam@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Aniket Kittur
nkittur@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

ABSTRACT

Despite the increasing complexity and scale of people's online activities, browser interfaces have stayed largely the same since tabs were introduced in major browsers nearly 20 years ago. The gap between simple tab-based browser interfaces and the complexity of users' tasks can lead to serious adverse effects – commonly referred to as "tab overload." This paper introduces a Chrome extension called Tabs.do, which explores bringing a task-centric approach to the browser, helping users to group their tabs into tasks and then organize, prioritize, and switch between those tasks fluidly. To lower the cost of importing, Tabs.do uses machine learning to make intelligent suggestions for grouping users' open tabs into task bundles by exploiting behavioral and semantic features. We conducted a field deployment study where participants used Tabs.do with their real-life tasks in the wild, and showed that Tabs.do can decrease tab clutter, enabled users to create rich task structures with lightweight interactions, and allowed participants to context-switch among tasks more efficiently.

CCS CONCEPTS

• **Information systems** → **Browsers**; • **Human-centered computing** → *Graphical user interfaces*; *Web-based interaction.*

KEYWORDS

browser tab management, task management, to-dos, bookmarking, sensemaking, exploratory search, tab overload

ACM Reference Format:
Joseph Chee Chang, Yongsung Kim, Victor Miller, Michael Xieyang Liu, Brad A. Myers, and Aniket Kittur. 2021. Tabs.do: Task-Centric Browser Tab Management. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3472749.3474777

This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.

UIST '21, October 10–14, 2021, Virtual Event, USA

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8635-7/21/10.
https://doi.org/10.1145/3472749.3474777

1 INTRODUCTION

Despite our browsers being responsible for how we accomplish an increasingly significant proportion of the tasks in our professional and personal lives [13, 20], browser interfaces for managing those tasks have changed little in the past 20 years since tabbed browsing was popularized [13]. Today's internet users interact with a dramatically different web than that of two decades ago which has grown tremendously in size and complexity [40]. The amount of time the average internet user spends online has also grown: when tabs were introduced to the Mozilla browser in 2002, people spent on average 7 hours online per week;[1] that number is now approaching 7 hours per day.[2]

The mismatch between the growing size and usage of the internet with relatively static web browser interfaces suggests the possibility that the original tabbed browsing paradigm may no longer be sufficient for today's complex online tasks. There is mounting evidence for this, including dozens of popular press articles characterizing issues such as "Tab Overload" or "Tab Hoarding" [26–28, 41, 44, 45] as well the rise of bookmarking tools, such as Pocket (over 1 billion pieces of content saved) and Pinterest (over 450 million users), and tab management tools such as OneTab or SessionBuddy (over 1 million users each [21, 23]) aiming to reduce the number of open tabs users have open. Some browsers such as Chrome and Firefox have introduced or are experimenting with enabling users to combine several tabs into a single group to help with tab overload. The general approach taken by the above tools is to save tabs and close them, either individually, as groups, or as whole sessions, putting them out of sight, enabling users to free their attention and reduce clutter while being able to (in theory, at least) reload those tabs later.

However, a recent study interviewing information professionals and surveying a wider audience, many of whom tried using solutions such as the above, points to more fundamental problems with tabbed browsing that raises concerns about the above approaches [13]. Specifically, [13] noted that browser tabs are often used for a variety of task management functions that they were not necessarily designed for, ranging from reminding to prioritization, but function suboptimally for doing so. For example, users keep tabs open so they can resume progress on their tasks but cannot easily switch focus between sets of tabs for their different tasks; as

[1] https://theharrispoll.com/wp-content/uploads/2017/12/HI-Harris-Poll-Time-Spent-Online-2009-12-23.pdf
[2] https://datareportal.com/library

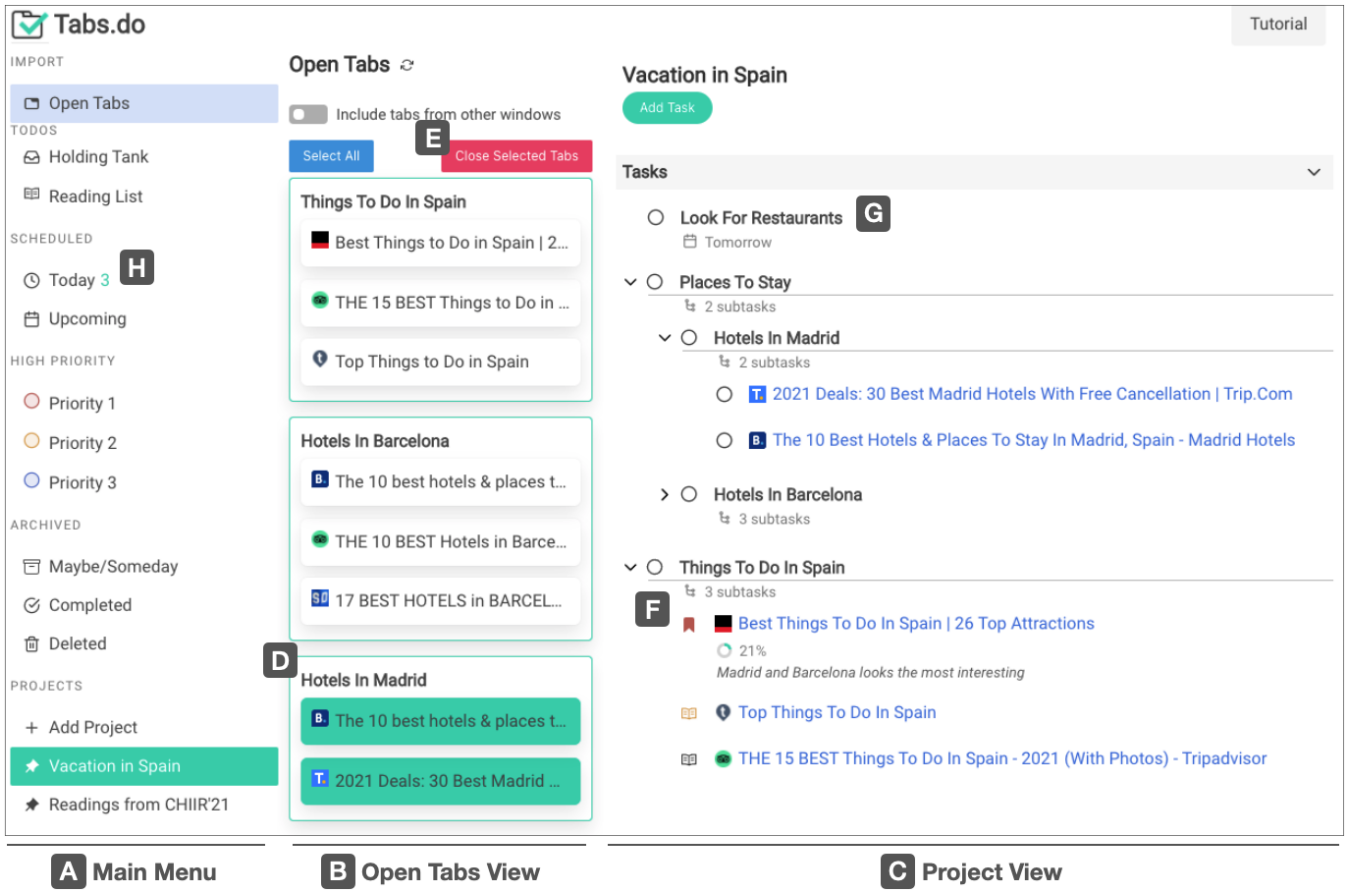

Figure 1: The main interface of Tabs.do that replaces the new tab page after installation. [A] The Main Menu for switching among projects and views that filter tasks with different priorities across projects. For example, [H] the Today View lists tabs that are due today. [B] The Open Tabs View allows users to import their tabs into Tabs.do via dragging and dropping into project labels in [A] or the current project [C]. [C] The Project View contains a list of tabs and tab bundles saved by the user. [D] Open tabs are automatically grouped by a deep learning model so that users can more easily import them as tab bundles and close them by clicking [E] to remove tab clutter. [F] Users can create hierarchical structures for their saved tabs to reflect their mental models; or [G] create a manual task similar to general task management applications.

reminders that quickly lose reminding value as they pile up; as reading lists of items that are never actually read and result in clutter; and as manifestations of their mental models that are artificially forced into a simple, temporal and linear list [13]. This suggests a divide between current browser designs that treat browser tabs as stacks of individual webpages and users who see bundles of tabs as their current and future tasks [13].

As a result, many attempts to address issues with tabbed browsing by addressing the surface level problem of closing tabs run the risk of conflicting with tabs' implicit task management functions. For example, bookmarking and closing tabs results in a lack of reminding and resurfacing functionality that users describe as leading to a "black hole" effect in which closed tabs are unlikely to ever be encountered again [13]. Other issues result from approaches such as tab groups, which may serve as a temporary stopgap but can result in overload as their numbers grow. More sophisticated approaches allow users to create workspaces of tabs that they can suspend and resume, such as in the Toby or Workona tab managers [22, 24]. However, while these workspaces can work well for relatively static tasks, users noted the challenges of manually creating and managing static workspaces for complex online tasks that involved collecting and organizing information that were constantly changing in priority or relevance. They also noted the challenges with evolving tasks that were too small, ephemeral, or undeveloped to merit their own workspace, but were still important to manage and keep track of.

In this paper we introduce and explore the idea of a task-centric approach to managing browser tabs that bridges the gap between managing individual browser tabs and managing users' online tasks and subtasks. Task-centric approaches have been shown to work well for domains such as file systems, application windows and email [2, 3, 7, 31, 54], suggesting there may be a profitable design space to be explored for browsing as well. However, while browsing, users are often exploring and sampling items from a nearly infinite

space for a myriad of purposes, leading to tasks that are often more ad hoc, uncertain, and ephemeral than the traditional projects and desktop applications that prior systems have targeted [13, 38]. To investigate these challenges, we instantiate a task-centric tabbed browsing approach in a prototype browser extension, Tabs.do, and evaluate its effectiveness in a field deployment (Figure 1). The basic intuition behind Tabs.do is that tabs can be "bundled" together and treated as tasks, with the system providing task management functionality such as reminding, prioritization, complex structure, task switching, and support for tasks both early and late in maturity. To further lower the friction of importing open tabs into the system, Tabs.do uses machine learning to make predictions about which open tabs correspond to the same tasks by exploiting behavioral and semantic features, allowing users to drag and drop groups of tabs into the system to create pre-labeled, bundled tasks. To protect users' privacy, the task prediction model runs locally inside users' browsers, so that Tabs.do does not transmit information about users' open tabs before they explicitly save their tabs into our system.

After several months of internal usage and iteration by the research team, we conducted a field deployment study with participants using Tabs.do with their real-life tasks and tabs. Based on interviews and log data, we found evidence that Tabs.do allowed participants to create rich task structures from their tabs with lowered interaction costs, to keep fewer tabs and be more focused on their important tasks but also context-switch among tasks efficiently when needed.

The contributions of this paper include:

(1) Exploring the idea of a task-centric approach to tabbed web browsing that aims to support the ad hoc and exploratory aspect of web browsing in addition to more stable collections of documents and resources.
(2) A prototype browser extension, Tabs.do, instantiating this approach through supporting a set of task-management affordances that help users manage a variety of browsing tasks. Our system enables users to create complex task structures by grouping and nesting tabs, allows them to fluidly suspend and relaunch tasks to reduce tab clutter, reduces friction through automatic task suggestions, and helps users manage attention through task prioritization, scheduling, and a variety of task types and statuses.
(3) A field deployment study with participants showing that Tabs.Do changed the way they interacted with their browser tabs on their own real-world tasks, helping them reduce tab clutter and increase focus.

2 RELATED WORK

2.1 Tabbed Web Browsing

Tabbed web browsing behavior was extensively studied when major browsers started to support tabbed interfaces [11, 20, 51, 56, 57]. This early work focused on the gradual user adoption of tabbed browsing and its benefits over using only browser windows. For example, researchers observed that the use of the back button decreased from 40% in the mid-90s to 7% in the mid-2010s when browsers that supported tabbed browsing reached 50% combined market share [55]. This suggested users preferred opening links using tabs and switching among them instead of loading multiple webpages using the same tab [33]. More closely related to our work, Huang et al. [34] estimated that 60% of users' browser tabs are related to at least one other tab of the same task. Our work builds on this observation that users often open multiple tabs to support the same task, and provides mechanisms for users to group them together as task bundles. More recently, Bento Browser [29] explored a search-centric mobile browser that scaffolds users search tasks by treating all search results as opened or unopened tabs. In the current work, we also exploit users' search activities to scaffold their task structures. However, instead of forcing tabs opened from a search results to be grouped together, Tabs.do uses a machine learning model that considers search activities and other behavioral and semantic features at the same time to produce tab grouping suggestions. This allows Tabs.do to produce grouping suggestions for tabs not opened from search results, and allows users to make adjustments to the grouping suggestions to better fit their mental models and correct mistakes made by the model. Most directly related to this current work is a recent interview and survey study that investigated tab management issues users face today [13]. At a high level, one of the main observations was that users need support managing their tabs based on the tasks they were conducting [13]. In the current work, we explore taking a task-centric approach to tab management by allowing users to group their tabs and save them into the system as tab bundles. The system, in turn, provides affordances to manage them as tasks. For example, users can also group their tabs into tasks and subtasks to better reflect their mental models, context-switch between their tasks or subtasks more efficiently.

2.2 Task Management

The idea of assisting users in better managing their attention by grouping related applications, files, and contacts by task contexts has been extensively explored in the *activity-based computing* literature [4, 5]. Early systems focused on building workspaces for the desktop environment [2, 3, 31, 54], for example allowing users to group application windows by different virtual desktops and switch among them [31], organize files and application shortcuts in a 3D desktop environment [3], or create integrated workspaces by grouping application windows, files, and contacts for knowledge work [54]. In the current work, we build on ideas from prior research in activity-based computing, but focus on the important yet relatively unexplored context of managing browser tabs and its unique research challenges. Specifically, tasks in the browser are often more ad hoc, uncertain, and ephemeral than the traditional projects and desktop applications that prior systems have targeted [13, 38], reflecting that users are exploring and sampling items from a nearly infinite space for a myriad of purposes in addition to keeping track of a finite set of their own documents. This fundamental difference introduces challenges such as allowing for fluid task structures that can support externalizing small tasks of uncertain importance that can later become full-fledged projects once explored more deeply [38], prioritizing and reminding users of their previously suspended tasks to avoid a black hole effect often associated with bookmarking [13], and de-prioritizing low importance tasks to avoid clutter while coping with users' aspiration to collect and process too much information [13].

Closely related to our work, another thread of research in the early 2000s explored how emails often represented users' tasks and explored how providing task management affordances in email clients can benefit users [7] – for example, prioritizing and scheduling deadlines for individual emails or grouping multiple messages, threads, and attachments into larger tasks. Since email tasks are by nature collaborative (i.e., receiving and delegating work with others), many prior systems focused on deadline management, communication, and coordination with others in business settings. In this current work, we also assume that tabs are often seen as tasks by users in both personal and professional scenarios and explore a task-centric approach to managing them. While there is also prior work that looked at the real-time collaboration of web browsing (i.e., collaborative search), in this work, we focus on managing browser tabs in a single-user scenario and see collaboration as potential future work. Fundamentally, as the browser has increasingly become the primary "habitat" [7, 58] of our digital tasks as desktop applications and communication channels continue to migrate to web-based platforms [20], it is crucial for research to understand better ways to support task management in the browser environment.

3 DESIGN GOALS AND MOTIVATIONS

When developing our design goals, we look to a recent interview and survey study on modern tab browsing behavior that focused on the issues that users face when managing their online activities using browser tabs [13]. The study provided deep qualitative insights based on interviewing ten researchers four times each over two weeks to sample their open tabs on their work computers, combined with survey data from another 103 participants. That study outlined a set of issues preventing users from closing their tabs, causing serious adverse effects, and developed a set of design implications for future browser interfaces. Here, we summarize three core issues and implications from this prior work [13] that were the primary motivations when designing our system:

Firstly, [13] found participants often saw browser tabs as external mental models of their online tasks, but the simple linear structure of tabs often insufficient for capturing their complex task structures with tasks, subtasks, and notes from reading the webpages. This led users to resort to solutions incurring high interaction and computing costs; for example, some participants simulated hierarchies using multiple browser windows, multiple browser applications, or multiple computers. Others used external tools such as word processors or spreadsheets to keep track of their tasks by copying and pasting URLs.

Secondly, [13] pointed to how users manage their attention at the task level, yet the current browser design makes it costly for users to context-switch between sets of tabs supporting their different tasks. While built-in features such as Chrome's tab groups and tab management extensions such as Workona support creating task contexts from multiple tabs, the initial manual cost of grouping tabs to create them can be prohibitively high. Specifically, when creating a new task context, users would still need to go through each of their open tabs to gather ones that are relevant to avoid losing important tabs.

Thirdly, [13] found participants had tabs that corresponded to tasks of varying importance, ranging from urgent and important tasks to casual readings that they may never get to but nonetheless did not want to put out of sight for fear of never re-encountering them. However, browser tabs have the same visual saliency (i.e., the same tab-width) and are ordered by default by creation time in a simple list, making it difficult for users to prioritize their tabs and focus their attention on important tasks.

Motivated by the three core issues summarized above, we list our core design goals as follows:

- **[D1]** Allow users to group tabs into task bundles and context-switch at the task level.
- **[D2]** Allow users to create rich and fluid structures that better reflect their evolving task mental models.
- **[D3]** Allow users to prioritize and de-prioritize their tabs so they can focus on their important tasks.

4 SYSTEM DESIGN

4.1 Fundamental Primitives and Design Approach

The fundamental primitive that Tabs.do introduces to support the above design goals is the *tab bundle*. A tab bundle can hold zero or more tabs and zero or more tab bundles, which can be nested to an arbitrary level of depth via drag and drop (Figure 2). If empty, a tab bundle consists only of a text title, which essentially acts as a to-do item as found in a typical to-do manager (Figure 2 C). Tab bundles have a variety of task functionality as described below. When containing one tab, the bundle is displayed and acts like a single tab (Figure 2 D), though it supports the same set of task functionality as when it is shown as a to-do. However, a tab bundle becomes particularly useful when it contains multiple tabs (Figure 2 E), at which point the entire bundle can be treated as a single task, and can be assigned a priority, scheduled (Figure 4), moved into a project (Figure 1), or otherwise managed as a task. Furthermore, tab bundles can be nested in other tab bundles to an arbitrary level (Figure 2), supporting complex task and subtask structures.

There are several common user patterns Tabs.do supports for transforming tabs into tasks and managing them using tab bundles. One approach is post-hoc task management, in which a user has already started a task through a search query and may have several tabs opened from that query or related pages. In this case a user can open the Tabs.do interface by opening a new tab page (see Figure 1), selecting the tabs they wish to bundle from the "Open tabs" pane (which can be auto-suggested by the system, as described below), and dragging them into another pane. At this point the system will create a bundle for them named using the terms of the search query it originated from if available (shown in previous work to be an effective heuristic for initial naming of search-based tasks [29]), or else the title of the first tab in the bundle. The user can then close the related tabs, while managing the task through the bundle's task functionality or resuming the bundled task when they wish.

Another common approach discussed in the literature is a priori task management, in which a user creates a placeholder for a task that they wish to complete later, potentially scaffolding that placeholder with multiple subtasks (e.g., creating a task for a trip to Barcelona, along with subtasks for restaurants, shopping, sights,

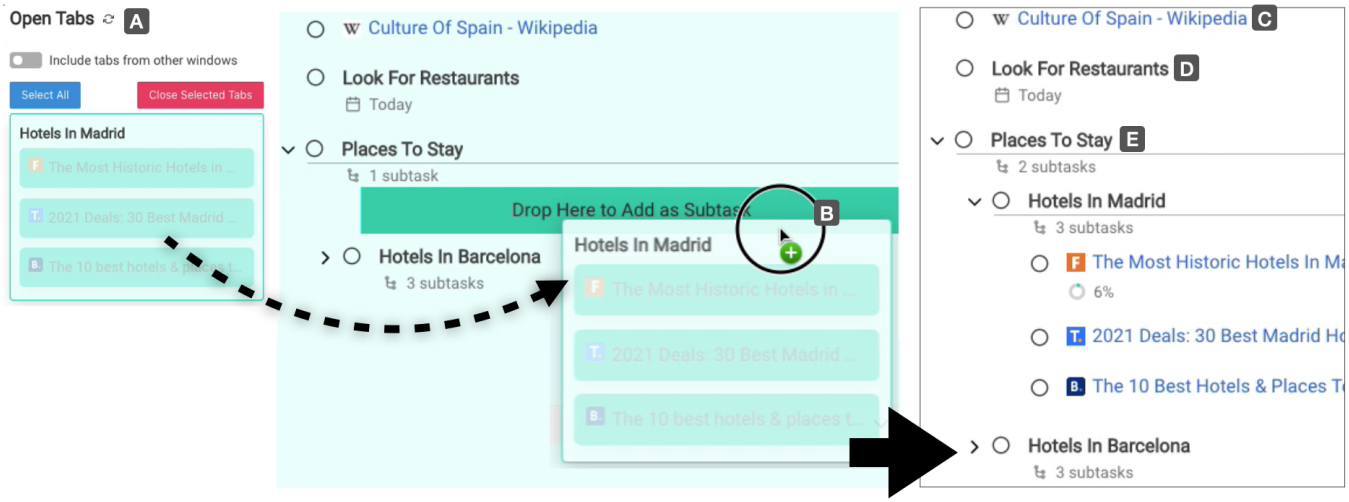

Figure 2: Users create and organize their tabs via drag and drop into their projects. They could create hierarchy from previously saved tabs or new tabs by dragging to a specific location in their task hierarchy from their open tab list in [A]. [A] A User drags a tab grouping suggestion from the Open Tabs view into a specific location [B] in the task hierarchy. When dragging, the light green background indicates the selected tabs that are being dragged [A] and area in the main view where they can be dropped [B]. This creates a new tab bundle nested under a previously created tab bundle [E].

and transportation). This approach is typical of standard to-do lists, in which users queue up the tasks they need to work on and use the list as a reminder [6]. In this case a user can add a manual task to the system (Figure 1 G) , and give it a title in the same way as they would a to-do and can similarly serve as a reminder and be scheduled, prioritized, etc. (By default, Tabs.do also adds a circle to the left of each item modeled after typical to-do list systems, which the user can use to check off and complete the item, or change it into another item type as described later.) When the user decides to work on the task they can add any relevant tabs under the to-do item by dragging them into it.

Finally, a user can simply add a single tab (technically, a degenerate tab bundle containing only one tab) by dragging it out of the "Open tabs" pane, at which point it appears as an item representing that tab, including the title and the favicon (or tab icon) of that tab, but supporting the same functionality as any other tab bundle.

Below, we describe the ways in which the core tab bundle primitive can be combined with various task-based functionality to support users' complex online task management needs. To ground this discussion we first describe it in the context of an example user experience, and then unpack details of the various system features and how they address the design goals above.

4.2 Example User Experience

Consider a university student who has been considering taking a vacation somewhere in Europe. She casually searches on Google for "things to do in Spain', and opens a few webpages in tabs. It quickly becomes apparent that she is most interested in two cities – Barcelona and Madrid. She starts to wonder about their lodging choices, so she creates two new searches for "Hotels in Barcelona" and "Hotels in Madrid" and opens a few hotel websites from each search results page. At this point, she has accumulated more than 15 tabs from the three searches and from opening more links into tabs as she read some of the webpages, and what started as a casual exploration to pass the time has quickly grown into a more intensive research session. She feels overwhelmed by her open tabs, and cannot easily switch focus between her tabs for researching *hotels in the two cities* and her tabs for researching *things to do in Spain*.

She opens Tabs.do and finds all her open tabs automatically grouped into three task bundle suggestions in the Open Tabs panel (Figure 1 B, D). To start a new workspace for them, she creates a new project in Tabs.do titled "Vacation in Spain." To save her open tabs in an organized way, she drags each of the automatic tab groups and drops them into the Project View to create three task bundles (Figure 1 C). Tabs.do automatically assigned the search terms as titles of the three task bundles. To further organize her tasks, she creates an empty task bundle titled "Places to Stay" and nests her two hotel task bundles under it (Figure 1 C). She then creates other empty task bundles titled "Restaurants in Madrid" and "Restaurants in Barcelona" nesting them under "Places to Eat" via drag and drop (Figure 2) as reminders for what she needs to research next. To continue her research, she closes all her tabs for researching hotels with three clicks in the Open Tabs view to first select the two tab bundles for "Hotels in Barcelona" and "Hotels in Madrid" and use the "Close Selected Tabs" button to close them (Figure 1 E). Now all her open tabs are about *things to do in Spain* again, she switches her focus back her initial subtask.

As she continues to read from the webpages, she opens Tabs.do's Popup Menu to save notes from the webpage she is reading (Figure 3 B). When she returns to her project in Tabs.do (Figure 1 C, F), she can see an overview of all the notes she took on the individual tabs. After a while, she notices in Tabs.do that there is a task bundle for a class assignment due today (Figure 1 H). She decides she should work on her assignment, so she closes all her tabs for Spain

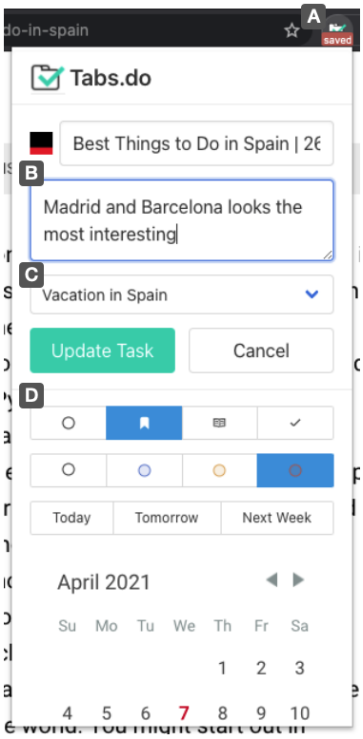

Figure 3: User can also access Tabs.do while they process information on the current tab using the Popup View via the extension button and without switching to the full interface (Figure 1). [A] If the current tab was previously saved, a "saved" badge will appear whenever user switches to this tab. Previously saved notes [B] and attributes, such as project [C] and priorities [D] are also reflected. the popup view allows users quick access to their notes about the page, allowing them to accumulate notes as they read from the individual tabs.

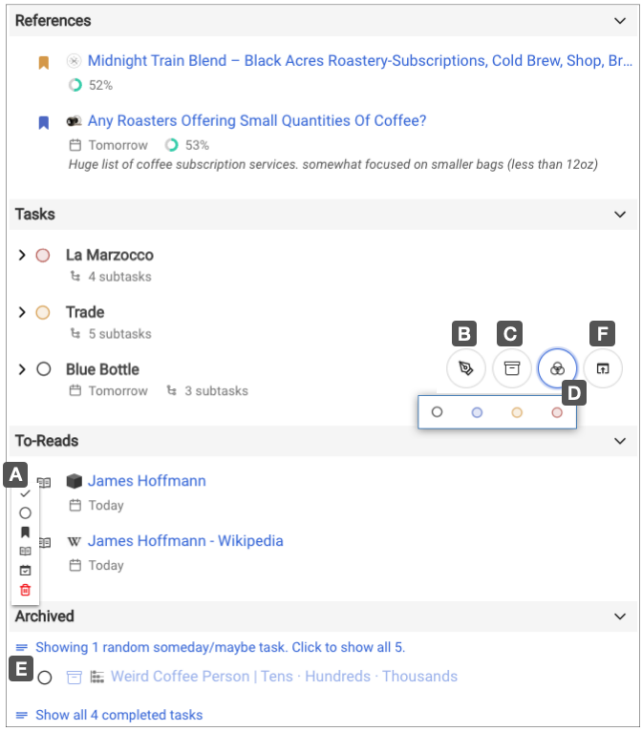

Figure 4: Tabs.do provides a set of affordances for prioritizing tab bundles. [A] Users can categorize tabs as References, To-dos, To-reads, Completed, or Deleted, which sorts it into different sections in their projects; [B] The edit button bring up an Edit View (identical to the Popup view in Figure 3) where users can assign due dates to their tabs. [C] Marking tabs as the "Maybe/Someday" compresses them at the bottom [E] but are not completely out of sight with one random tab showing. [D] Users can also assign color coded priorities, which sorts their tasks within the sections. [F] Users can open a tab bundle back into open tabs (independent to prioritization).

vacation research, feeling confident knowing that she can resume her research progress any time by reopening her task bundles into tabs from Tabs.do, and all her notes and scroll positions will be restored. Finally, she navigates to a Tabs.do project she had previously created for the class and reopens its task bundles back into open tabs (Figure 4 F), including presentation slides, her notes on Google Docs, and the link to the homework instructions to start working on her homework.

4.3 [D1] Task-Centric Context-Switching

As reflected in the example above, users often manage their attention at the task level and need support when switching focus between sets of tabs supporting their different tasks and subtasks – for example, switching from one set of tabs about *hotels in Barcelona* to another set about *hotels in Madrid*. Tabs.do supports this by allowing users to group tabs and save them as a tab bundle. To do so, users can click and select a set of open tabs listed in the Open Tabs pane (Figure 1 B) and use drag and drop to save them into a default holding area, a project they had previously created (detailed below), or a new project. On creation, the system automatically generates a title suggestion for the tab bundle so it is easier to recognize in the future (e.g., *Hotels in Barcelona*), and it can be edited by the users. After saving, users could use the *close selected tabs* button to close the set of tabs (Figure 1 E). Creating tab bundles in Tabs.do enables users pause and resume progress at the task level. To resume a previously closed task, users can reopen tabs under a bundle using button that shows up on hover (Figure 4 F), either in the current browser window or in a newly created browser window. Tabs.do automatically restores their scroll positions so users can more easily resume their progress.

One key challenge here is the cost of sifting through open tabs to group all relevant tabs and naming them afterward, especially for users who keep a large number of tabs opened. To lower the interaction and cognitive costs of this process, Tabs.do uses machine learning to make tab grouping suggestions in the Open Tabs pane by showing a green border around the suggested name with a set

of tabs (Figure 1 D). Users can save a suggested tab group using its title as the handle for dragging and dropping into the Project view (Figure 2 and Figure 1 C). Alternatively, before dragging, they can click on the title to select tabs in the bundle suggestion which allows users further select or deselect tabs to fix any mistakes made by the machine learning model. The automatic task grouping lowers the interaction costs of creating tasks as well as giving users a better overview of their open tabs even before saving them (Figure 1 B).

To generate the task grouping predictions, we collected browsing history from four authors and labeled them to train a neural network model that can make predictions about which tabs belong to the same tasks. We used TensorFlowJS as our machine learning library [48], which allowed us to distribute the trained model with the extension to make task predictions inside users' browsers. This design has the benefit of allowing Tabs.do to make predictions about users' open tabs without having to transmit to a remote server their browsing history which may contain sensitive personal information. Detailed description of this dataset, the task prediction model, and its accuracy is described in Section 4.6.

4.4 [D2] Task Mental Models

While tabs are typically instantiated as a linear, temporally ordered list, users' task structures are often more complex. In our example user's scenario above, a vacation to Spain had several subtasks including researching places to stay and places to eat, in both Madrid and Barcelona (Figure 1 C). To support this task structure, tab bundles can be nested within other tab bundles by dragging and dropping them (Figure 2), acting as subtasks that can be expanded and collapsed and given different priority levels, notes, or other task functions. Nesting can be done to arbitrary levels of depth; to address issues with real-estate and visual clutter at high levels of depth the system provides a "focus" button which fills the view at the selected level of depth with a breadcrumb allowing them to exit the focused view.

Another challenge with task structure is supporting different types of projects and projects at different stages of progress. One common task type involves the long term collection, organization, and re-access of content, such as collecting content relevant to a field of scientific study, a kitchen remodel, a design mood board, a course being taught, or a term project for a course. To support such tasks, Tabs.do allows users to define tab bundles as long term projects, which have a privileged position in the Main Menu (Figure 1 A). Such projects are a familiar metaphor and correspond to the use of workspaces in the Workona or Toby tab manager, or to projects in to-do list tools such as Todoist. Unlike in such tools, we aim to address the challenge that even the number of long term projects can grow unwieldy and can go in and out of relevance over time; to support this we enable users to pin projects to the top of the list, similar to pinning messages in an email client. Although potentially a mixed metaphor, we found this to work well in practice.

However, the larger challenge in supporting various task structures are the many tasks in the long tail that are short term, ephemeral, or in the early stages, and which often outnumber the set of long term projects [8, 13, 29]. Users find projects and workspaces too heavy for such tasks [13], requiring too much effort to create and, more importantly, to get rid of or refactor; as well as "polluting" their important long term projects with a large number of short term or less developed tasks. To address these tasks we introduce a holding tank which acts as the default view for participants' tab bundles. The holding tank aims to make it easy for users to throw in tab bundles, single tabs, or even manual to-dos with no tabs attached without spending the cognitive effort to figure out how to structure and organize them and without polluting their curated project information space. Such tasks can act as reminders for the user to come back to them; can be easily removed by changing their status to "completed", deleting them, or simply ignoring them as they drop below the fold; and can be refactored into larger tab bundles by dragging and dropping them.

Beyond creating structures, Tabs.do also provides two mechanisms for keeping track of users' progress on their individual tabs. Firstly, users can save tabs into Tabs.do when they are reading from webpages in their open tabs without switching into the main interface of Tabs.do. To do so, users can click on the extension button to see the Popup view of Tabs.do (Figure 3), allowing them to save the current tab and set detailed attributes for it, such as priority and due date (Figure 3 D). To help user maintain task context, the Popup view saves tabs into the most recently accessed project from the same browser window (Figure 3), but users can also select a different project or create a new project (which changes the project context for the browser window). In the Popup view, users can also change the title of the tab and take notes to externalize useful information they gathered from the current page and use Tabs.do as the external memory for their task. Whenever users open or switch to an open tab that was previously saved, a "saved" badge appears on the extension button. Users can open the Popup view to access previously saved notes to remind them of their progress, and accumulate more information by editing the notes field in the Popup view.

Secondly, Tabs.do proactively estimates the reading progress of each tab to help users remember the level of progress they had made (Figure 1 F "21%"). To do so, Tabs.do tracks the scroll position and focus state of each tab using the `scroll`, `blur`, and `focus` JavaScript events. Combined with the tab's viewport height, Tabs.do generates a heat map in the background of how many seconds different regions on the webpage were in the viewport while the tab was in focus. Finally, Tabs.do generates a reading progress estimation based on the heat map assuming users process 100 vertical pixels per second. While there may be other more sophisticated approaches for progress estimation, such as analyzing page content [25], this simple heuristic was straightforward to implement, required minimal computing resources at runtime, and worked reasonably well during our own testing. This estimation is calculated locally on all open tabs, but only synchronized with the backend database for tabs that the user had saved to Tabs.do. This is so we only obtain information for tabs that users had explicitly saved into our system to avoid tracking private information.

4.5 [D3] Prioritizing

One issue with the current browser tab design is that it does not reflect users varying task types and priorities. More specifically, tabs represent frequently visited references, important but unfinished

tasks, or casual readings picked up from social media. However, besides favicons, tabs have the same visual saliency (i.e., tab-width), making it difficult for users to prioritize their tasks and focus on the task at hand [13]. Tabs.do addresses this by providing four prioritization mechanisms that can be flexibly combined to address different user needs (Figure 4):

Status-based: Tabs.do allows users to categorize their tabs into five general statuses (Figure 4 A) indicated by the leading icon of each saved tab: to-do (circle icon), to-read (book icon), reference (bookmark icon), completed (check icon), and deleted (trash icon). Saved tabs are automatically sorted under collapsible sections based on their statuses. References are sorted at the top for quick access, followed by to-do, to-read, completed, and deleted. The completed sections are collapsed by default so that they do not distract users from their primary tasks (Figure 4 E), and deleted items are moved into a global Trash Can view accessible from the menu on the left (Figure 1 A).

Priority-based: Users can prioritize their tabs by assigning priority 1 to priority 3 (Figure 4 D), which changes the color of their status icon from gray to red, yellow, and blue, respectively. The priorities are used as a secondary sort key; within each status section, priority 1 tabs are sorted to the top of the section, followed by priority 2, priority 3, and default priority (unassigned).

Schedule-based: An alternative way to prioritize tabs saved in Tabs.do is to assign a due date to them. Tabs with a due date assigned to them have a calendar icon under their title, followed by the due date. Scheduling a tab does not change its order, but puts it into a global scheduling view accessible from the menu on the left side.

Someday/Maybe: Prior studies in both general task management and browser tab management have identified that some users tend to keep low-priority tasks or tabs that they do not expect to ever complete [6, 13]. To prevent low priority tasks creating clutter in users' workspaces, Tabs.do allows users to mark tasks as *Someday/Maybe* which moves them to the bottom of their project view (Figure 4 E). Similar to completed tasks, Someday tasks are collapsed by default to prevent clutter. One challenge here is that users do not want their tasks out of sight (e.g., such as if saved as bookmarks) to avoid the "black-hole" effect in which when tasks become out of sight, the chances of completing them are significantly reduced [13]. For this, Tabs.do encourages users to mark their tasks as *Someday/Maybe* by showing a random low-priority item every time users open their project so that they are not entirely out of sight (Figure 4 E), yet also do not clutter users' workspaces.

While we provided four mechanisms for prioritizing tasks in Tabs.do, we do not expect users to utilize all four mechanisms. Instead, our goal was to provide prioritization mechanisms flexible enough to accommodate different users, as prior work in general task management has pointed to users having varying strategies when prioritizing their tasks [30].

4.6 Automatic Task Bundle Suggestions

To lower the user costs of adopting the system, Tabs.do provides tab grouping suggestions to make it easier for users to save tab bundles into the system. Driving this feature is a deep learning model that segments browsing history into sessions containing page-loads supporting the same tasks. While page-loads are not equivalent to tabs (i.e., a user could load multiple pages over time in the same tab), this approach allowed us to more easily collect training data using the Browser Extensions History APIs.[3]

We collected a small set of labels to train and test our model where four research team members (a designer, a product manager, and two researchers) provided their recent browsing history resulting in a total of 2,278 page-loads. The first two authors went through each page load in their history to identify whether it was either the beginning of a new task session or not. To ensure labeling consistency, the first two authors first labeled 10.9% of the data independently and compared their labels. The two sets of labels had a high agreement level (Cohen's $k = 0.901$, $p<0.00001$, $N=248$), so they proceeded to label the rest of the dataset without duplication. After labeling all 2,278 page-loads, 23.8% were labeled as the beginning of a new task session. We used the labeled dataset to train a simple feed-forward network with three hidden layers of 32, 64, 128 nodes, respectively, and used ReLu6 as the activation function [36]. We used the below six features to make predictions about whether a page-load in history is the beginning of a new task session or not.

(1) Title similarity based on Universal Sentence Embedding vectors [12] between the current and previous page load
(2) Normalized Levenstein distance between the URLs of current and previous page load
(3) Normalized Levenstein distance between the domains of current and previous page load
(4) Whether the current page load was a Google search
(5) Whether the previous page-load was a Google search
(6) The number of times this URL was visited in the past
(7) The number of times URL was entered in the address bar
(8) Seconds between the current and the previous page load

These included features based on semantic similarity between the page load and its previous page load (features 1 through 5) and behavioral features based on users' past interactions with the webpage (features 6 through 8). Features 1 to 3 were designed to capture task topic changes by measuring the semantic similarity between titles, URLs, and domain names [12]. Features 4 and 5 were based on the intuition that many online tasks begin with a web search to fulfill some information needs. Features 6 and 7 were based on the intuition that frequently visited portal pages, such as Google Drive, are often used as task launchers. Finally, feature 8 was a feature commonly used by search engines for identifying new search topics [35]. At runtime, the model makes predictions on users' browsing history that covers their open tabs. We then extract query terms from any Google search results pages to use as the suggested bundle name, and if there were no Google search results pages within the suggested tab bundle, we use the title of the first tab within that bundle.

4.6.1 Prediction Accuracy. We randomly sampled 80% of labeled data for training, 10% for validation to prevent over-fitting, and 10% for testing model accuracy. The complete model trained on all eight features had an overall labeling accuracy of 92.1% (precision: 0.86; recall: 0.83; F1: 0.84 for the start-of-task label). We further compared the labeling accuracy for using only semantic features versus only behavioral data. Results showed that the model trained

[3] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/history

on semantic features (1 through 5) had a labeling accuracy of 76.3%, and the model trained on behavioral features (6 through 8) had a labeling accuracy of 86.4%. This result suggests that both semantic and behavioral features contributed to the higher accuracy of the model trained on all features. To reduce model complexity and improve runtime efficiency, we iterated through different feature combinations to see if we can use only a subset of features and achieve similar performance to the complete model. In the end, we used feature 1, 2, 3, 4, and 8 to generate the model we used in the extension, which had an overall labeling accuracy of 90.8% (precision: 0.84; recall: 0.79; F1: 0.81 for the start-of-task label).

We acknowledge that this preliminary result was limited by the size of our dataset, and the accuracy of the model in a field deployment will likely be lower than on the test set due to behavioral and task topics differences between individuals. To address this, the Open Tabs pane (Figure 2 A) allows users to recover from the model's mistakes by first clicking on a bundle's title to select all tabs in it, and then unselect tabs that do not belong in the same task, or select additional tabs to include them. While more sophisticated models with larger training data could further improve accuracy, in this current work, we focused on examining the effects of providing automatic task bundling suggestions on user experience holistically by conducting a field deployment study of Tabs.do.

4.7 Implementation Notes

In order to produce a research prototype that is robust enough for a field deployment study, we spent eight months developing Tabs.do as a browser extension while the research team used the extension ourselves for the last four months to identify bugs and usability issues. Admittedly, modifying the browser program would have allowed us to explore the design space of changing existing tab interfaces, but we think it is a reasonable trade-off for the significantly lowered development effort required for browser extensions and is also sufficient to test our task-centric approach for managing browser tabs.

Tabs.do was implemented in approximately 13,000 lines of TypeScript and used the ReactJS library and the Bulma CSS framework for building UI components. Firestore was used for backend functions, database, and user authentication, which allowed our participants to access their tabs across devices. For privacy concerns, TensorFlowJS was used to drive the task bundle prediction feature [48], which allowed Tabs.do to make tab grouping predictions locally on participants' computers without sending their open tab information to a backend server. Tabs.do was implemented as a cross-platform browser extension using the now standardized Web Extensions APIs, but we only recruited participants who used Google Chrome and Microsoft Edge as their primary browsers during the field deployment study to minimize testing efforts during development.

5 FIELD DEPLOYMENT STUDY

To understand how our task-centric approach can benefit users and to evaluate Tabs.do, we conducted a field deployment with participants performing their everyday tasks in the wild. Ten participants were recruited by posting to authors' social media feeds and online forums (mean age: 29.70; SD=8.97; 6 male, 3 female, 1 non-binary; 4 students, 3 software engineers, 1 faculty, 1 account manager, 1 entrepreneur). The posts were brief and asked for participants who have ever "*felt overwhelmed by their browser tabs*." The posts also contained a link to an online screener survey to recruit participants who used Chrome or Edge as their primary browsers which were the two browsers that we tested during development. Each participant was interviewed remotely before using Tabs.do and after using it via a video conferencing service that supported screen sharing. The pre-interviews lasted around 20 minutes which covered collecting their consent and demographic information, a brief walk-through of the interface, and installing the extension on their personal computers. We then scheduled each participant for a 30 minute post-interview approximately one week after installation, determined by their availability. During the post-interviews, participants shared their screen and performed a retrospective walk-through of their usage of the system. All 10 participants completed the study and were each compensated a 50 USD Amazon gift card for their time. The interviews were recorded (both audio and video) and transcribed for an open coding analysis to capture rich qualitative insights grounded in data [9, 17]. The first author went through the 5 hours of recordings and transcriptions in three passes to iteratively highlight interesting quotes, generating summaries and potential categories until clear high-level themes emerged. Throughout the iterations, inputs from the third author who conducted the interviews were also incorporated. This study was approved by our institutional review board.

5.1 Results

In general, participants responded favorably in the interviews about their experience with Tabs.do during the week-long study, using Tabs.do to manage their tabs supporting both their personal and professional tasks in the wild. Log data showed that participants were actively engaged with the system during the week-long deployment (Table 1). We examined the log data and found that 7 out of the 10 participants continued to be actively engaged with Tabs.do on a daily basis at the time of writing (or for more than 10 weeks total). Considering participants still had to endure a few bugs in our research prototype and were under no obligation nor rewards for the continued usage after the study had concluded, we see this as an encouraging indication that our task-centric approach continued to provide value to our participants. Below we list the most common themes from coding the interviews to provide in-depth understandings of how participants interacted with Tabs.do during the field deployment study.

5.1.1 ***[D1] Task-Centric Context-Switching.*** Tabs.do enabled users to manage their attention at the task-level by introducing our basic primitive of tab bundles that groups sets of tabs supporting the same task together. Once created, tab bundles allowed users to "relaunch" a task by reopening tabs in the bundles. Log data showed that our participants were actively using our tab bundle primitive. For example, on average, each participants saved 50.8 tasks (SD=26.2) to Tabs.do and nested them 45.5 times (SD=26.2). This included both dragging a group of tabs from the Open Tab view (Figure 2) to create a tab bundle, as well as using drag and drop to create bundles from previously saved tabs. In the interviews,

Table 1: Behavioral log data from participants in the field deployment study. Session: The number of times participants opened and interacted with Tabs.do, either from the new tab page (Figure 1) or the popup view (Figure 3). Tasks: The number of times participants created a manual task or saved an open tab. Nest Tasks: the number of times a task was nested under another; for example, if a bundle with 5 tabs were saved, it counts as saving and nesting 5 tabs. Reopen: number of tabs opened from Tabs.do either by clicking on the title of a tab or using the reopen button on a bundle (Figure 4 E). For example, if a bundle with 5 tabs were reopened, it counts as 5 openings.

	Action Count	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	M	SD
Session	New Tab Page	58	11	21	15	28	11	22	28	11	3	20.8	15.4
	Popup View	19	4	39	4	7	3	6	4	14	11	11.1	11.1
	Total	77	15	60	19	35	14	28	32	25	14	31.9	21.1
	Create Project	11	1	5	6	4	4	2	1	4	2	4.0	3.0
Task	Create Manually	10	5	8	10	10	8	4	8	8	2	7.3	2.8
	Save Open Tabs	63	19	50	70	49	37	11	47	78	11	43.5	23.8
	Total	73	24	58	80	59	45	15	55	86	13	50.8	26.2
	Nest Tasks	65	33	36	48	75	44	8	49	86	10	45.4	25.4
	Edit Notes	7	7	2	0	12	7	7	2	0	12	5.6	4.5
Reopen	Individual Tabs	42	2	7	4	3	6	14	13	6	1	9.8	12.1
	From Bundles	25	13	12	0	12	7	2	5	95	4	17.5	28.2
	Total	67	15	19	4	15	13	16	18	101	5	27.3	31.3
Priority	Set Status	10	1	27	30	11	5	1	11	1	3	10	10.6
	Set Priority	1	3	1	3	12	5	0	4	0	5	3.4	3.6
	Set Due Date	1	1	2	8	31	3	4	0	0	0	5	9.5
	Set Maybe/Someday	1	2	0	1	2	3	2	12	0	2	2.5	3.5
	Total	13	7	30	42	56	16	7	27	1	10	20.9	17.6
Total Usage in Minutes		49.6	13.1	31.2	43.6	51.4	24.3	21.5	20.7	27.2	16.0	29.9	13.8

participants were enthusiastic about the ability to create tab bundles, especially when they first discovered the automatic bundle suggestions:

> "Having subprojects [task bundles] and projects is really helpful, because that's kind of what my workflow looks like. It's like, I have these four tabs that are related to this new post I'm writing and these [other] four tabs are related to analytics, and like being able to organize them, I think that's a big benefit." – P3

> "I saw when it automatically grouped them before I even did anything. So that was so helpful. I was able to just make them into a task, like a big bundle... and then from there just close them out completely and know that I could come back to it. So that was really, really helpful." – P6

Many participants also frequently reopened individual tabs from Tabs.do, averaging 27.3 reopened tabs (Table 1). This suggests that users consider reopening from Tabs.do to be more efficient compared to 1) keeping and switching to a set of tabs; or 2) re-tracing their steps for opening them in the first place (such as using same query on Google, as reported in [1, 52]). On average, we found more tabs were opened from "relaunching" tasks using tab bundles (M=17.5; SD=28.2) than from clicking on and opening individual tabs (M=9.8; SD=12.1), although the difference was not significant under a paired T-Test (t(9)=-1.965,p=0.08). Closer examination suggests a bimodal distribution with some participants strongly preferring reopening sets of tabs using bundles while others preferring opening individual tabs. During the interviews, participants described how the ability to reopen sets of tabs from bundles allowed them to context-switch at the task level and remind them of all the subtasks they needed to completed:

> "I was able to kind of switch to another task, and then close all those tabs [referring to the task she switched away from]. So starting again on a task I was working on is pretty easy... To open them all at once, kind of also remind me of all the steps I had to do." – P1

> "I work in batches, for example, in the morning, I come to the inbox [a project]. At night, 10:30, I come back to this [another project]. Somewhere between 5pm to 8pm, I go to Learning [another project] and open Kindle or Blinkist or Audible [referring to reopening tabs]. So according to what time of the day it is, it directly corresponds to what project I'm using. " – P4

The ability to "relaunch" tasks was also commonly mentioned with time-saving and lowered interaction costs when compared to not using Tabs.do:

> "So per day, it's probably saving me about anywhere from 30 to 45 minutes... just because that's the time that I would spend like searching through all of my assignments, and then opening them up and then trying to find each of the readings that I have to do separately and then open up in a separate tab" – P6.

One significant challenge brought up by prior work on tab overload is that users have trouble closing tabs because they serve

several task-centric functions, ranging from reminding to externalizing their working memory, resulting in clutter that, ironically, reduces the effectiveness of those task functions [13]. We instead found our participants' expressing confidence closing tabs that they originally felt strongly attached to as a result of using Tabs.do, suggesting that our task-centric approach can address some of the tab issues brought up in prior work [13].

> "So one big thing is I used to have two windows open all day. One with personal stuff, and one with work stuff. Now I don't have the personal one open anymore. I basically used the tab manager to completely manage personal stuff that I wanted to get back to... So it prevented me from having two windows opening Chrome, which was the biggest gain; I didn't do that this whole week since I started using Tabs.do." – P3

> "Probably being able to close a bunch of tabs I had open for, like, days. Just because I didn't want to lose those tabs." – P1

5.1.2 ***[D2] Task Mental Models.*** Tabs.do supports capturing users' task mental models by allowing them to create a hierarchy of tasks with tab bundles as well as saving them into larger projects. Based on log data, each participant created an average of 4.0 projects (SD=3.0) and were actively creating nested tasks and subtasks from their tabs (an average of 45.4 times; SD=25.4). In the interviews, participants described how creating rich structures in Tabs.do allowed them to work in a more organized manner when compared to using the linear tab list of current browsers:

> "I would say that before the Tab Manager [Tabs.do], I didn't really have any structure or sense of priority of my tabs. They were just all just a mess. You know?" – P6

Interestingly, P6 further pointed to how the automatic tab grouping feature allowed her to have more situational awareness with her open tabs even before saving them, allowing her to find important tasks that she should focus on and encouraged her to create task bundles from the suggestions:

> "The automatic grouping is everything to me. It kind of puts me in the mindset that those things are related to each other, and that they are somewhat important. Even now, I'm getting the urge to group these [saving a tab group into the holding tank], because this is all related to my JavaScript homework... so it just kind of changed my relationship with my tabs." – P6

These suggested that Tabs.do has a low upfront cost for participants to start benefiting from the system. Specifically, before saving tabs into the system, the automatic task groupings can provide a better overview that promotes situational awareness than the built-in tab UI; and that even saving one tab bundle allowed participants to immediately close them confidently, knowing that they could relaunch their tabs when needed.

Participants also suggested potential features that would allow them to further benefit from the task bundling feature. For example, P1 pointed to a tighter integration between her task structures and the current tab by showing other tabs from the same tab bundle or project in the Popup View:

> "You could have the extension button be able to open up related tabs [to the current tab] Like tabs that are in the same project or subtask [tab bundle]." – P1

Tabs.do also allowed users to take notes in the Popup view (Figure 3) as they read from their individual tabs to use as external memory. Log data showed moderate use of the Popup view, accounting for an average of 35% of users' total sessions with Tabs.do, and 13% of all tabs saved on average. However, participants only edited notes 5.6 times (SD=4.5). Prior work in general task management showed that people tend to spent minimal effort when naming their tasks, often with short description enough to provide salient cues. This offers a potential explanation to the lower usage of note editing in Tabs.do, suggesting participants primarily used Tabs.do as a task management tool instead of a note-taking tool in its current state, in contrast to the findings of other lightweight browser note-taking tools [53].

5.1.3 ***[D3] Prioritization.*** Similar to prior work on general task management [30], we also found that our participants used varying strategies to prioritize their tasks in Tabs.do. Log data showed some participants who rarely used the prioritization features such as status, priority level, and due date (P2, P7, P9 in Table 1) as well as participants who used them extensively (i.e., P3, P4, P5, P8 in Table 1). There were also differences in how participants used the prioritization features. For example, P3 and P4 mostly marked tabs with statuses such as references and to-reads in order to pull them out into different sections, whereas P5 most frequently scheduled due dates for their tabs, and P8 used a combination of statuses and marking tabs as *Maybe/Someday* to de-prioritize them.
Upon further investigation during the interviews, it turned out that some participants who did not extensively use the built-in prioritization features did end up prioritizing their tasks in Tabs.do, but used more ad-hoc methods. Most commonly, participants used a combination of open tabs, the Holding Tank, and Projects to triage their tasks from lower priority to higher priority:

> "One-off research things... I don't think I would create a task for it... Like looking for recipes, I opened a lot of tabs, but then went through most of them and closed them within like 30 seconds or a minute each… And then I have the holding tank, which is like... just for one-off things that didn't belong in a project and were temporary, but longer than I guess, a minute or five minutes." - P1

Other strategies included using a *zero inbox* strategy (described in [58] as *frequent filers* for emails) in which users initially saved most tabs using the on-page Popup View into the Holding Tank (Figure 3), and frequently opened the main Tabs.do interface to subsequently catalog them into projects (Figure 1). Similarly, some used the Holding Tank to keep track of urgent tasks while creating projects to store longer term tasks that had a lower priority (P6):

> "Anything that was in the holding tank, I was either moving to the reading list or to one of these projects that I made... I consider the holding tank to be a place where you just throw things [in] so you can organize

> them... I would feel uncomfortable just leaving things hanging out in there."– P3

> "Projects are, like, I'm gonna get around to watch all that Anime [a project] and I'm gonna do this Art Challenge [another project]. These [my projects] are presents for the future... It's not an immediate thing. Whereas up here [in the Holding Tank], I'm like, okay, I have code [a task bundle] due tomorrow." - P6

One surprising finding was that few participants used scheduling features of the system, despite their ubiquitous presence in to-do list managers. When we asked about the lack of use of our scheduling feature, participants noted that many of their lower importance tasks do not have clear "deadlines." For their more important tasks in the browser, they pointed to their existing use of other calendar services (i.e., Google Calendar and the calendar feature in Notion), and instead suggested integration with third-party services as a feature that would make scheduling more useful.

> "It's [scheduling due dates in Tabs.do] kind of useless to me if I can't see it in my Google Calendar. Any kind of integration in the future would just be great. I kind of live and die by my Google Calendar... If it's not in my Google Calendar, it's not really gonna happen" - P6

In sum, we found participants used a wide range of different approaches to better prioritize their tasks in the browser. As a result, participants said they were able to be more focused on the task at hand and not be distracted by all other tasks that they had accumulated.

> "It's made me more focused on whatever I'm working on right now, and not distracted. – P1

> "[The biggest benefit is] being focused on one tab at one given time... [When] you have so many tabs, keep juggling here and there, don't know what to do. I like to keep life simple, and I want to achieve what I'm doing at that point in time. – P4

6 DISCUSSION AND FUTURE WORK

In this current work, we explored a task-centric approach to tabbed browsing through a research prototype, Tabs.do. To enable this approach we introduced the tab bundle primitive, and task management affordances and views built on top of it. In an evaluation study we found that participants using the system found the approach useful, and identified changes in their behavior including decreased tab and window clutter, the creation and use of rich, nested task structures, and frequent context-switching among tasks.

Our results are promising in suggesting that a task-centric approach may be profitably employed in tabbed browsing interfaces. In doing so they are consistent with the beneficial use of activity-based computing approaches in other contexts including general task management [6], desktop applications and local files [2, 3, 54], and email [7], and there are strong parallels between these contexts and tabbed browsing (for example, the need for efficient context-switching, reminding and avoiding "black-hole" effects, or collecting lower-priority tasks that they do not expect to complete [6, 13]) that suggest that users might indeed be treating their tabs as elements of larger underlying tasks.

However, there are also interesting differences between tasks in the browser and in other contexts such as email or file systems that may suggest the need for different functionality going forward. One fundamental difference is that many online tasks are inherently exploratory [38], requiring users to proactively seek out and make sense of many different pieces of information [43, 46], not all of which are necessarily useful, and iteratively refine their goals [38]. To design for this fluid task structure, we introduced the concept of tab bundles that allowed users to structure and restructure their tasks to reflect their changing mental models when conducting tasks in the browser, as well as the ability to prioritize and triage collected information. However, further support for refactoring of tasks and managing multiple promising branches is likely an important area for future work.

Prior work in activity-based computing has pointed to benefits in providing users access to their tasks across multiple devices and applications [10, 18]. While the current implementation of Tabs.do synchronizes in-browser tasks across computers, extending it to support mobile devices and other desktop applications could be an interesting directions for future work. For example, task bundles in Tabs.do could potentially be used as basic building blocks to connect applications and devices to build a more holistic system [4, 5]. Such an approach could enable users to schedule and surface a to-read task bundle on their mobile phones during an upcoming commute or seamlessly bundle browser tabs with other local applications or files supporting the same tasks.

Some participants pointed to the possibility of seeing the task bundle suggestions directly on the browser interface without switching to the new tab page to see them in the Open Tabs view. One practical challenge we faced when exploring new browser interactions was that current Web Extension APIs have very limited support for changing the interfaces and interactions of browser tabs. For example, it would be difficult to change the structure or visual saliency of tabs on native UI (such as colors or widths) to surface our tab bundle suggestions with current browser APIs. Participants also pointed to limitations imposed by current Web Extension APIs. For example, Tabs.do used the "saved" badge on the extension icon to show that it was previously saved, but Web Extension APIs lack mechanisms for Tabs.do to further surface statuses or structures that the users assigned to their tabs, such as due dates or projects.

While Tabs.do's task-centric approach may provide a useful step forward in helping people with their online tasks, it represents only one piece of a richer tapestry of functionality that would be necessary to support the complex learning, decision-making, and sensemaking that people engage in on the internet. One way to think of this larger ecosystem might include Tabs.do as a hub for creating, organizing and managing tasks, but with additional functionality on each end. The need for saving clips, snippets, and annotations when exploring unfamiliar information is well documented [14, 32, 39, 49, 50], and supporting the collection of such information in users' tasks could be an important extension for our approach. On the other end, as users collect more information there is an increasing need for workspaces that can help them structure, compare, synthesize, and take action on it [15, 16, 19, 37, 42, 47]. Enabling Tabs.do to seamlessly pass information and synchronize

with specialized workspaces for different types of tasks could be a fruitful future direction.

Finally, a more prolonged deployment could reveal more insights into Tabs.do's longer-term costs and benefits, for example the scalability of the system as users accumulate more tasks over time or once any novelty effects have worn off. Early evidence on this question is promising: we continued to monitor the usage logs and found that 7 out of the 10 participants voluntarily continued to use Tabs.do daily for more than ten weeks after the study had concluded. A follow-up discussion with five of them revealed that they saved more items and continued to benefit from the system, and they emphasized that being able to collapse and expand their bundles allowed for efficient navigation within their projects. We are now building the next version of Tabs.do for a larger and longer deployment as a follow-up study.

7 CONCLUSION

This paper explored how using a task-centric approach in the browser can better support users in managing their browser tabs. Our designs were motivated by the growing evidence that current browser designs have become insufficient to support modern online tasks for a significant segment of users [13]. We introduced Tabs.do, a browser extension that instantiates this idea by allowing users to save their browser tabs as "tab bundles" and use a set of task-centric affordances to manage them. Using a deep learning model, Tabs.do minimizes the cognitive and interaction costs of creating tab bundles, lowering the adoption barrier to our task-centric approach. Through a week-long field deployment study with 10 participants using Tabs.do on their computers to manage their real-world tabs, we found evidence that our task-centric approach allowed users to manage their browser tabs more effectively. Specifically, Tabs.do enabled participants to efficiently context-switch among tasks, reduce tab clutter, and create task structures that better reflected their mental models. As online tasks become increasingly complex, new interfaces and interactions that can bridge the divide between *tab management* and *task management* in the browser may become increasingly important. Tabs.do represents a first step towards bringing task-centric approaches to browser tab management that have stayed relatively static for the past 20 years to better support users conducting complex online tasks today.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (PFI-1701005 and SHF-1814826), the Office of Naval Research, Google, and the Carnegie Mellon University Center for Knowledge Acceleration.

REFERENCES

[1] Eytan Adar, Jaime Teevan, and Susan T Dumais. 2008. Large scale analysis of web revisitation patterns. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1197–1206.

[2] Anand Agarawala. 2006. Enriching the desktop metaphor with physics, piles and the pen. In *Masters Abstracts International*, Vol. 45.

[3] Anand Agarawala and Ravin Balakrishnan. 2006. Keepin'it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 1283–1292.

[4] J. Bardram, Jonathan Bunde-Pedersen, and Mads Søgaard. 2006. Support for activity-based computing in a personal computing operating system. In *CHI*.

[5] J. Bardram, S. Jeuris, Paolo Tell, Steven Houben, and Stephen Voida. 2019. Activity-centric computing systems. *Commun. ACM* 62 (2019), 72 – 81.

[6] Victoria Bellotti, Brinda Dalal, Nathaniel Good, Peter Flynn, Daniel G Bobrow, and Nicolas Ducheneaut. 2004. What a to-do: studies of task management towards the design of a personal task list manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 735–742.

[7] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. 2003. Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 345–352.

[8] Michael Bernstein, Max Van Kleek, David Karger, and MC Schraefel. 2008. Information scraps: How and why information eludes our personal information management tools. *ACM Transactions on Information Systems (TOIS)* 26, 4 (2008), 1–46.

[9] Richard E Boyatzis. 1998. *Transforming qualitative information: Thematic analysis and code development*. sage publications, inc, Thousand Oaks, California, United States.

[10] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, C. Klokmose, and Nicolai Marquardt. 2019. Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019).

[11] Lara D Catledge and James E Pitkow. 1995. Characterizing Browsing Strategies in the World-wide Web. *Computer Networks and ISDN Systems* 27, 6 (1995), 1065–1073.

[12] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[13] Joseph Chee Chang, Nathan Hahn, Yongsung Kim, Julina Coupland, Bradley Breneisen, Hannah S Kim, John Hwong, and Aniket Kittur. 2021. When the Tab Comes Due: Challenges in the Cost Structure of Browser Tab Usage. In *Proceedings of the 2021 SIGCHI conference on Human factors in computing systems*.

[14] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2016. Supporting Mobile Sensemaking Through Intentionally Uncertain Highlighting. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. ACM, New York, NY, USA, 61–68. https://doi.org/10.1145/2984511.2984538

[15] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2020. Mesh: Scaffolding Comparison Tables for Online Decision Making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 391–405. https://doi.org/10.1145/3379337.3415865

[16] Joseph Chee Chang, Nathan Hahn, Adam Perer, and Aniket Kittur. 2019. SearchLens: Composing and capturing complex user interests for exploratory search. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. ACM, New York, NY, USA, 498–509.

[17] Kathy Charmaz and Linda Liska Belgrave. 2007. *Grounded Theory*. Wiley Online Library, Hoboken, New Jersey, United States.

[18] David Dearman and Jeffrey S. Pierce. 2008. It's on my other computer!: computing with multiple devices. In *CHI*.

[19] Mira Dontcheva, Steven M Drucker, David Salesin, and Michael F Cohen. 2007. Relations, cards, and search templates: user-guided web data integration and layout. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 61–70.

[20] Patrick Dubroy and Ravin Balakrishnan. 2010. A Study of Tabbed Browsing Among Mozilla Firefox Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. ACM, New York, NY, USA, 673–682. https://doi.org/10.1145/1753326.1753426

[21] Chrome Extension. 2020. OneTab. https://chrome.google.com/webstore/detail/onetab/chphlpgkkbolifaimnlloiipkdnihall. Accessed: 2020-09-10.

[22] Chrome Extension. 2020. Workona. http://workona.com/. Accessed: 2020-09-15.

[23] Chrome Extension. 2021. SessionBuddy. https://chrome.google.com/webstore/detail/session-buddy/edacconmaakjimmfgnblocblbcdcpbko. Accessed: 2021-01-07.

[24] Chrome Extension. 2021. Toby. https://chrome.google.com/webstore/detail/toby-for-chrome/hddnkoipeenegfoeaoibdmnaalmgkpip. Accessed: 2021-01-07.

[25] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. 2003. DOM-based content extraction of HTML documents. In *Proceedings of the 12th international conference on World Wide Web*. 207–214.

[26] Life Hacker. 2012. Master Your Browsers Tabs with These Tricks and Extensions. http://lifehacker.com/5883299/master-your-browsers-tabs-with-these-tricks-and-extensions. Accessed: 2017-09-10.

[27] Life Hacker. 2013. It's Okay to Open More Than Nine Browser Tabs; Here's How to Easily Manage Them. http://lifehacker.com/5985462/its-okay-to-open-more-than-nine-browser-tabs-you-just-need-to-manage-them-properly. Accessed: 2017-09-10.

[28] Life Hacker. 2013. Why You Should Never Have More Than Nine Browser Tabs Open. http://lifehacker.com/5984149/why-you-should-never-have-more-than-nine-browser-tabs-open. Accessed: 2017-09-10.

[29] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento Browser: Complex Mobile Search Without Tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–12.

[30] Mona Haraty, Diane Tam, Shathel Haddad, Joanna McGrenere, and Charlotte Tang. 2012. Individual differences in personal task management: a field study in an academic setting. In *Proceedings of Graphics Interface 2012*. 35–44.

[31] D Austin Henderson Jr and Stuart Card. 1986. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG)* 5, 3 (1986), 211–243.

[32] Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal Information Gathering Techniques for Active Reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1893–1896. https://doi.org/10.1145/2207676.2208327 event-place: Austin, Texas, USA.

[33] Jeff Huang, Thomas Lin, and Ryen W White. 2012. no search result left behind: branching behavior with browser tabs. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*. ACM, ACM, new york, ny, usa, 203–212.

[34] Jeff Huang and Ryen W white. 2010. Parallel Browsing Behavior on the Web. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*. ACM, ACM, new york, ny, usa, 13–18.

[35] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, ACM, New York, NY, USA, 699–708.

[36] Alex Krizhevsky and Geoff Hinton. 2010. Convolutional deep belief networks on cifar-10. *Unpublished manuscript* 40, 7 (2010), 1–9.

[37] Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A Myers. 2019. Unakite: Scaffolding Developers' Decision-Making Using the Web. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 67–80.

[38] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[39] Catherine C. Marshall and Sara Bly. 2005. Saving and Using Encountered Information: Implications for Electronic Periodicals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 111–120. https://doi.org/10.1145/1054972.1054989 event-place: Portland, Oregon, USA.

[40] Brian H Murray and Alvin Moore. 2000. Sizing the internet. *White paper, Cyveillance* 3 (2000).

[41] Hacker News. 2018. Open tabs are cognitive spaces (rybakov.com). https://news.ycombinator.com/item?id=16671957. Accessed: 2020-09-13.

[42] Sharoda A Paul and Meredith Ringel Morris. 2009. CoSense: enhancing sensemaking for collaborative web search. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1771–1780.

[43] Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological review* 106, 4 (1999), 643.

[44] Reddit. 2013. I have a serious problem with browser tab hoarding. https://www.reddit.com/r/declutter/comments/1jpw13/i_have_a_serious_problem_with_browser_tab_hoarding/. Accessed: 2017-09-10.

[45] Reddit. 2016. I'm a digital hoarder. I opened chrome to find all my tabs gone. I feel relieved. https://www.reddit.com/r/declutter/comments/4qkomc/im_a_digital_hoarder_i_opened_chrome_to_find_all/. Accessed: 2017-09-10.

[46] Daniel M Russell, Mark J Stefik, Peter Pirolli, and Stuart K Card. 1993. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 269–276.

[47] MC Schraefel, Max Wilson, Alistair Russell, and Daniel A Smith. 2006. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM* 49, 4 (2006), 47–49.

[48] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. 2019. Tensorflow.js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350* (2019).

[49] Craig S. Tashman and W. Keith Edwards. 2011. Active reading and its discontents: the situations, problems and ideas of readers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2927–2936. https://doi.org/10.1145/1978942.1979376

[50] Craig S. Tashman and W. Keith Edwards. 2011. LiquidText: A Flexible, Multitouch Environment to Support Active Reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 3285–3294. https://doi.org/10.1145/1978942.1979430 event-place: Vancouver, BC, Canada.

[51] Linda Tauscher and Saul Greenberg. 1997. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human-Computer Studies* 47, 1 (1997), 97–137.

[52] Jaime Teevan, Christine Alvarado, Mark S Ackerman, and David R Karger. 2004. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, USA, 415–422.

[53] Max G Van Kleek, Michael Bernstein, Katrina Panovich, Gregory G Vargas, David R Karger, and MC Schraefel. 2009. Note to self: examining personal information keeping in a lightweight note-taking tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1477–1480.

[54] Stephen Voida, Elizabeth D Mynatt, and W Keith Edwards. 2008. Re-framing the desktop interface around the activities of knowledge work. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 211–220.

[55] W3C, W3Counter. 2017. Web Browser Usage Trends. https://www.w3counter.com/trends. Accessed: 2017-09-10.

[56] Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. 2006. Off the Beaten Tracks: Exploring Three Aspects of Web Navigation. In *Proceedings of the 15th International Conference on World Wide Web*. ACM, ACM, New York, NY, USA, 133–142.

[57] Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. 2008. Not Quite the Average: An Empirical Study of Web Use. *ACM Transactions on the Web (TWEB)* 2, 1 (2008), 5.

[58] Steve Whittaker and Candace Sidner. 1996. Email Overload: Exploring Personal Information Management of Email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, ACM, 276–283.