Grounded Task Prioritization with Context-Aware Sequential Ranking

CHUXU ZHANG, Brandeis University
JULIA KISELEVA, SUJAY KUMAR JAUHAR, and RYEN W. WHITE, Microsoft Research

People rely on task management applications and digital assistants to capture and track their tasks, and help with executing them. The burden of organizing and scheduling time for tasks continues to reside with users of these systems, despite the high cognitive load associated with these activities. Users stand to benefit greatly from a task management system capable of prioritizing their pending tasks, thus saving them time and effort. In this article, we make three main contributions. First, we propose the problem of task prioritization, formulating it as a ranking over a user's pending tasks given a history of previous interactions with a task management system. Second, we perform an extensive analysis on the large-scale anonymized, de-identified logs of a popular task management application, deriving a dataset of grounded, real-world tasks from which to learn and evaluate our proposed system. We also identify patterns in how people record tasks as complete, which vary consistently with the nature of the task. Third, we propose a novel contextual deep learning solution capable of performing personalized task prioritization. In a battery of tests, we show that this approach outperforms several operational baselines and other sequential ranking models from previous work. Our findings have implications for understanding the ways people prioritize and manage tasks with digital tools, and in the design of support for users of task management applications.

CCS Concepts: • **Information systems** → **Information systems applications**; • **Computing methodologies** → **Artificial intelligence**; **Machine learning;**

Additional Key Words and Phrases: Tasks, Task prioritization, Contextual recommendation, Sequential ranking

ACM Reference format:
Chuxu Zhang, Julia Kiseleva, Sujay Kumar Jauhar, and Ryen W. White. 2021. Grounded Task Prioritization with Context-Aware Sequential Ranking. *ACM Trans. Inf. Syst.* 40, 4, Article 68 (December 2021), 28 pages.
https://doi.org/10.1145/3486861

1 INTRODUCTION

Task management applications such as Google Tasks, Microsoft To Do, and Todoist (todoist.com) enable their users to capture, organize, and manage tasks from their work and life [6]. In recent years, the research community has made several attempts to facilitate intelligent task assistance [70, 71] through work on contextual reminders [28, 39], task duration estimation [80], task completion detection [81], microtask identification [82], and complex task decomposition [88],

Chuxu Zhang's work was done during an internship at Microsoft Research.

Authors' addresses: C. Zhang, Brandeis University, Waltham, MA 02453; email: chuxuzhang@brandeis.edu; J. Kiseleva, S. K. Jauhar, and R. W. White, Microsoft Research, Redmond, WA 98052; emails: {jukisele, sjauhar, ryenw}@microsoft.com.

1046-8188/2021/12-ART68 $15.00
https://doi.org/10.1145/3486861

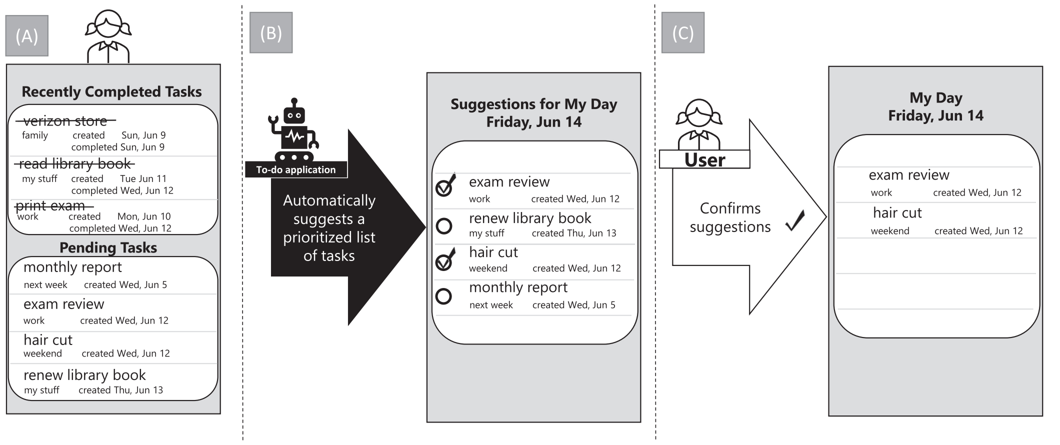

(A)
Recently Completed Tasks
~~verizon store~~ family created Sun, Jun 9 completed Sun, Jun 9
~~read library book~~ my stuff created Tue Jun 11 completed Wed, Jun 12
~~print exam~~ work created Mon, Jun 10 completed Wed, Jun 12
Pending Tasks
monthly report next week created Wed, Jun 5
exam review work created Wed, Jun 12
hair cut weekend created Wed, Jun 12
renew library book my stuff created Thu, Jun 13
(B)
To-do application
Automatically suggests a prioritized list of tasks
Suggestions for My Day Friday, Jun 14
exam review work created Wed, Jun 12
renew library book my stuff created Thu, Jun 13
hair cut weekend created Wed, Jun 12
monthly report next week created Wed, Jun 5
(C)
User
Confirms suggestions
My Day Friday, Jun 14
exam review work created Wed, Jun 12
hair cut weekend created Wed, Jun 12

Fig. 1. An example of suggested task prioritization in a task management application. (A) shows two lists: recently completed user tasks and pending tasks. (B) illustrates a re-ranked list of pending tasks from (A), prioritized automatically by the system. Finally, (C) represents the agency a user exercises in selecting tasks from the suggested priority list (B) for planning their day.

among others. One aspect of task intelligence that has remained under-explored is that of *prioritizing* a user's pending tasks according to factors such as their context, underlying intent, and the relationship between individual tasks. Urgency and importance (well-established aspects of task prioritization [55, 57]) may only be part of the reason why people elect to do tasks in a certain order. We explore this more, later in the article. An example of automatic task prioritization in support of users planning their day is depicted in Figure 1. Effective planning and prioritization of tasks is important. A recent study of R&D engineers showed that 27% of tasks assigned to to-do lists at the start of the workday are still incomplete at the end of the day [16]. Tackling tasks under time pressure has been shown to induce stress [54]. A failure to complete daily planned tasks increases work-related rumination during periods of downtime [75].

Users of task management applications often have backlogs of pending tasks that have not yet been completed (Figure 1(A): "Pending Tasks"). With current technology, such a backlog must be managed manually by users when organizing their schedules or planning their day. Meanwhile, given the limitations of human short-term memory [49], triaging and prioritizing tasks is a cognitively demanding endeavor [6], especially for users whose backlog contains many tasks. This is why busy executives still delegate calendar management to human assistants. Intelligent systems can help with scheduling of tasks of a known duration [20, 77], but task priority has been ignored.

Given the challenges that people face in managing tasks, methods that can assist them in generating a plan for their day, while taking into account their previous interactions (and other aspects such as context and goals, if available), could yield significant productivity benefits. Thus, the focus of this article is on algorithmic methods to provide assistance to users in their efforts to effectively manage and schedule their tasks. Specifically, this assistance is in the form of a model that automatically prioritizes users' pending tasks.

This is illustrated in Figure 1(B), where a re-ranking of pending tasks is presented to the user as an automatically prioritized list. In this scenario, the agency of final decision-making still resides with the user, who can select tasks from the prioritized list of pending tasks to add to their plan for the day (Figure 1(C)). These interactions, in turn, can be used as feedback to improve the system over time [46, 47].

Existing task ranking solutions rely on heuristic methods for prioritization. For example, Microsoft To Do ranks pending tasks based solely on the time of their creation. Simple heuristic methods, while reasonable first approaches, are sub-optimal since they ignore the broader context of tasks, including user intent, temporal attributes of tasks, and longer sequential interactions with the task management application. At the same time, existing methods for sequential ranking [34, 76] or contextual recommendation [33, 72] cannot simply be ported to rank pending tasks, since they must comprise modeling components that can meaningfully represent a task for grounded prioritization. Grounding in this case means using real-world tasks from which to learn and evaluate our proposed system.

Temporary, limited access to a sample collection of anonymized user logs from the now-defunct Wunderlist task management application allows us to tackle this problem in a grounded manner, at scale, for the first time. While these logs were accessible through opt-in by customers from the application's user agreement, we further scrubbed, de-identified, and anonymized them to mitigate privacy and ethical concerns. It may be noted, that with the shutdown of Wunderlist, the logs are now completely purged and no longer accessible other than in aggregate statistics. A detailed description of the privacy mitigation steps is provided in Section 4.1.

We begin by defining the problem of task prioritization in a manner that is conducive to practical integration with real-world to-do applications (the focus of the investigation in this article). Specifically, at an arbitrary point in time, the goal of task prioritization is to rank the tasks that a user has not yet completed in order of estimated priority. In order to further ground the problem in real data, we attempt to optimize the ranking order of task prioritization on the true temporal ordering in user logs, i.e., the sequence in which the user actually then went on to complete their pending tasks (Figure 1(A): "Recently Completed Tasks"). Given this general problem formulation, we attempt to extract reliable labels for the underlying optimization objective from the sample of real-world Wunderlist log data. A significant challenge in working with these logs is that the time a task was marked complete does not necessarily equal the time that the task was actually completed (although it does represent an upper bound for completion time). We tackle this challenge to obtain approximately reliable completion time labels by eliminating tasks that are either marked complete in bulk or belong to lists where the temporal ordering of completion events is not informative (e.g., groceries, shopping, and packing). Finally, as a first attempt at modeling the problem of task prioritization, we design a deep learning solution that performs task prioritization by aggregating both textual and contextual attributes of tasks, as well as the temporal dependencies between them. The model, once trained, can be used to predict a personalized ordering of pending tasks for previously unseen collections at any given point in time.

In summary, we make the following contributions in this article:

- **C1.** We propose the novel research challenge of pending task prioritization, which, to the best of our knowledge, is the first such attempt. Furthermore, we formalize the problem in a practical manner by grounding the objective of task prioritization in a large-scale, heterogeneous, and noisy data sample of logs containing anonymized and de-identified user interactions with the now-defunct Wunderlist task management application.
- **C2.** We identify patterns in how people mark tasks as complete in that application (i.e., there is evidence of batching behavior, where task completion labels are assigned in bulk, often only a few seconds apart), and strong differences in those patterns per the nature of the task (e.g., more batching for shopping and packing lists, less batching for work task- and finance-related lists).
- **C3.** We analyze aggregated Wunderlist statistics on time-to-task-completion and report on the relationship between priority/activity and completion time.

C4. We design a deep learning solution that effectively incorporates content, contextual attributes, and temporal dependencies between tasks into a ranker (*TaskRank*) for users' pending tasks.

C5. We demonstrate, through rigorous testing, that *TaskRank* outperforms several strong baselines, and that the different feature attributes and components of the neural network model all contribute positively to its efficacy.

The remainder of the article is structured as follows. Section 2 describes related work. Sections 3, 4, and 5 define the problem, the data, and the task prioritization model, respectively. We evaluate our proposed model in Section 6, and discuss the implications of our findings in Section 7.

2 RELATED WORK

Our research on automated prioritization of users' tasks is related to prior work on intelligent task management (Section 2.1), prioritization in general (Section 2.2), and contextual recommendation (Section 2.3). We now review related work in each of these areas, and conclude by highlighting our contributions over previous work (Section 2.4).

2.1 Intelligent Task Management

Research on task management typically focuses on prospective planning of to-dos (e.g., finding time for tasks, deciding what tasks to do when, deciding which tasks to do next) [3, 24]. Managing tasks manually is cognitively demanding and people may benefit from assistance [43]. Digital assistants (Amazon Alexa, Google Assistant, etc.) and task management applications (Google Tasks, Todoist, etc.) can help people enumerate and track tasks.

Task management assistance has been a topic of research for decades. In one of the earlier explorations, Bellotti et al. [6] examined personal strategies in task management. Selected tasks were influenced by a variety of factors, including location, time, social relationships, and routines. Studies have found that most people manage their tasks using general-purpose tools such as paper scraps and note-taking functionality in digital devices [6, 10, 12]. Others have focused on different aspects of task organization including managing emails [30], automatically extracting tasks from messages [9, 61, 63], calendar management [11, 68], smart alerting with the goal of minimizing context switching [36], offering the right information at the right time [13], and suggesting decomposition and planning to facilitate the burden of complex tasks [88]. Email is especially important given its popularity as a task management tool [7, 83]. Assistive technologies can extract tasks from email messages [9, 78], facilitating commitment tracking and scheduling, and help users triage tasks in email [25].

A considerable amount of system support for task management has also been developed. Gil and Ratnakar proposed one such system (*CALO*) based on mapping tasks onto pre-determined arguments. These arguments could then be used to interpret and then recommend tasks [27]. *Towel* [19] allowed users to manage to-do tasks directly via instant messaging, including delegating tasks to *CALO*. Refanidis and Alexiadis [67] attempted to incorporate tasks into a calendar. Their system, *SelfPlanner*, relies on user input information to schedule tasks alongside events in Google Calendar. These systems show forays into doing the planning work a human would normally do but depends heavily on reliable and structured human input [67]. The availability of large-scale anonymized interaction logs and recent notable advances in deep learning have enabled the development of methods that can use unstructured input to offer support such as task duration estimation [80] (including the automatic detection of microtasks [82]), task completion detection [81], and enhancing notifications to maximize completion [28].

Beyond task organization, intelligent systems can assist with other aspects of task completion. Myers et al. [62] designed a system to support repetitive or mechanical tasks, so that people could focus on those tasks that require human problem-solving skills. *TaskGenies* generated action plans decomposing tasks into small, actionable sub-tasks [48]. *ProactiveTasks* [5] suggests short tasks that extend beyond simple application notifications and into "review" interactions requiring human cognitive capacities. Stumpf et al. [73] and Kiseleva et al. [44] explore methods to understand people's task intent and provide the resources required to complete those tasks.

2.2 Prioritization

Prioritization has traditionally referred to the ability to arrange or deal with assigned responsibilities in order of urgency and/or importance [55, 57]. It is a central aspect of how people manage their tasks [6]. Prioritization involves a range of activities, including agenda setting, resource allocation, task switching, and interruptions [21, 65]. Several studies have examined prioritization during task execution, including focusing on specific job roles such as airline pilots [18] or healthcare workers [17]. Other studies have shown that people are more likely to choose easier and shorter tasks [23, 29] and are more likely to prioritize tasks subjectively deemed salient [64].

Systems have been developed to directly support prioritization. Horvitz et al. [36] developed *Priorities*, a prototype system to assign criticality scores to incoming email messages, enabling users to focus their attention on the most pressing items. Other research has focused on predicting the importance and relevance of meetings [15, 37]. Focusing directly on tasks, the *TaskVista* prototype [6] helped users prioritize tasks based on criteria such whether they are non-discretionary, participant importance, or the relationship to a deadline.

Studies of personal task management have also found that users have a preset tendency for how they organize their tasks [6, 32]. This includes either using general tools such as to-do lists or calendars, or adapting existing tools to suit their personal preferences [32]. Gan et al. [26] found that people who are more proactive in preparing for future events were (a) less likely to favor immediate rewards, and (b) more likely to demonstrate preferences for urgent but less important tasks. Arrington et al. [4] found that participants tended to favor task sequences that they had previously adopted when similar options were presented again. The seemingly significant role of personal preferences and tendencies in prioritization also creates opportunities for personalizing task management support [31].

2.3 Contextual Recommendation

Context-aware recommendation systems [33, 72, 79] aim to leverage contextual information to improve the quality of the services they offer. They can be characterized by two stages [14]: (1) defining context and (2) incorporating that context. Context integration can further be classified into three groups [2]: (1) pre-filtering, (2) post-filtering, and (3) contextual modeling. Various approaches have been proposed to facilitate contextual recommendations for real-world applications. These include news recommendations via contextual bandits [51] or time-varying bandits [87], context-aware point-of-interest recommendations [84, 86, 89], calendar-aware email recommendations [90], sequential item recommendations [34, 76, 85], contextual travel recommendations [45], and context-sensitive reminding [22, 39].

Focusing on tasks, contextual factors such as busyness have been shown to impact the effectiveness of the task assistance and the extent to which people depend on it [40, 50]. Recent research has sought to understand the physical locations (contexts) in which people perform certain types of tasks and activities [8]. By combining location with time, context-aware task management systems can assist users by suggesting the right tasks at the right time [41, 69].

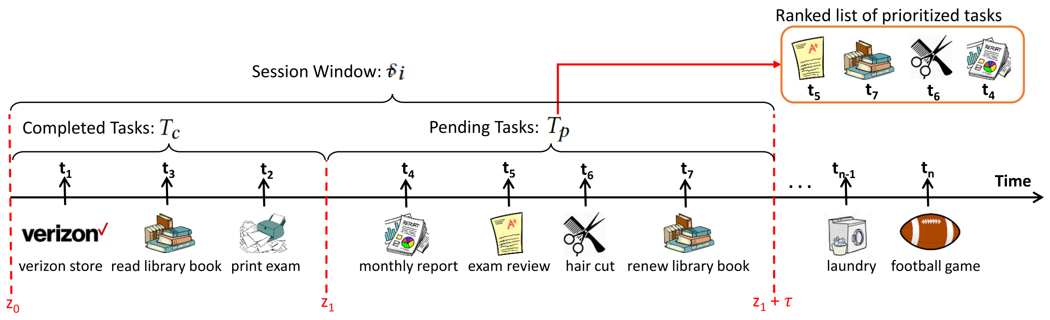


Fig. 2. Illustration of the session-based problem setting.

2.4 Contributions Over Previous Work

To our knowledge, we are the first to successfully propose a fully automated method for prioritizing a user's backlog of pending tasks. Previous research has tackled related issues, but has not focused directly on automatic task prioritization. *Priorities* [36] estimated the criticality of email messages and *Coordinate* [37] predicted the importance and relevance of meetings. *TaskVista* [6] relied on users specifying the importance of tasks, which only becomes a priority ordering when tasks are manually sorted by that property. Systems have leveraged contextual signals (e.g., time, location, calendar, and recent activity) to recommend or rank salient information [41, 51, 76] or minimize forgetting [22, 39], but task recommendation based on priority has been little explored. Much of the prioritization research has involved either building systems based on small-scale data [36, 37] or developing systems that require considerable human input [6, 32]. Uniquely, the deep model that we develop is grounded in and learned from real-world anonymized and de-identified user interactions across millions of tasks, and ranking is performed automatically, without requiring user input. Temporary access to this large-scale dataset enabled us to learn generalizable patterns and derive insights on how people manage their tasks lists in natural settings (e.g., batching together completion events). We show that our task prioritization model has strong offline performance compared to the heuristic methods (e.g., order by creation time) currently used for scheduling by popular task management applications, such as Microsoft To Do.

3 DEFINING THE TASK PRIORITIZATION PROBLEM

In this section, we formalize the problem of prioritizing a user's pending tasks. We consider the setting illustrated in Figure 1(A). A user u is interacting with a task management application by creating a task t, which is represented as a textual description, e.g., "exam review," "hair cut." They may then assign t to one of their personalized and previously-defined categories or *lists*, e.g., "work," "family," or "my stuff." Once t is done, u marks it as completed. The history of interactions of a particular user u with the task management application is denoted as a set of tasks $T = \{t_j\}_{j=1}^{n}$ which consist, at any given point of time z, of two distinct non overlapping sets: a sequence of completed tasks T_c, and a set of pending tasks T_p.

To model user interactions with the task management application we introduce the notion of session, which is illustrated in Figure 2. Each session $s_i \in S$ (the set of all sessions for each user) consists of two time windows defined as follows:

Completed Tasks: T_c is a history of user interactions represented by an *ordered* time sequence of tasks completed by u within an *arbitrary* time window $[z_0, z_1]$, such as (t_1, t_3, t_2) in Figure 2.

Pending Tasks: T_p is an *unordered* list of pending tasks created by u before z_1, that are completed within the *fixed* time period $[z_1 + \tau]$, such as (t_4, t_5, t_6, t_7) in Figure 2.

We further define the ordered set of task pairs in a session $\mathcal{P}_{s_i} = \{\langle t_+, t_- \rangle \mid \text{s.t. } t_+, t_- \in T_p; t_+ \neq t_-\}$, where we enforce that for any task pair $\langle t_+, t_- \rangle$, t_+ is marked complete by the user before t_-.

Given these inputs, we formulate the problem of *task prioritization* as follows:

> Finding a ranking T_p^* over T_p, such that the ordering of task pairs $\left\langle t_+^*, t_-^* \in \mathcal{P}_{s_i}^* \right\rangle$ in the predicted ranking are maximally consistent with the set of ground truth orderings $\mathcal{P}_{s_i}$.

Accordingly, the objective for this problem is formulated as:

$$\min_{\Theta} \sum_{s_i \in S} \sum_{\langle t_+, t_- \rangle \in \mathcal{P}_{s_i}} \mathcal{L}_{\Theta} \left(\langle t_+, t_- \rangle; T_p, T_c \right), \tag{1}$$

where $\mathcal{L}_{\Theta}(\langle t_+, t_- \rangle; T_p, T_c)$ denotes a pairwise ranking loss, and Θ is the set of parameters of the model, which will be elaborated in Section 5.

Next, we will describe task data used for modeling to accommodate the defined problem formulation (Section 5) and experimentation (Section 6).

4 TASK DATA

In this section, we describe the log data used in this work. We begin with explaining the privacy mitigation steps we took to ensure ethical usage of the data (Section 4.1). We then describe the sample of Wunderlist logs used in this investigation (Section 4.2) and describe the methods we applied to reduce inherent noise and extract approximately reliable labels from user interactions (Section 4.3).

4.1 Privacy and Ethical Usage of Wunderlist Data

We obtained a temporary and limited sample of data from the logs of the now-defunct[1] Wunderlist application. These logs were accessible through users' explicit opt-in agreement with the application's terms of use. Following the Wunderlist shutdown, the logs are no longer accessible and have been permanently purged. We did retain non user-level aggregate statistics of the Wunderlist data that helped us perform the limited analysis described in Section 4.4.

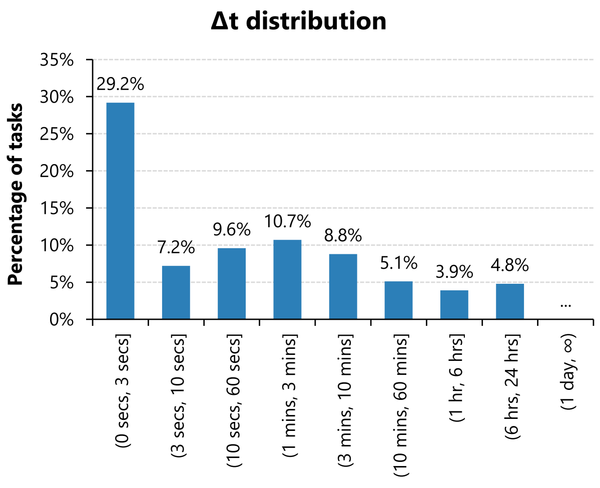
Despite explicit user opt-in, we appreciate the private and sensitive nature of the task data stored in these logs. We therefore passed the logs through a rigorous pipeline to scrub, de-identify, and anonymize the Wunderlist data. Specifically, we removed all personally identifiable information in textual data by replacing proper names and numbers with randomly-selected alternatives, using an enterprise grade, legal- and trust-approved pipeline. Furthermore, the only user identifier we retained was a GUID that was not tied back to any other user-level data or information.

Finally, rather than work with the full history of logs, we only experimented on a randomly sampled subset of logs between January 2016 and April 2019. To comply with **General Data Protection Regulation** (**GDPR**) requirements,[2] we periodically purged and then re-sampled the log data using the scrubbing and de-identification pipeline.

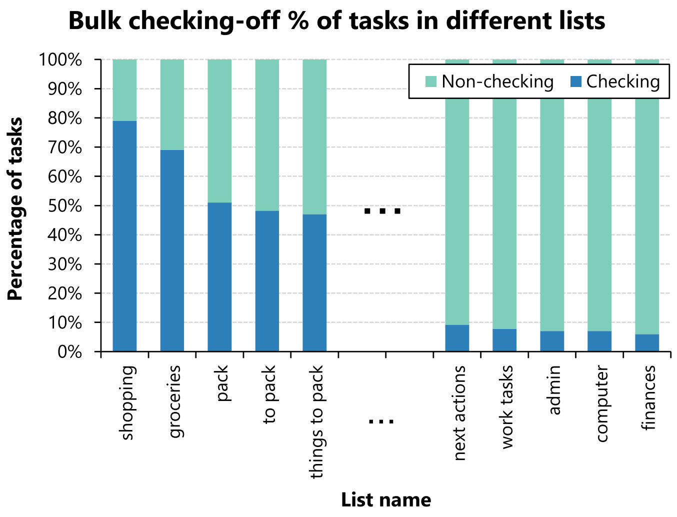
4.2 Sampled Data

Given the scrubbed, de-identified, and anonymized version of our data, we look to sample logs from *active* users. We empirically set the criteria for an active user by analyzing the data and truncating in order to filter out the tail of the distribution. Specifically, in this article, *active* users are defined as those that have used the task management application for more than 30 days and have created at least 20 tasks. For each of these users, we obtain a collection of tasks along with

[1]https://twitter.com/Wunderlist/status/1258034474274750466.
[2]https://gdpr-info.eu/.



(a) Δt distribution over all tasks



(b) Percentages of checking tasks of different lists (Δt < 3s).

Fig. 3. Analysis of bulk checking-off behavior.

several task specific attributes; notably these include the title of the task ("verizon store," "hair cut," "read library book," etc.), the list it belonged to (e.g., "groceries" and "work"), as well as the creation and completion times of the task. For consistency, we eliminate logged tasks that were missing any of these key attributes. Note that we only require completion times for pending tasks in order to evaluate the model in this log-based study. In general, our approach is agnostic to the completion times for pending tasks, and can predict a ranking over them regardless. In the end, our dataset contained 321M individual tasks from 1.1M users.

4.3 Extracting Approximately Reliable Completion Labels

Recall that, as per our problem definition, we use the completion time of a task as the ground truth for training and evaluating a ranking order over pending tasks. We only had access to the marked completion time in the Wunderlist dataset. There were no signals in the data indicating that a task was in progress and hence no way to differentiate between users working on a task and finishing a task. This may be important for more complex, longer-term tasks such as writing a report, where users may prioritize it highly and make progress on the task incrementally over time, but in the data we only observe a single task completion event and no evidence of task progress. Unfortunately, in practice, the completion time of a task, as recorded in the logs, is not necessarily the real time at which the task was completed—only when the user *marked* that task as completed. For example, a user may take stock of their day before going to bed and mark all the tasks that they completed during the day all at once. Furthermore, for certain kinds of tasks, the ordering of the completion times may not be meaningful: for example, when marking tasks on a grocery list as completed it does not matter whether eggs were bought before cheese, or vice versa. To account for these issues with the data, we designed two data-driven strategies to derive approximately reliable completion labels.

Strategy 1. The first strategy targets the user behavior of marking tasks complete in bulk. Let Δt denote the difference in completion time between two consecutively completed tasks. Tasks that are checked-off in bulk will correspond to small values of Δt, whereas those checked-off individually will have larger values. Figure 3(a) shows a histogram over values of Δt. From the plot it is evident that almost 30% of tasks having $\Delta t < 3$ seconds, which clearly indicates that bulk checking-off behavior is prevalent in the data. We use the three-second threshold in the remainder of this article, to help eliminate such rapid-fire completions (and associated tasks) from our dataset.

Strategy 2. The second strategy is designed to find instances where ordering of individual tasks is not meaningful, for example, in lists that represent collections, such as groceries, or packing. As with bulk checking-off behavior, tasks in such lists are often marked complete one after the other in quick succession (although not necessarily always within the set threshold of 3 seconds). Figure 3(b) illustrates this point by plotting lists ranked by the percentage of their tasks that are marked complete in bulk (i.e., $\Delta t < 3$ seconds). Based on our empirical analysis, the lists with the highest percentage of bulk checked-off tasks are shopping and packing related lists, whereas those with the lowest percentages are work- and finance-related lists. In this initial study, we are interested in the latter kind of list, because the distinctness of individual tasks (and by consequence their ordering) is important to our learning objective. Thus, we eliminate all tasks that belong to the former kind of list. To account for lexical variation in the naming of lists (e.g., "pack", "to pack", "things to pack" have the same semantic intent), we performed clustering on list titles. Specifically, we constructed a list-task frequency co-occurrence matrix for the top 2,000 most frequent lists and top 10,000 most frequent tasks across all users. We performed repeated bisection clustering with CLUTO[3] on this co-occurrence matrix, using a cosine similarity on the vector representation of lists and obtain k clusters (in this work k is set to 10). Table 1 summarizes the results of this clustering with a set of representative list names for each cluster. According to the table, Isim (average internal cluster similarity) is much larger than Esim (average external cluster similarity) for each cluster, demonstrating the generally high quality of the obtained clusters. From this clustering result, we selected clusters 0 and 8 as being representative of non-collection type lists and remove tasks belonging to lists in any of the other clusters.

After this filtering process, we were left with a dataset of approximately reliable completion time labels for tasks. Of course, there is no guarantee of truly perfect labels—for example, a user may have simply forgotten to mark a task complete and only checked it off later—but our filtering strategies cover the most egregious sources of noise in the data. Moreover, our prediction problem

[3]http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview.

Table 1. List Clustering Result (Isim and Esim[3] Denotes Average Internal and External Cluster Similarity, Respectively)

ID	Isim	Esim	A set of representative list titles
0	0.32	0.04	next action, projects, short-term, today list, saturday to do, weekend projects, to do today, today tasks
1	0.29	0.06	vacation packing, to bring, trip, baby list, camping pack list, my packing list, packing for trip, what to pack
2	0.25	0.02	gifts, christmas gifts, presents, birthdays, christmas ideas, thank you notes, christmas shopping, xmas gifts
3	0.29	0.06	home items, new home, house, furniture, buy for house, apartment things, apartment shopping, house repairs
4	0.20	0.04	dinner menu, meal ideas, cooking, lunch, dinner options, meal plans, dinner for the week, meal prep
5	0.25	0.11	amazon list, walmart, costco, buy now, non-grocery shopping, shopping walmart, target/amazon, target/costco
6	0.23	0.09	grocery, shopping list, to buy, costco, grocery store/target, shopping food, food items, food & groceries
7	0.11	0.01	compras, trgovina, zakupy, handleliste, dagligvarer, boodschappenlijst, supermercado, depot compras
8	0.12	0.03	school, finances, work projects, computer, assignments, work to do, work tasks, top priority
9	0.18	0.10	food, pharmacy, food shop, buy groceries, grocery list, supermarket list, foodstuffs, regular grocery

does not rely on exact task completion time anyway, but rather only on their relative ordering, and is therefore tolerant to any remaining noise.

While these filtering strategies do eliminate a non-negligible portion of the original dataset, our goal was to obtain a relatively clean dataset in which our approach to task prioritization can be grounded. Thus for this initial foray, we are left with a still sizeable dataset of 17M tasks from 500k users that focus on actionable tasks with approximately reliable completion times.

4.4 Analyzing Time to Task Completion

The completion labels in this article depend on *when* people do tasks. There may be many, often latent, factors that determine the order in which tasks get done. We explore two potential factors in this section: (i) the priority of the task (i.e., is it urgent and/or important, as defined by third-party judgments on the task title), and (ii) the activity being performed in the task (defined by the verb in the initial position in the task title and the associated suffix describing the specifics of the task). We selected these two factors based on the availability of the labeled data needed to compute them (priority) or whether they could be computed using automated methods (activity).

To perform this analysis, we used aggregate statistics from the Wunderlist dataset described earlier to calculate bucketed task completion time distributions.[4] As with the other analysis in this article, task completion time was the time until the task was marked as complete by the user. It is an upper bound of actual completion time, since all that we know is that the user completed the task at some point before marking it complete in the Wunderlist application. In computing the statistics, task completion times were grouped into five buckets over Δt: (i) >0 seconds and ≤1 hour, (ii) >1 hour and ≤1 day, (iii) >1 day and ≤1 week, (iv) >1 week and ≤2 weeks, and (v) >2 weeks. The aggregate statistics contained a frequency count of at least five for each <task title, time bucket >pair.[5]

Effects of Priority (Urgency and Importance). We might expect priority to be a significant contributor to task completion ordering, with people completing higher priority tasks before lower priority tasks. To investigate this, we used labeled data from another study, where tasks were labeled for priority by third-party judges (volunteer employees in our organization), and joined those labeled tasks with the aggregate statistics from the aggregate Wunderlist data. Judges labeled a random sample of 1,100 tasks in the Wunderlist dataset as either high priority (e.g., “get passport photos,”

[4]These statistics were extracted to facilitate limited task-level analysis after the original raw data were purged. They were generated using k-anonymization, with $k = 5$, meaning that tasks had to be created by at least five users to be included. The statistics are proprietary and cannot be released externally.
[5]Note the statistics contained no data at the individual user level that would let us examine the sequence ordering of tasks on a per-user basis.

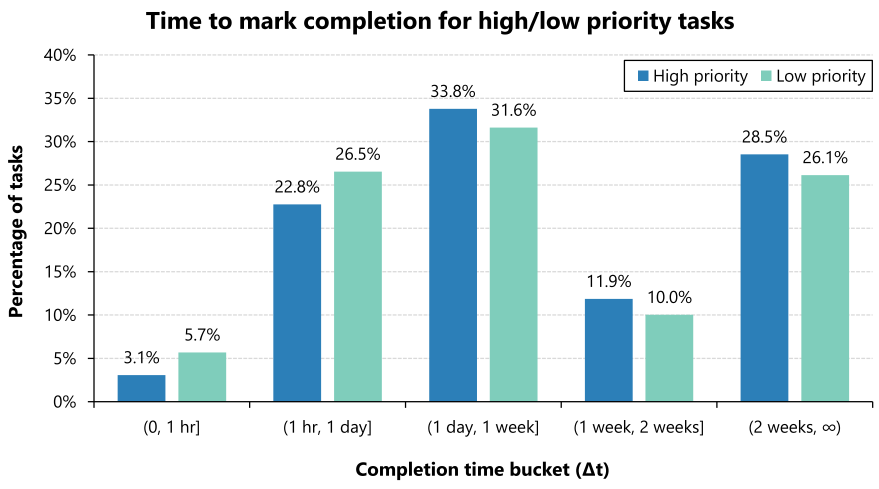


Fig. 4. Distribution of task completion times from the aggregate statistics ($\Delta t > 0$) for high- and low-priority tasks.

"pay credit card") or low priority (e.g., "organize closet," "mow lawn"). Judges were instructed that priority in this case referred to the urgency and/or importance the task and they were asked to estimate that from the task title alone. One of the authors of this article also labeled all grocery-related tasks (e.g., "3 min noodles," "garbage bags") in this labeled dataset and excluded them from the present analysis, as grocery items are also removed in later analyses. 42.7% of the tasks were labeled as grocery items and were thus excluded. Three judges labeled the priority of each task independently. The Krippendorff's Alpha inter-rater reliability metric was fair to good ($\alpha = 0.667$). To improve label robustness for our analysis, we took non-grocery tasks labeled as high priority by all three judges as high-priority examples ($n = 218$) and all other non-grocery tasks as low-priority examples ($n = 406$).[6] We filtered the Wunderlist aggregate dataset to the subset intersecting with the high- and low-priority tasks lists and computed the frequency distributions across Δt.

Figure 4 shows the frequency distribution of the completion times, broken out by high and low priority. If priority were the sole explanation for when tasks get done, we would expect the high priority distribution to be left skewed (completed quickly) and the low priority distribution to be right skewed (completed slowly). The difference between the frequency distributions is statistically significant ($\chi^2(4) = 586.98$, $p < 0.001$) but does not follow the expected pattern. Instead, Figure 4 shows that low priority tasks are being completed more quickly than high priority tasks, suggesting that other factors contribute to task completion order. Inspecting the tasks in each category, it appears that the high-priority tasks are often complex and have external dependencies (e.g., "mom hospital," "fix car tires"), while the low priority tasks are often simpler, with fewer dependencies (e.g., "break up boxes," "order door hangers") and are hence more conducive to being done quickly and/or opportunistically. This has also been reported in other studies [23, 29].

Effects of Activity. One potential additional factor that we could explore statistically was the high-level activity associated with a task. To do this, we ran the task titles though the Natural Language Toolkit **part of speech** (**POS**) tagger[7] and identified tasks with a base form verb ("VB"

[6]Note that the results do not change significantly if we instead regard tasks with two or more positive labels as high-priority examples.
[7]https://www.nltk.org.

Table 2. Top 20 Verbs for Activity Tasks (and their Relative Percentages) with Completion times within Each of the Five Time Buckets ($A_{\Delta t}$)

Rank	Time bucket (Δt) (0, 1 hr]	(1 hr, 1 day]	(1 day, 1 week]	(1 week, 2 weeks]	(2 weeks, ∞)
1	make (13.15%)	get (12.60%)	buy (13.49%)	buy (14.85%)	get (14.57%)
2	get (9.80%)	make (12.28%)	get (13.08%)	get (13.81%)	buy (13.88%)
3	buy (9.26%)	buy (11.14%)	pick (12.83%)	make (11.38%)	make (10.38%)
4	take (8.57%)	pick (10.88%)	make (11.52%)	pick (10.14%)	go (5.36%)
5	go (7.72%)	take (8.43%)	take (5.96%)	go (5.37%)	pick (4.34%)
6	do (6.32%)	go (6.99%)	go (5.79%)	take (5.04%)	take (4.05%)
7	pick (5.12%)	do (4.65%)	do (4.73%)	do (4.49%)	clean (3.85%)
8	clean (2.89%)	call (3.20%)	call (3.19%)	clean (3.34%)	do (3.80%)
9	put (2.59%)	put (3.08%)	clean (2.70%)	call (2.91%)	find (3.53%)
10	**run (2.27%)**	clean (2.85%)	put (2.63%)	put (2.85%)	organize (3.08%)
11	call (1.94%)	**run (2.69%)**	send (2.02%)	find (2.29%)	call (2.75%)
12	check (1.55%)	follow (1.83%)	find (1.89%)	send (1.90%)	put (2.62%)
13	follow (1.47%)	send (1.76%)	**run (1.75%)**	organize (1.83%)	replace (2.38%)
14	send (1.34%)	check (1.57%)	follow (1.74%)	follow (1.67%)	check (1.85%)
15	add (1.22%)	find (1.47%)	check (1.59%)	pay (1.57%)	follow (1.78%)
16	organize (1.14%)	organize (0.89%)	organize (1.17%)	check (1.51%)	sell (1.45%)
17	pay (1.14%)	pay (0.80%)	pay (1.14%)	**run (1.28%)**	send (1.43%)
18	find (1.05%)	read (0.68%)	read (1.06%)	replace (1.05%)	pay (1.29%)
19	be (0.99%)	turn (0.63%)	replace (0.72%)	read (1.01%)	read (1.13%)
20	wake (0.94%)	add (0.60%)	add (0.57%)	sell (0.58%)	**run (0.92%)**
% $A_{\Delta t}$	80.47%	88.99%	89.58%	88.87%	84.42%

The tasks referenced in the text are highlighted with bold ("run*,") and underline ("find*" and "organize*"). Also shown in the last row is the percentage of all activity tasks in $A_{\Delta t}$ that are comprised by the top 20 verbs listed in the table.

tag) e.g., "get," "buy," "call," and so on as the first token (0.73% of all tasks, 4.54% of all unique tasks). We refer to these tasks as *activity tasks*, and denote the set of all such tasks as A.

Table 2 lists the 20 most common verbs and their respective frequencies within all activity tasks with verbs within each of the five time buckets. As shown in the last row of Table 2, within each bucket, the top 20 verbs account for 80–90% of the activity tasks. From the table, we observe that there are some types of tasks ("get*", "buy*", "make*", "take*") which are consistently popular across all of the five different time horizons. Some of these (such as "get*", "make*", "take*") are light verbs, which contain little semantic information by themselves, but rather depend on their contextual noun-phrases to give them meaning—and could thus represent a wide range of tasks. We do, in fact, see such differences for these popular activities; for example, "get*" tasks that are marked as done more quickly are generally simpler: "get cash from atm" at ≤1 hour versus "get enhanced driver's license" at >2 weeks.

Activity tasks starting with "be," "wake," "add," and "turn" appear in the top 20 for tasks completed within a week, but do not appear thereafter. These tasks are replaced in the top 20 ranking by tasks with other activities such as "sell" and "replace." Activity tasks that appear in all five of the time buckets can also change in relative frequency depending on Δt. For example, we observe that less defined tasks (e.g., "organize*," underlined in Table 2) or tasks with uncertain outcomes (e.g., "find*," also underlined in Table 2) become relatively more frequent with longer time durations. Other activities (e.g., "run*," bolded in Table 2) are relatively more frequent with shorter time durations and decrease in relative frequency with duration. Examining the full task text for these "run*" tasks, we also observe differences in the full task titles, e.g., simpler, more tactical tasks such as "run 5 km," and "run laundry" at ≤1 hour versus more complex, more aspirational tasks such as "run ethernet to shed" and "run a marathon" at >2 weeks. Overall, these variations in activities and tasks with time show a clear relationship between which things get done and when they get done, and provides additional evidence that priority alone is insufficient to forecast task completion order.

Summary. The results show that both priority and activity can impact time to task completion. However, there are other factors, such as task feasibility (task may only be doable in certain settings, e.g., in office), temporal and social dependencies, and task complexity, as well as cognitive factors such as enjoyability [60] and locus of control [38], which may affect how quickly people complete their pending tasks. This is clearly a complex issue. We will never fully understand why users complete tasks in the order they do from analyzing retrospective task completion data alone. To do so, we need to also engage with users directly via methods such as surveys and interviews. We were unable to do this given the anonymized nature of the data used in this study. However, by considering the order in which tasks were completed, we bypass the need to focus on any single explanation and learn patterns from the data directly that help us account for some of these latent factors during task prioritization without the need to explicitly model them.

5 THE TASKRANK MODEL

In this section, we present details of our proposed model for pending task prioritization (*TaskRank*), which is illustrated in Figure 5. The model consists of four main steps:

(1) Featurizing task attributes (Section 5.1);
(2) Aggregating task features (Section 5.2);
(3) Modeling task dependencies (Section 5.3);
(4) Making a formal inference (Section 5.4).

5.1 Task Attributes

As described in Section 4, our log data associates a number of attributes to each task. In this article, we focus on two categories of task attributes. The first are **content attributes**, or broadly, textual attributes associated with each instance of a task. These include the:

(1) Title describing the task (e.g., "monthly report", "exam review"), and
(2) Title of the list it belongs to (e.g., "next week", "history class").

In addition, we also featurize **contextual attributes** associated with the task. In particular, we focus on computing two time-based attributes, namely:

(1) Longevity of a task, and
(2) Time since last task creation.

Here, *longevity* is interpreted slightly differently depending on whether a task is pending or has been completed at a given point in time. For pending tasks, longevity is the time since it was created, whereas for completed tasks, it is the time that elapsed between creation and completion. Meanwhile, *time since last task creation* for a given task is the time that elapsed since the user created their previous task.

In addition to these task attributes, we experimented with a number of additional features. These included the actual hour of day and day of week of task creation and completion, the average longevity of tasks, whether tasks were repeating/cyclical, and (if they were) their average periodicity. Unfortunately, in preliminary experiments none of these features yielded significant predictive

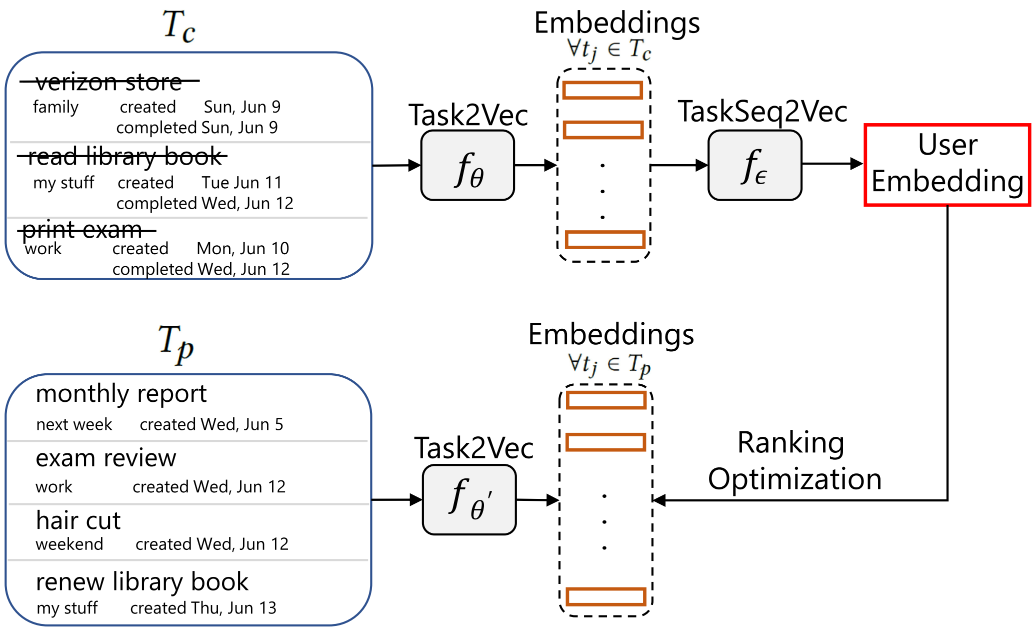


Fig. 5. Illustration of the overall *TaskRank* framework.

improvements to our model. As a result, in the rest of the article, we work with and report results on the four task attributes described above.

5.2 Task Features Aggregation (Task2Vec)

The language used to describe tasks and lists is domain specific, containing short textual elements that are often sloppily typed. Moreover, these elements are paired with numeric valued contextual metadata that we need to integrate into model featurization. Therefore, rather than using a set of off-the shelf embeddings (such as, for example, GloVe [66] or Word2Vec [59]) paired with a recurrent encoding layer, we take inspiration from character based recurrent neural networks [58, 74] to design a module that aggregates task attributes and learns its parameters from data. This module is called Task2Vec and it has two components, one each to deal with the content and contextual attributes of tasks as illustrated by Figure 6.

Content Features Encoding. The content attributes extracted from task data are short textual fragments. To encode these fragments we use a character RNN (C-RNN) augmented with a word attention mechanism. Formally, let t be a task whose content features we wish to encode; without loss of generality, this fragment can be the title of a task, or the list to which it belongs. Then denote $CO_t = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{w_{\max}}\}$ as the sequence of words in the text fragment associate with task t (after lowercasing and eliminating punctuation), where $w_{\max}$ is the pre-defined maximum length of any fragment. Furthermore, let us define the word $\mathbf{w}_i$ as being a sequence of character vectors $\mathbf{w}_i = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{c_{\max}}\}$, where $\mathbf{c}_j \in \mathbb{R}^{26\times 1}$ is a one-hot vector of jth character, and $c_{\max}$ is the pre-defined maximum length of any word in characters. Given this representation, we encode each $\mathbf{w}_i$ in the text fragment by passing it through a C-RNN layer. Specifically, we compute the jth hidden state $\mathbf{h}_j$ of the sequence, given the input $\mathbf{c}_j$ and previous hidden state $\mathbf{h}_{j-1}$, by:

$$\mathbf{h}_j = \mathrm{LSTM}(\mathbf{c}_j, \mathbf{h}_{j-1}), \tag{2}$$

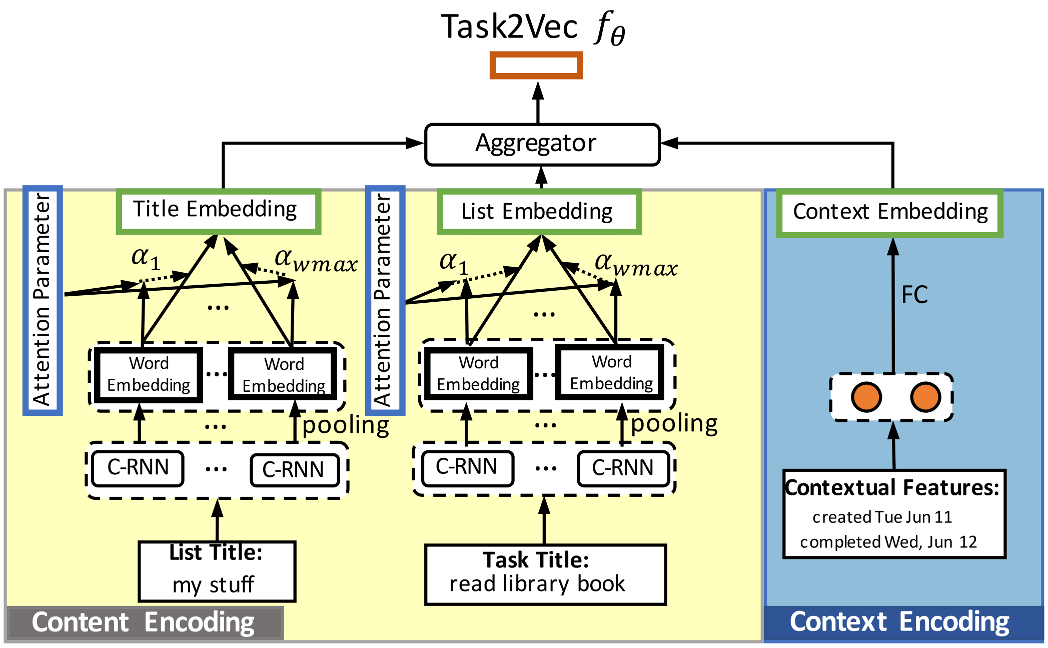


Fig. 6. Illustration of the Task2Vec module for aggregating task attributes.

where the LSTM cell is defined as suggested in [35]. We then apply a mean pooling layer[8] over all hidden states of the character sequence to obtain the word embedding $\mathbf{e}_{w_i} \in \mathbb{R}^{d\times 1}$:

$$\mathbf{e}_{w_i} = \mathbf{Mean}(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{c_{\max}}). \tag{3}$$

Given that different words in the titles of tasks or lists have different contributions to their semantic content, we further introduce a word attention layer to account for this impact. Specifically, we define:

$$\begin{gathered} g(CO_t) = \sigma\left(\sum_i \alpha_i \mathbf{e}_{w_i}\right), \text{ where} \\ \alpha_i = \frac{exp\left\{\mathbf{u}_{co}^T\left(\mathcal{W}_{co}\mathbf{e}_{w_i} + \mathbf{b}_{co}\right)\right\}}{\sum_j exp\left\{\mathbf{u}_{co}^T\left(\mathcal{W}_{co}\mathbf{e}_{w_i} + \mathbf{b}_{co}\right)\right\}} \end{gathered}. \tag{4}$$

Here $\mathbf{u}_{co} \in \mathbb{R}^{d\times 1}$, $\mathcal{W}_{co} \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_{co} \in \mathbb{R}^{d\times 1}$ are trainable parameters of the model. While we use the same network structure to encode the title of tasks and lists, we instantiate a different set of parameters for each attribute type to distinguish between the two.

Contextual Feature Encoding. Recall that the time-based contextual features computed from our dataset are numerical values. Rather than work with the raw numbers directly, we allow our model to scale and find correlations between the two contextual features by encoding them with a non-linear transformation. Concretely, we denote the set of contextual features of task t as CX_t

[8]We also experimented with max pooling and taking the last hidden state of the sequence, but found a loss in performance with these approaches.

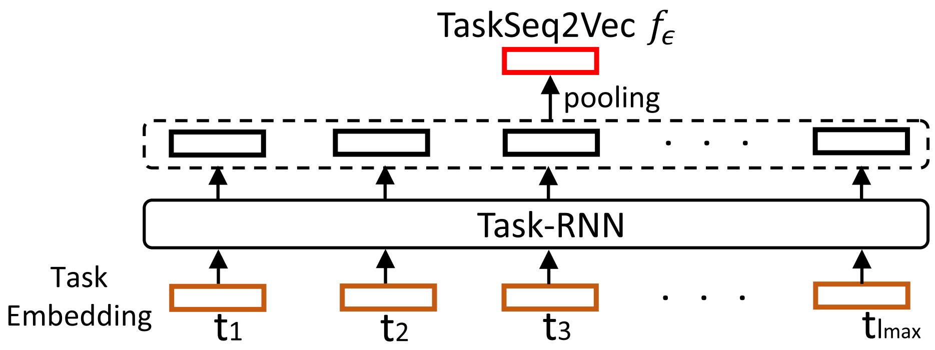


Fig. 7. Illustration of the TaskSeq2Vec module for encoding task dependencies.

and pass the resulting concatenated contextual feature vector (in this instance a pair of numbers) through a non-linear transformation:

$$g(\mathcal{CX}_t) = \sigma\{\mathcal{W}_{cx}\mathcal{CX}_t + b_{cx}\}. \tag{5}$$

Here σ is the non-linear activation unit (we use ReLU), and $\mathcal{W}_{cx} \in \mathbb{R}^{d\times|\mathcal{CX}_t|}$ and $b_{cx} \in \mathbb{R}^{d\times 1}$ are trainable parameters of the model.

Feature Aggregation. Finally, we combine the encoding of the content and contextual attributes using an aggregation layer. Specifically, given the encoding of the task title $g(\mathcal{CO}_t^{task})$, the list title $g(\mathcal{CO}_t^{list})$, and the contextual attributes $g(\mathcal{CX}_t)$, we compute an aggregated representation:

$$f_\theta(t) = \mathcal{AG}\left\{g(\mathcal{CX}_t), g\left(\mathcal{CO}_t^{title}\right), g\left(\mathcal{CO}_t^{list}\right)\right\}, \tag{6}$$

where θ denotes the set of all trainable parameters of Task2Vec, and $\mathcal{AG}$ is an aggregation function which can be, say, a pooling operator or a fully-connected layer. In this article, we set $\mathcal{AG}$ to a mean pooling operator since it yielded the best empirical results in our experiments.

5.3 Encoding Task Dependencies (TaskSeq2Vec)

Recent work in sequential item recommendation [34, 76] has demonstrated a benefit in modeling the dependency of consecutive items in the underlying problem. Therefore, we attempt to model the sequential nature of task prioritization by composing the representation of individual tasks computed by Task2Vec, which is further used to generate a user representation. Specifically, given an embedding sequence of a user's previously completed tasks, we leverage an RNN model to compute task hidden state embeddings:

$$\mathcal{H}_i = \textbf{LSTM}(f_\theta(t_i), \mathcal{H}_{i-1}), \tag{7}$$

where $f_\theta(t_i)$ and $\mathcal{H}_i \in \mathbb{R}^{d\times 1}$ are the input and hidden state at step i (i.e., task t_i), respectively. Then we apply a MP[9] over all hidden states H_i to obtain the user embedding:

$$f_\epsilon(u) = \textbf{Mean}(\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{l_{\max}}), \tag{8}$$

where $l_{\max}$ is the maximum sequence length, ϵ denotes all the trainable parameters of this module (TaskSeq2Vec). We name this module as TaskSeq2Vec and Figure 7 gives the illustration.

[9]We also experimented with max pooling and taking the last hidden state of the sequence, but did not find significant differences in performance.

5.4 Model Training and Inference

Given the user embedding $f_\epsilon(u)$, we now show how to train *TaskRank* and apply it to new unseen instances at test time.

Loss function and Training. Recall that in Section 3 we defined the notion of a session $s_i \in S$ (with S being the set of all sessions over all users in the training data) and $\langle t_+, t_- \rangle \in \mathcal{P}_{s_i}$ as the set of task pairs in the session, ordered by completion time. We then define the loss as:

$$\mathcal{L}_{TaskRank} = \sum_{s_i \in S} \sum_{\langle t_+, t_- \rangle \in \mathcal{P}_{s_i}} \max \left\{ \xi + r^u_{t_-} - r^u_{t_+}, 0 \right\}, \tag{9}$$

where ξ is the margin of the hinge loss, and $r^u_{t_+} = f_\epsilon(u)^\top f_\theta(t_+)$[10] denotes the similarity score between the user embedding and a pending task. We minimize the loss $\mathcal{L}_{TaskRank}$ using a batch sampling training procedure over tasks in each session s_i. The loss is optimized using Adam [42].

Test time Inference. Once training is complete, the model can be used to infer a ranking over pending tasks in new sessions without re-training or fine-tuning. Specifically, we can calculate a preference score for a user u and each of their pending tasks t by computing:

$$r^u_t = f^*_\epsilon(u)^\top f^*_\theta(t). \tag{10}$$

We can then rank pending tasks in descending order of their preference score r^u_t. Here f^*_ϵ and f^*_θ denote the optimized TaskSeq2Vec and Task2Vec modules, respectively.

Note that we follow the standard setting of sequential recommendation systems [34, 76], once we infer the best tasks in next session (e.g., next day), we use previous completed tasks to update the user state, and then move on to the next session, and so on and so forth.

6 EXPERIMENTS

We conduct experiments on ranking pending tasks in unseen sessions and evaluate *TaskRank* against several baselines and previous approaches to sequential ranking in the literature. The results demonstrate that our model outperforms the other approaches across experimental settings and on several different metrics. We then perform ablation studies that show that each of the task attributes we featurize, as well as the model components we develop, contribute positively to the overall performance of the model. We begin by providing details of our implementation and outlining baseline methods in Section 6.1. We then present results of our experiments in Section 6.2.

6.1 Experimental Methodology

Implementation Details. We implement our model in PyTorch.[11] For our experiments, we use sessions (defined in Section 3) collected from all of 2018 for training, and reserve sessions from January through April 2019 for test. To make the prediction problem meaningful, we select sessions with at least a minimum number of pending and completed tasks; specifically we enforce $|T_c| \geq 10$ and $|T_p| \geq 5$. The training data is split into training and validation parts in a ratio of 9:1. We performed grid search over different hyperparameter values on the validation data and selected the best values. The model is trained with the following hyperparameter settings:

(1) The embedding dimension is set to $d = 32$;
(2) For Task2Vec we enforce $w_{\max} = 5$ and $c_{\max} = 10$;
(3) For TaskSeq2Vec we make $l_{\max} = 5$; and
(4) We set the margin of the hinge-loss to $\xi = 1.0$.

[10] $r^u_{t_-}$ is similarly defined.
[11] https://pytorch.org/.

Baseline Methods. In order to more comprehensively evaluate our model we introduce a number of baseline methods, consisting of naïve rule-based solutions, general ranking models, and sequential ranking approaches. A comparative study across the spectrum of model sophistication, from simplistic methods all the way to state-of-the-art sequential rankers allows us to ensure that the gains we observe with our model justify its complexity. The models we compare against can be divided into three broad categories, defined as follows.

(1) ***Naïve methods*** consist of simple heuristic predictors that rank pending tasks according to their attributes. We distinguish between two methods that leverage recency as a proxy for prioritization.
   - **Creation Recency** hypothesizes that tasks that were created more recently should be ranked higher than tasks that have been pending longer. This is, in fact, the default method currently used for task prioritization in some real-world to-do applications, such as Microsoft To Do.
   - **List Recency** builds on the intuition of Creation Recency by also hypothesizing that tasks grouped together in a task list in the Wunderlist application are often tackled together. Specifically, this method first ranks pending tasks based on the list to which the most recently completed task was assigned, then ranks the remaining tasks (not in that list) using Creation Recency. If two or more pending tasks belong to the same list, Creation Recency is used as a tie-breaker for ranking purposes.

(2) ***General ranking models*** are the category of ranking methods that predict an ordering over pending tasks by featurizing pairwise task attributes. These models are similar to *TaskRank* in that a score is assigned to pending tasks by computing the dot product between a user embedding and task feature vector. We differentiate two variants of this baseline approach.
   - **Linear Model** represents tasks as a concatenation of their textual and time-based numeric contextual attributes. Here, the text representation of a task is obtained by using pre-trained GloVe embeddings [66] to featurize words and then take their concatenation. In this model, the user embedding is simply the featurized representation of the most recently completed task, similarly represented using GloVe. A linear model is fit to the pairwise ranking loss in Equation (9) by computing the weighted inner product between the user embedding and a pending task feature vector.
   - **Neural Network** is almost identical to the Linear Model. The only difference is that task features are computed, not by a linear combination with a weight vector, but rather by passing them through a single fully-connected non-linear layer.

   In both these models, once training has been completed, inferences can be made on unseen sessions by using the simplified, model-appropriate user and task representations and computing task preference scores with Equation (10).

(3) ***Sequential ranking models*** build on the *general ranking models* by modeling the sequential associations between completed tasks in a user's history of interactions. Specifically we compare against two popular methods from the literature to model sequences for ranking: **convolutional neural networks** (**CNN**) [76], and **gated recurrent units** (**GRU**) [34].
   - **CNN4Vec** begins by constructing representations for tasks in the same way as the *general ranking models*. However, instead of simply using the most recently completed task as the user embedding, a history of completed tasks is instead passed through a CNN layer followed by a **fully-connected layer** to generate an aggregate view of the sequential interaction between tasks. This aggregate representation is then set as the user embedding to train the pairwise loss from Equation (9).

- **GRU4Vec** is very similar to the CNN4Vec model. However, instead of using a CNN to compute an aggregate representation for the history of completed tasks, a GRU is used to generate hidden representations that are combined with a **mean pooling layer**.

  As with the *general ranking models*, CNN4Vec and GRU4Vec can be used to make inferences on pending tasks in unseen sessions using the task preference score in Equation (10).

Evaluation Metrics. Since our problem fits a ranking setup we evaluate model performance on standard and popular metrics in the ranking literature [53]. Specifically, for every model we report:

- Precision@n, where $n = 1, 2, 3$. The values for n were chosen because we observed from the Wunderlist data that the average number of tasks marked as complete in a day is three.
- **Mean Reciprocal Rank (MRR).** Besides being a popular ranking metric, we selected MRR due to its simplicity and easy interpretation, while attaching a high priority to the first element in a ranked list.

The former definitions of these two metrics are as follows.

- **Precision@n.** This reflects the precision of top-n ranked pending tasks, which is defined as:

$$Precision@n = \frac{1}{|S_{test}|} \sum_{s_i \in S_{test}} \frac{|\hat{R_n}(s_i) \bigcap R_n(s_i)|}{n}, \tag{11}$$

  where $R_k(s_i)$ and $\hat{R_k}(s_i)$ denote the sets of true and predicted top-n pending tasks (the pending tasks completed in the top-n order) of session s_i, S_{test} is the set of all task sessions in the test data.
- **MRR.** This measures the ranking quality based on the rank position of the first pending task, which is defined as:

$$MRR = \frac{1}{|S_{test}|} \sum_{s_i \in S_{test}} \frac{1}{r^b_{s_i}}, \tag{12}$$

  where $r^b_{s_i}$ represents the ranking position of first completed pending task of session s_i.

For each of these metrics, scores are computed by micro-averaging results across all sessions in the test set.

6.2 Results

Performance Comparison. The result of our primary set of experiments are reported in Tables 3 and 4, with the length of the pending task window (denoted by τ in Figure 2) being set to one day and one week, respectively. The best performing models at different metrics (Precision@n and MRR) are highlighted in bold and the best baseline results are indicated by underlines. Overall, the scores indicate that task prioritization is a difficult problem. Nevertheless, the results indicate that there is signal present in the data that we are successfully able to capture. Notably, this is evidenced by our model's gains over all baseline methods, on all metrics and in both experimental settings. The results are statistically significant at $p < 0.01$ using a paired t-test. Additionally, and unsurprisingly, the improvements of *TaskRank* over baseline methods on a one week window are smaller than a one day window. This is because one week windows are likely to have more pending tasks to rank, thereby increasing the uncertainty of modeling sequential dependencies from previously completed tasks in *TaskRank*.

Table 3. Performance for One Day Window

Method	Precision@1	Precision@2	Precision@3	MRR
Creation Recency	0.238	0.368	0.512	0.453
List Recency	<u>0.267</u>	0.386	0.523	0.476
Linear Model	0.254	0.398	0.547	0.479
Neural Network	0.257	0.409	0.553	0.485
CNN4Vec	0.261	<u>0.416</u>	0.555	<u>0.488</u>
GRU4Vec	0.261	0.412	<u>0.556</u>	<u>0.488</u>
TaskRank	**0.283**	**0.437**	**0.571**	**0.510**

Differences between *TaskRank* and each baseline method are statistically significant at $p <$ 0.01 using a paired t-test. The highest overall values are bolded. The highest baseline values are underlined.

Table 4. Performance for One Week Window

Method	Precision@1	Precision@2	Precision@3	MRR
Creation Recency	0.268	0.405	0.540	0.486
List Recency	<u>0.274</u>	0.407	0.539	0.491
Linear Model	0.270	0.416	0.553	0.494
Neural Network	0.271	0.418	0.555	0.495
CNN4Vec	0.271	<u>0.419</u>	<u>0.557</u>	0.496
GRU4Vec	<u>0.274</u>	<u>0.419</u>	<u>0.557</u>	<u>0.497</u>
TaskRank	**0.283**	**0.429**	**0.563**	**0.506**

Differences between *TaskRank* and each baseline method are statistically significant at $p <$ 0.01 using a paired t-test. The highest overall values are bolded. The highest baseline values are underlined.

Interestingly, List Recency performs unusually strongly on Precision@1 on both one day and one week windows, indicating how a relatively simple intuition—that users tend to focus on tasks within lists together, rather than jumping between lists—can yield surprising results. More generally, the best set of baselines are the *sequential ranking models*, which demonstrates the importance of modeling the dependency between consecutive tasks.

Collectively, these results also have some key insights for the design of future user-facing systems for task prioritization. Firstly, the Precision@n numbers indicate the need for an interface that provides users with multiple ranked task options rather than simply a single top-ranked suggestion. This is because at 28.3%, even the best Precision@1 of *TaskRank* is likely not high enough to provide tangible value to a user.[12] An important investigation for future work is to conduct a focused user study to find the number n at which Precision@n becomes a beneficial tool for the user, without inundating them with too many options.

Additionally, the results seem to indicate a benefit in designing a system that only models a one day window, instead of an entire week. In this scenario, not only does *TaskRank* yield the best absolute numbers across metrics, additionally (and perhaps more importantly) it also represents a simpler modeling setup, thereby reducing effort for learning and inference. Interestingly, every other baseline works better on a one week window when compared with itself on a one day window. However, the numerical differences—while sometimes statistically significant—are small enough that the added complexity of modeling a week's worth of historical data is arguably not beneficial in the cost of data or computation.

[12] Arguably, even the highest Precision@3 number of 57.1% is fairly pedestrian, from a user's perspective.

Table 5. Ablation Study of *TaskRank*, with the Removal of Different Task Attributes

Metric	Precision@1	Precision@2	Precision@3	MRR
− Task Title	0.281	0.419	0.553	0.506
%Δ	−0.7%	−4.3%	−3.3%	−0.8%
− List	0.272	0.427	0.564	0.499
%Δ	−4.0%	−2.3%	−1.2%	−2.2%
− Longevity	0.239	0.386	0.539	0.466
%Δ	−18.4%	−13.2%	−4.6%	−9.4%
− Time Since Last Task	0.258	0.417	0.560	0.486
%Δ	−9.7%	−4.8%	−2.0%	−4.9%
All Features	**0.283**	**0.437**	**0.571**	**0.510**

Differences between *TaskRank* and each variant model are statistically significant at $p < 0.05$ using a paired t-test.

Table 6. Ablation Study with Module Variants: MP denotes Mean Pooling Layer, FC is Fully-connected Layer

Metric	Precision@1	Precision@2	Precision@3	MRR
TaskSeq2Vec-MP	0.272	0.422	0.562	0.498
%Δ	−4.0%	−3.6%	−1.6%	−2.4%
TaskSeq2Vec-FC	0.273	0.428	0.567	0.501
%Δ	−3.7%	−2.1%	−0.7%	−1.8%
Task2Vec-MP	0.282	0.424	0.563	0.504
%Δ	−0.4%	−3.1%	−1.4%	−1.2%
Task2Vec-FC	0.273	0.419	0.561	0.497
%Δ	−3.7%	−4.3%	−1.8%	−2.6%
TaskRank	**0.283**	**0.437**	**0.571**	**0.510**

Differences between *TaskRank* and each variant model are statistically significant at $p < 0.05$ using a paired t-test.

Finally, the predictive strength of recency within a list and the sequential interactions and dependencies between previously completed tasks warrants further qualitative investigation. A user study that probes these findings and can explain them from a user-centric standpoint would be immensely beneficial in designing a system that can not only rank tasks by priority but also explain the ranking to users. Over time this helps build trust with the user, transparency into the system's workings, and the ability for the model to fail gracefully.

Ablation Study. In addition to evaluating our model's performance against baselines, we also conduct ablation studies to evaluate the contribution of different feature attributes and module components of *TaskRank*. These results for the one day time windows are reported in Tables 5 and 6.

Specifically, in Table 5 we measure the impact of removing each of the four feature attributes (see Section 4) of a task in turn: its Title, the List it belongs to, its Longevity, and the Time since the last task was created. From this table, we gather that each of the four feature attributes contributes positively to model performance, as demonstrated by the performance drop measured after removing features. The differences with the variant of the model that uses all features are statistically significant at $p < 0.05$ using a paired t-test. Interestingly, the contextual time-based features seem to be significantly more important for overall performance than the textual content features. In particular, the importance of the Longevity attribute indicates why relatively naïve baselines such as Task or List Recency are still performant.

In the ablation study in Table 6, we measure the impact of replacing *TaskRank*'s neural module components with other variants. For the TaskSeq2Vec module we study how the replacement of the sequential encoding component (Equations (7) and (8)) with a simpler variant impacts results. We distinguish two cases: one where we pass the encoding of previously completed tasks through a mean pooling layer, and another where we pass them through a fully-connected layer. We also study two variants of the Task2Vec module. In the first, we replace the attention mechanism over textual attribute encodings (Equation (4)) with a mean pooling layer, and in the second, we substitute the feature aggregation function (Equation (6)) with a fully-connected layer.

According to the results in Table 6, substituting *TaskRank*'s modules with variants leads to worse performance in all cases. The differences with the original *TaskRank* model are statistically significant at $p < 0.05$ using a paired t-test. This suggests that our choices for the different neural components of *TaskRank* are, in fact, sound. For TaskSeq2Vec, the replacement with a fully-connected layer yields better results than a mean pooling layer—but both are worse than full *TaskRank*'s LSTM layer. Interestingly, however, both fully-connected layer and mean pooling layer variants still yield better results than the *sequential ranking* baselines. This means that while modeling sequences with appropriate neural architecture is important, so is the featurization of task attributes and their representation with expressive embeddings. The importance of this featurization is separately demonstrated in the drop in performance we observe when components of Task2Vec are swapped out for less expressive variants.

Interestingly, the study highlights the difference in impact when ablating different features or components at different levels of Precision@n. For example, while the omission of the Longevity feature leads to a massive drop of 18.4% in Precision@1 numbers, it leads to a much more modest drop of 4.6% in Precision@3. Conversely, ablating the Task Title creates almost no impact in Precision@1 (with only a 0.7% drop), while Precision@2 and Precision@3 scores drop more significantly (at 4.3% and 3.3%, respectively). A similar trend, albeit less pronounced, is also observed in the replacement of model components; take, for example, using mean pooling layers for TaskSeq2Vec and Task2Vec instead of more complex variants in the full *TaskRank* model.

Importantly, the Precision@3 scores all demonstrate a comparatively even performance drop across the ablation of features and model components. This implies that not only do all our features and modeling choices positively impact performance—as we have previously already confirmed—they become roughly equally important when a ranked list of several task options is yielded by our model. In conjunction with our previously highlighted insight that a user-facing application with *TaskRank* is likely to be more acceptable to users when displaying a list of several ranked task options (due to the higher Precision@n scores for progressively greater values of n), this leads us to recommend designing and deploying a task prioritization system that uses the full *TaskRank* model.

Nevertheless, this ablation study provides a useful gauge for deciding which features to omit, or which model components to replace with simpler options, should the practical needs of a real-world system not support the full deployment of *TaskRank*.

7 DISCUSSION AND CONCLUSIONS

Time management is a key desiderata of information systems such as task management applications and digital assistants. Yet most of these systems continue to place the burden of task triage and scheduling on their users. While the long-term vision of a digital assistant is the prescient ability to surface the right tasks to users at the right time, that vision remains elusive. As a first step in that direction, we have presented an attempt at building a fully automated context-aware method for prioritizing pending tasks to help users of task management applications organize their schedules more efficiently.

To demonstrate that our approach is practical, grounded, and learnable from real user data, we were able to leverage limited access to a large sample of anonymized and de-identified logs from the now-defunct Wunderlist task management application. We conducted detailed analyses on these logs, focusing on user behavior surrounding task completion. These analyses led us to construct filtering strategies that allowed us to extract approximately reliable training labels for our target scenario of grounded task prioritization. Although prioritization is typically associated with urgency and importance [55, 57], we presented initial evidence that shows that this may only be part of the reason why people complete their tasks in the order they do. Our findings are, to the best of our knowledge, the first to shed light on the way users interact with digital task management tools, as they create and complete tasks, at large scale.

Finally, we designed a neural model—named *TaskRank*—to tackle the problem of automatic pending task prioritization. Notably, this model contextualizes its predictions, considering both task attributes as well as the sequential interactions between them. We conducted experiments demonstrating the efficacy of the model, which outperformed several baselines and previous approaches to sequential ranking. The performance of *TaskRank*, especially compared with heuristic methods currently in production in task management applications such as Microsoft To Do, warrant a deployment in user-facing operational systems, with suitably instrumented feedback mechanisms. These mechanisms are needed to ensure the continued improvement and adaptation of the model from user feedback.

Despite these promising initial results, we must acknowledge some limitations in our approach that stem from the retrospective nature of the log-based analyses and modeling we conducted. Our assumptions about the data, while reasonable first approaches, have not been validated by follow-up studies.

First, we assume that the order in which users completed their tasks is, in fact, the order we should be optimizing. This assumption adopts a *descriptive* view of the data (maximizing the explanatory power of the model); an alternative—and potentially better—approach is the *prescriptive* view (where we attempt to maximize user engagement and productivity). Of course, given the offline nature of our data, model and experiments, a *prescriptive* approach was not possible, but remains a potentially impactful avenue for future research. Our second assumption is that the order in which people mark tasks as complete, corresponds to the actual order in which they complete their tasks. The analyses in Section 4.3 demonstrated that this assumption does not always hold true, and while we did take steps to remove the most egregious sources of noise using targeted filtering rules, the data is still likely to contain residual noise. Future work could attempt to quantify the magnitude of this residual noise, using, for example, a smaller scale targeted study, as well as measure its impact on the *TaskRank* model. Third, we focus on active users (e.g., active for 30 days, created at least 20 tasks) given the need to observe sufficient numbers of tasks to train and test the prioritization models proposed and studied in this article. We also dropped lists where the temporal ordering of completion events is not informative (e.g., groceries, shopping, and packing). While we could apply similar filters in production, an important research direction involves understanding the impact of data filtering strategies on model performance and how we can make grounded task prioritization more robust and resilient to noisy user signal. More work is needed to understand the impact of the data filtering on our results and experiment with other filter settings. We also need to develop strategies to handle the cold start problem, when we have little or no historic data about users. Finally, our insights are based on interactions with a single task application, Wunderlist, one of the most popular to-do applications on the market. While we had unique, temporary access to anonymized and de-identified Wunderlist log data, it is only one application and there may be nuances in the interface design that affect the data available. Going forward, we need to understand how well these findings generalize to other task management applications or scenarios. We also need public datasets for large-scale task analysis. To date, we have released a dataset from a related study on complex task decomposition [88][13] and plan to also do so for ongoing studies, e.g., on context-aware task recommendation [91].

Our initial foray into grounded task prioritization opens up several avenues of future research. While we no longer have access to the large-scale logs that facilitated this work, we are looking to investigate using privacy-preserving machine learning techniques (such as differential privacy [1] or federated learning [56]) in order to train and improve our model in online settings without compromising users' rights to privacy. One such initiative is the improvement of our model by privately incorporating richer sources of contextual information as features in the model —such as, for example, knowledge about cyclical or repeating tasks, complex goals that can be further broken down into smaller, more manageable sub-tasks, and information about users including their schedules, location, or their social and professional connections. Survival analysis methods (e.g., a Cox proportional-hazards model [52]) may be appropriate for the challenge of ranking tasks by time-to-completion and should be considered in future work on pending task prioritization. Deeper analysis of the results, such as which tasks benefit most from our method, which users benefit most, and which features (not just feature classes) carry most evidential weight in the model are essential elements of future work in follow-up studies. We no longer have access to the data required to perform such analysis now.

Finally, we are also interested in deploying *TaskRank* in a production setting and evaluating its performance online with real users. This raises a host of challenging research questions around how to integrate task prioritization with existing task management and calendaring applications in ways that are natural to users and do not disrupt flow, how to collect meaningful user feedback (both implicitly and explicitly) and telemetry data from interactions with the system, and what success metrics to compute to measure the performance of the prioritization models. Integration options include providing users with a list of prioritized task suggestions, from which they can select the highest-priority tasks (as in Figure 1). These tasks can be added to users' to-do lists for a particular day and/or added as appointments to their calendars, to lessen the chance of forgetting. High-priority tasks can also become reminders which can be triggered by contextual elements of the tasks, e.g., time, location, people. Priority is a combination of (at least) urgency *and* importance: it is personal, and it is contextual. All of these factors combine to make it difficult, if not impossible, for third-party judges to reliably assess the priority of a task for a given user. We addressed this issue in our study by deriving per-user task orderings for training and testing the models from anonymized and de-identified log data. Post-deployment, we would have access to data about user engagement with the prioritized task lists (e.g., percentage of task lists with at least one selected suggestion, average rank of selected suggestions, percentage of suggestions selected), which could be used to evaluate the prioritization algorithms. Optimizing the user experience, including explaining the model's prioritization ordering so as to build trust with users, is also an important research direction that needs to be investigated for a successful deployment, as is allowing users to provide feedback that can be used to personalize the ordering for future task rankings. Task *feasibility* is an important future step in realizing prioritization in practice. Beyond simply ranking the tasks in estimated order of priority, we also need to understand whether the tasks are even achievable given the situation and the resources available during the day. For example, some tasks (e.g., "take out trash," "fold laundry") may only be doable at home, while others may only be doable in a work setting (e.g., "team all hands meeting," "fix office printer"). Combining task feasibility and task priority can help to ensure that only important tasks that are possible in the current context or a future context are surfaced to users.

[13]https://github.com/microsoft/MSComplexTasks.

Overall, the primary contributions of this submission are introducing the novel research challenge of pending task prioritization, developing sophisticated methods to address that challenge (grounded in large-scale task data that we had temporary access to), and showing that these methods perform well compared to state-of-the-art baselines. There are many exciting avenues for future work, many of which are outlined in the text above. This includes deploying the models at scale in live task management systems (first in flights and perhaps then in production) and evaluating the system with real user feedback, which requires metric definitions and mechanisms for integrating implicit and explicit feedback into the user experience and modeling infrastructure. The broader area of task intelligence, within which methods such as these fall, is important and needs significant attention from the research community, especially at a time (during the COVID-19 pandemic) when many employees are still working from home and there are so many competing demands on people's time and attention across their work and personal lives.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. *Recommender Systems Handbook*. Springer, Boston, MA217–253.

[3] David Allen. 2015. *Getting Things Done: The Art of Stress-free Productivity*. Penguin.

[4] Catherine M. Arrington, Starla M. Weaver, and Rachel L. Pauker. 2010. Stimulus-based priming of task choice during voluntary task switching. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 36, 4 (2010), 1060.

[5] Nikola Banovic, Christina Brant, Jennifer Mankoff, and Anind Dey. 2014. ProactiveTasks: The short of mobile device use sessions. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 243–252.

[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 ACM 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] Jan R. Benetka, John Krumm, and Paul N. Bennett. 2019. Understanding context for tasks and activities. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*. 133–142.

[9] Paul N. Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 585–586.

[10] 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* 26, 4 (2008), 1–46.

[11] Pauline M. Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. 2011. PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology* 2, 4 (2011), 40.

[12] Ann E. Blandford and Thomas R. G. Green. 2001. Group and individual time management tools: what you get is not what you need. *Personal and Ubiquitous Computing* 5, 4 (2001), 213–230.

[13] R. N. Brewer, Meredith Ringel Morris, and Siân E. Lindley. 2017. How to remember what to remember: Exploring possibilities for digital reminder systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 38.

[14] Huanhuan Cao, Tengfei Bao, Qiang Yang, Enhong Chen, and Jilei Tian. 2010. An effective approach for mining mobile user habits. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. 1677–1680.

[15] David Maxwell Chickering, David Heckerman, and Christopher Meek. 1997. A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence* (1997), 80–89.

[16] Brigitte J. C. Claessens, Wendelien Van Eerde, Christel G. Rutte, and Robert A. Roe. 2010. Things to do today...: a daily diary study on task completion at work. *Applied Psychology* 59, 2 (2010), 273–295.

[17] Lacey Colligan and Ellen J. Bass. 2012. Interruption handling strategies during paediatric medication administration. *BMJ Quality & Safety* 21, 11 (2012), 912–917.

[18] Kurt Colvin, Ken Funk, and Rolf Braune. 2005. Task prioritization factors: Two part-task simulator studies. *The International Journal of Aviation Psychology* 15, 4 (2005), 321–338.

[19] Kenneth Conley and James Carpenter. 2007. Towel: Towards an intelligent to-do list. In *Proceedings of the AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*. 26–32.

[20] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. 2017. Calendar. help: Designing a workflow-based scheduling agent with humans in the loop. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing System*. 2382–2393.

[21] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 175–182.

[22] Anind K. Dey and Gregory D. Abowd. 2000. Cybreminder: A context-aware system for supporting reminders. In *Proceedings of the International Symposium on Handheld and Ubiquitous Computing*. Springer, 172–186.

[23] Geoffrey B. Duggan, Hilary Johnson, and Petter Sørli. 2013. Interleaving tasks to improve performance: Users maximise the marginal rate of return. *International Journal of Human-Computer Studies* 71, 5 (2013), 533–550.

[24] Mark Forster. 2006. *Do It Tomorrow and Other Secrets of Time Management*. Hachette U.K.

[25] Michael Freed, Jaime G. Carbonell, Geoffrey J. Gordon, Jordan Hayes, Brad A. Myers, Daniel P. Siewiorek, Stephen F. Smith, Aaron Steinfeld, and Anthony Tomasic. 2008. RADAR: A personal assistant that learns to reduce email overload.. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1287–1293.

[26] Yiqun Gan, Yu Wang, Ran Meng, Min Wen, Guangyu Zhou, Yingyue Lu, and Miao Miao. 2015. Temporal discounting mechanisms of future-oriented coping: Evidence from delay discounting and task prioritization paradigms. *Journal of Behavioral Decision Making* 28, 5 (2015), 529–541.

[27] Yolanda Gil and Varun Ratnakar. 2008. Towards intelligent assistance for to-do lists. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*. 329–332.

[28] David Graus, Paul N. Bennett, Ryen W. White, and Eric Horvitz. 2016. Analyzing and predicting task reminders. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. 7–15.

[29] Robert S. Gutzwiller, Christopher D. Wickens, and Benjamin A. Clegg. 2016. The role of time on task in multi-task management. *Journal of Applied Research in Memory and Cognition* 5, 2 (2016), 176–184.

[30] Jacek Gwizdka. 2004. Email task management styles: the cleaners and the keepers. In *Proceedings of the ACM SIGCHI Extended Abstracts on Human Factors in Computing Systems*. 1235–1238.

[31] Mona Haraty and Joanna McGrenere. 2016. Designing for advanced personalization in personal task management. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 239–250.

[32] Mona Haraty, Joanna McGrenere, and Charlotte Tang. 2016. How personal task management differs across individuals. *International Journal of Human-Computer Studies* 88 (2016), 13–37.

[33] Negar Hariri, Bamshad Mobasher, and Robin D. Burke. 2013. Query-driven context aware recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 9–16.

[34] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

[35] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[36] Eric Horvitz, Andy Jacobs, and David Hovel. 1999. Attention-sensitive alerting. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. 305–313.

[37] Eric J. Horvitz, Paul Koch, Carl Kadie, and Andy Jacobs. 2012. Coordinate: Probabilistic forecasting of presence and availability. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence* (2012), 224–233.

[38] Tracy Janssen and John S. Carton. 1999. The effects of locus of control and task difficulty on procrastination. *The Journal of Genetic Psychology* 160, 4 (1999), 436–442.

[39] Ece Kamar and Eric Horvitz. 2011. Jogger: Models for context-sensitive reminding. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. Vol. *3*. 1089–1090.

[40] Amirrudin Kamsin, Ann Blandford, and Anna L. Cox. 2012. Personal task management: My tools fall apart when I'm very busy! In *Proceedings of the ACM SIGCHI Extended Abstracts on Human Factors in Computing Systems*. 1369–1374.

[41] Angela Kessell and Christopher Chan. 2006. Castaway: A context-aware task management system. In *Proceedings of the ACM SIGCHI Extended Abstracts on Human Factors in Computing Systems*. 941–946.

[42] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[43] David Kirsh. 2000. A few thoughts on cognitive overload. *Intellectica* 1, 30 (2000), 19–51.

[44] Julia Kiseleva, Hoang Thanh Lam, Mykola Pechenizkiy, and Toon Calders. 2013. Discovering temporal hidden contexts in web sessions for user trail prediction. In *Proceedings of the 22nd International Conference on the World Wide Web*. 1067–1074.

[45] Julia Kiseleva, Alexander Tuzhilin, Jaap Kamps, Melanie JI Mueller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, and Djoerd Hiemstra. 2016. Beyond movie recommendations: Solving the continuous cold start problem in e-commerce recommendations. arXiv:1607.07904. Retrieved from https://arxiv.org/abs/1607.07904.

[46] Julia Kiseleva, Kyle Williams, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016. Predicting user satisfaction with intelligent assistants. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 45–54.

[47] Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016. Understanding user satisfaction with intelligent assistants. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*. 121–130.

[48] Nicolas Kokkalis, Thomas Köhn, Johannes Huebner, Moontae Lee, Florian Schulze, and Scott R. Klemmer. 2013. Taskgenies: Automatically providing action plans helps people complete tasks. *ACM Transactions on Computer-Human Interaction* 20, 5 (2013), 1–25.

[49] Mik Lamming and Mike Flynn. 1994. Forget-me-not: Intimate computing in support of human memory. In *Proceedings of the International Symposium on Next Generation Human Interface*. 4.

[50] Gilly Leshed and Phoebe Sengers. 2011. "I lie to myself that i have freedom in my own schedule" productivity tools and experiences of busyness. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 905–914.

[51] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on the World Wide Web*. 661–670.

[52] Danyu Y. Lin and Lee-Jen Wei. 1989. The robust inference for the cox proportional hazards model. *Journal of the American Statistical Association* 84, 408 (1989), 1074–1078.

[53] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[54] Gloria Mark. 2015. Multitasking in the digital age. *Synthesis Lectures on Human-Centered Informatics* 8, 3 (2015), 1–113.

[55] Brett McKay and Kate McKay. 2013. The eisenhower decision matrix: How to distinguish between urgent and important tasks and make real progress in your life. *A Man's Life, Personal Development* (2013).

[56] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera Y. Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[57] Sophie Middleton, Alexandra Charnock, Sarah Forster, and John Blakey. 2019. Factors affecting individual task prioritisation in a workplace setting. *Future Healthcare Journal* 6, Suppl 1 (2019), 114.

[58] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*.

[59] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Conference on Neural Information Processing Systems*. 3111–3119.

[60] Norman A. Milgram, Barry Sroloff, and Michael Rosenbaum. 1988. The procrastination of everyday life. *Journal of Research in Personality* 22, 2 (1988), 197–212.

[61] Sudipto Mukherjee, Subhabrata Mukherjee, Marcello Hasegawa, Ahmed Hassan Awadallah, and Ryen White. 2020. Smart to-do: Automatic generation of to-do items from emails. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[62] Karen Myers, Pauline Berry, Jim Blythe, Ken Conley, Melinda Gervasio, Deborah L. McGuinness, David Morley, Avi Pfeffer, Martha Pollack, and Milind Tambe. 2007. An intelligent personal assistant for task and time management. *AI Magazine* 28, 2 (2007), 47–47.

[63] Hamid R. Motahari Nezhad, Kalpa Gunaratna, and Juan Cappi. 2017. eAssistant: Cognitive assistance for identification and auto-triage of actionable conversations. In *Proceedings of the 26th International Conference on the World Wide Web*. 89–98.

[64] Gregory B. Northcraft, Aaron M. Schmidt, and Susan J. Ashford. 2011. Feedback and the rationing of time and effort among competing tasks. *Journal of Applied Psychology* 96, 5 (2011), 1076.

[65] Celeste Lyn Paul, Anita Komlodi, and Wayne Lutters. 2015. Interruptive notifications in support of task management. *International Journal of Human-Computer Studies* 79 (2015), 20–34.

[66] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[67] Ioannis Refanidis and Anastasios Alexiadis. 2011. Deployment and evaluation of selfplanner, an automated individual task management system. *Computational Intelligence* 27, 1 (2011), 41–59.

[68] Ioannis Refanidis and Neil Yorke-Smith. 2010. A constraint-based approach to scheduling an individual's activities. *ACM Transactions on Intelligent Systems and Technology* 1, 2 (2010), 12.

[69] Bradley J. Rhodes. 1997. The wearable remembrance agent: A system for augmented memory. *Personal Technologies* 1, 4 (1997), 218–224.

[70] Chirag Shah and Ryen W. White. 2021. Bridging task expressions and search queries. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval.* 319–323.

[71] Chirag Shah and Ryen W. White. 2021. Task intelligence for search and recommendation. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 13, 3 (2021), 1–160.

[72] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. 2014. CARS2: Learning context-aware representations for context-aware recommendations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management.* 291–300.

[73] Simone Stumpf, Xinlong Bao, Anton Dragunov, Thomas G. Dietterich, Jon Herlocker, Kevin Johnsrude, Lida Li, and J. Shen. 2005. Predicting user tasks: I know what you're doing. In *Proceedings of the AAAI Workshop on Human Comprehensible Machine Learning.*

[74] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning.* 1017–1024.

[75] Christine J. Syrek, Oliver Weigelt, Corinna Peifer, and Conny H. Antoni. 2017. Zeigarnik's sleepless nights: How unfinished tasks at the end of the week impair employee sleep on the weekend through rumination. *Journal of Occupational Health Psychology* 22, 2 (2017), 225.

[76] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining.* 565–573.

[77] Carlos Toxtli, Andrés Monroy-Hernández, and Justin Cranshaw. 2018. Understanding chatbot-mediated task management. In *Proceedings of the ACM SIGCHI conference on Human Factors in Computing Systems.* 1–6.

[78] Wei Wang, Saghar Hosseini, Ahmed Hassan Awadallah, Paul N. Bennett, and Chris Quirk. 2019. Context-aware intent identification in email conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 585–594.

[79] Ryen W. White, Peter Bailey, and Liwei Chen. 2009. Predicting user interests from contextual information. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 363–370.

[80] Ryen W. White and Ahmed Hassan Awadallah. 2019. Task duration estimation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining.* 636–644.

[81] Ryen W. White, Ahmed Hassan Awadallah, and Robert Sim. 2019. Task completion detection: A study in the context of intelligent systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 405–414.

[82] Ryen W. White, Elnaz Nouri, James Woffinden-Luey, Mark Encarnación, and Sujay Kumar Jauhar. 2021. Microtask detection. *ACM Transactions on Information Systems* 39, 2 (2021), 1–29.

[83] Steve Whittaker, Victoria Bellotti, and Jacek Gwizdka. 2006. Email in personal information management. *Communication of the ACM* 49, 1 (2006), 68–73.

[84] Lina Yao, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. 2015. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1007–1010.

[85] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 5709–5716.

[86] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 363–372.

[87] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2025–2034.

[88] Yi Zhang, Sujay Kumar Jauhar, Julia Kiseleva, Ryen White, and Dan Roth. 2021. Learning to decompose and organize complex tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 2726–2735.

[89] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S. Sheng, and Xiaofang Zhou. 2019. Where to go next: A spatio-temporal gated network for next poi recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 5877–5884.

[90] Qian Zhao, Paul N. Bennett, Adam Fourney, Anne Loomis Thompson, Shane Williams, Adam D. Troy, and Susan T. Dumais. 2018. Calendar-aware proactive email recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval.* 655–664.

[91] Sujay Kumar Jauhar, Nirupama Chandrasekaran, Michael Gamon, and Ryen W. White. 2021. MS-LaTTE: A dataset of where and when to-do tasks are completed. *arXiv 2111.06902.*

Received March 2021; revised July 2021; accepted September 2021