Microtask Detection

RYEN W. WHITE, Microsoft Research
ELNAZ NOURI, Microsoft Research
JAMES WOFFINDEN-LUEY, Microsoft Research
MARK ENCARNACIÓN, Microsoft Research
SUJAY KUMAR JAUHAR, Microsoft Research

Information systems, such as task management applications and digital assistants, can help people keep track of tasks of different types and different time durations, ranging from a few minutes to days or weeks. Helping people better manage their tasks and their time are core capabilities of assistive technologies, situated within a broader context of supporting more effective information access and use. Throughout the course of a day, there are typically many short time periods of downtime (e.g., five minutes or less) available to individuals. Microtasks are simple tasks that can be tackled in such short amounts of time. Identifying microtasks in task lists could help people utilize these periods of low activity to make progress on their task backlog. We define *actionable* tasks as self-contained tasks that need to be completed or acted on. However, not all to-do tasks are actionable. Many task lists are collections of miscellaneous items that can be completed at any time (e.g., books to read, movies to watch), notes (e.g., names, addresses), or the individual items are constituents in a list that is itself a task (e.g., a grocery list). In this article, we introduce the novel challenge of microtask detection, and present machine-learned models for automatically determining which tasks are actionable and which of these actionable tasks are microtasks. Experiments show that our models can accurately identify actionable tasks, accurately detect actionable microtasks, and that we can combine these models to generate a solution that scales microtask detection to all tasks. We discuss our findings in detail, along with their limitations. These findings have implications for the design of systems to help people make the most of their time.

CCS Concepts: • **Information systems** → **Information systems applications**; **Task models**; • **Computing methodologies** → *Artificial intelligence*; **Intelligent agents**.

Additional Key Words and Phrases: Tasks, Microtask detection, Information access, Information systems

ACM Reference Format:
Ryen W. White, Elnaz Nouri, James Woffinden-Luey, Mark Encarnación, and Sujay Kumar Jauhar. 2020. Microtask Detection. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2020), 28 pages. https://doi.org/10.1145/3432290

1 INTRODUCTION

Tasks pervade almost every aspect of our daily work and personal lives [1]. They involve different activities, have different constraints, and take different amounts of time to complete. Some tasks can be completed in a short time, while others take longer, sometimes spanning several days or weeks. Evidence of task durations can even be observed in search logs [35, 81], in social media

Authors' addresses: Ryen W. White, ryenw@microsoft.com, Microsoft Research, Redmond, WA, 98052; Elnaz Nouri, elnouri@microsoft.com, Microsoft Research, Redmond, WA, 98052; James Woffinden-Luey, jluey@microsoft.com, Microsoft Research, Redmond, WA, 98052; Mark Encarnación, markenc@microsoft.com, Microsoft Research, Redmond, WA, 98052; Sujay Kumar Jauhar, sjauhar@microsoft.com, Microsoft Research, Redmond, WA, 98052.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
1046-8188/2020/1-ART1 $15.00
https://doi.org/10.1145/3432290

[87], and in news articles [91]. An important aspect of task management is providing people with salient information regarding tasks on their backlog. Serving task content to users falls under the broad research area of supporting more effective information access and use. Task management applications such as Microsoft To Do, Google Tasks, and Todoist (todoist.com) help people track their pending and completed tasks. Digital assistants such as Amazon Alexa or Google Assistant have similar functionality. Studies have found that users of these and similar systems would benefit most from assistance with task planning (e.g., finding time for tasks, deciding what tasks to do when, deciding which tasks to do next) [4]. Scheduling and prioritizing tasks are both challenging activities [62], even though, as we demonstrate in our study, many tasks can be done in a short amount of time. Being able to accurately detect microtasks addresses some of these challenges by enabling systems to help find time for tasks. Surfacing such salient information (e.g., via reminder notifications) can help improve people's awareness about task completion possibilities, helping them better plan their time and improve prospective memory [12].

DEFINITION: We define *microtasks* as self-contained tasks that can be completed quickly (e.g., in five minutes or less). In the context of a to-do application (the focus of the investigation in this article), microtask detection include household chores (e.g., "clean dishes," "feed dogs"), required actions (e.g., "call in prescription," "pay power bill," "check in for flight"), making arrangements (e.g., "make restaurant reservation," "order football tickets"), performing work tasks (e.g., "check work email," "complete timesheet", "review reports"), and relaxation (e.g., "meditate," "listen to music," "play a game"). Microtasks can be standalone or form part of a larger macrotask; which can be difficult to find uninterrupted time to complete [13]. In fact, we observe some evidence of microtasks with the context of larger macrotasks in the labeled dataset collected for this article. For example, one crowdworker in our study appeared to be planning for a trip and listed the following three microtasks: "check in for flight," "download books," and "print directions," plus two non-microtasks: "pack suitcase" and "get travel supplies."

Automatically detecting microtasks in task lists enables many scenarios, e.g., intelligent systems could recommend tasks and/or find time to accomplish them (in short gaps between meetings, etc.) to help people make progress on their to-dos even if they only have a few minutes to spare. Figure 1 shows an example of a related scenario in a task management application or digital assistant. In that figure, the system has identified three microtasks from a user's backlog and is suggesting that they do these if they have some free time. In this example, reminders could persist in the system for the entire day until the user finds time for the tasks or is in a place where they can complete them (e.g., at home for the tasks in the figure). Microtasks may involve physical activities such as mundane chores or digital activities (e.g., common tasks such as email triage have been shown to be accomplished in short bursts of five minutes or less [30]). Implementing this requires automated methods to estimate task duration, an activity that humans have been shown to struggle with in light of cognitive biases such as overoptimism [42]. Previous research on personal task management has shown that automated duration estimation is feasible [83], but focused on time estimation and longer tasks given the nature of the training data that was available (i.e., only calendar appointments, which were found to lack extremely short tasks).

In this article, we present the first study focused on automatically detecting microtasks. This is a core capability for information systems, with many applications. We develop and evaluate machine-learned models to detect microtasks. Simply flagging tasks users can do quickly would boost their awareness and allow them to slot these tasks into an appropriate part of their day. Our experimental results are promising and this research paves the way toward imbuing information systems with the ability to help their users manage *and* accomplish their tasks, starting with those tasks that they can complete quickly. Coupling this capability with intelligent scheduling technologies [36, 62] and context-sensitive reminding techniques [43] would enable systems to

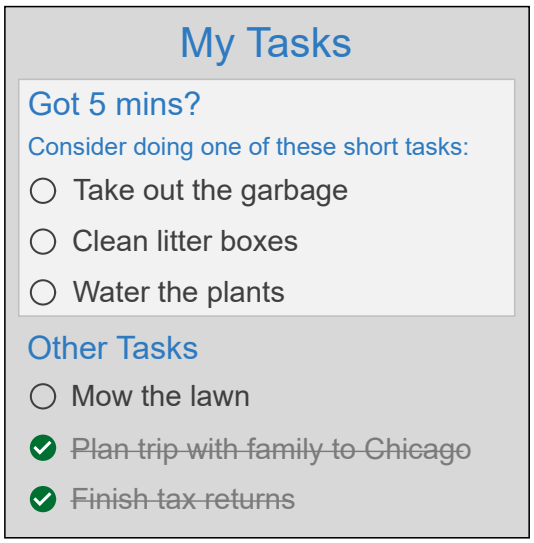


Fig. 1. Example of microtask detection as applied in a task management application or digital assistant. In this example, the system has identified several microtasks in the user's task list (e.g., "water the plants") and is suggesting that they consider doing one of those tasks if they have five minutes of free time on their schedule. Tasks predicted to take longer than five minutes appear under "Other Tasks." Note that in this example there is no indication of *when* or *where* users should complete the tasks, just that they need to do so.

find time to complete these tasks, recommend them as actions during already-reserved periods of focus time, or surface them to users in contexts where they can actually be completed.

The main research contributions of this article are as follows:

- Introduce *microtask detection* as an important new challenge in research on task and time management, and more broadly, in research on information access and use.
- Build machine-learned models to detect microtasks and demonstrate their strong performance on actionable tasks, leveraging signals including personalization;
- Develop an actionable task preclassifier and show that preclassification is necessary to realize reasonable classification performance from microtask detection on general task lists (which also include a sizeable quantity of non-actionable tasks), and;
- Present the implications of microtask detection for the design of information systems to help people more fully utilize their time.

Overall, microtask detection is a novel and important area that is central to the design of systems to help people better manage and utilize their time. The research described in this article addresses a shortcoming in our previous work on time estimation [83] (which focused on longer tasks), advances our understanding of microtasks and the factors that influence the utility of models to detect them (including some promising evidence on the potential for personalization), and paves the way toward enabling information systems to help people manage their tasks and time.

An important part of realizing the value from duration estimation is to first isolate the tasks that are most amenable to it. One such class of tasks are those that are *actionable* (self-contained tasks that need to be completed or acted on, e.g., "mow the lawn" or "submit report"). As part of this study, we developed a classifier to help make that determination and show its value when paired with microtask detection. Whether a task is truly actionable depends on many contextual factors (including time, location, access to resources, etc.) and we only scratch the surface in this study. In task management applications, there are at least actionable tasks, actionable collections (groups of

items usually done together, e.g., grocery lists, packing lists), and notes (items without a specific time frame to act but are recorded for reference, e.g., movies to watch, books to read). We focus on actionable *tasks* for this initial investigation.

The remainder of this article is structured as follows. In Section 2, we present related work. Section 3 presents the data and the methods developed to detect microtasks. In Section 4, we describe the preclassifier to identify actionable tasks. Section 5 describes combining the preclassifier with the microtask detector and the resultant effect on detection performance. We discuss the results and their implications in Section 6 and we conclude in Section 7.

2 RELATED WORK

As mentioned earlier, tasks pervade every aspect of our lives. As such, they have been studied extensively. Previous research in several areas is related to the research described in this article. This includes work on time estimation, task assistance, productivity, crowdsourcing, and tasks in search and retrieval. We will cover each of these areas in turn.

2.1 Time Estimation

Research in the psychology community on time estimation has revealed a number of cognitive biases that can impact people's ability to accurately estimate task durations [41, 42, 49]. This is relevant to our work because these biases mean that users may face difficulty self-identifying microtasks, making machine assistance more necessary. One such bias is the so-called "planning fallacy" [42], which suggests that people are overly optimistic when forecasting task durations. The durations of short future tasks are more likely to be overestimated, whereas longer tasks are more likely to be underestimated [26]. Strategies for overcoming such biases include helping users enumerate the task steps for complex tasks [51], an activity that could be facilitated by assistive technologies to help structure task completion activities.

Typical durations for verbs have been inferred from tweets [87]. Event durations have been mined from news articles [66] and search logs [31, 50]. Research on task intelligence has focused on applying machine learning to estimate different properties of search tasks, such as task completion status [84] and task duration [83]. The latter study focused on task durations inferred from calendar appointments, where short tasks are very infrequent (fewer than 1% of tasks in [83] are 30 minutes or less) or are meant as placeholder reminders (quite often with zero minutes duration) rather than accurate estimates of task duration. This limits the applicability of those methods to the important challenge of microtask detection, which requires sizeable numbers of short tasks to train and test machine-learned models. Other research has studied the temporal commonsense problem, developing methods to reason about the temporal aspects of events (including duration) from natural language using machine reading comprehension [65, 90, 91].

2.2 Task Assistance

Studies have examined how people manage their tasks [4], focused primarily on task tracking and prioritization, and where they need most assistance (mainly in task planning). Strategies to help people better manage their tasks have become popular (e.g., [1]). Task management systems can leverage modalities including email [5, 32] and instant messaging [16], to support more natural engagement, especially for activities such as task delegation [20].

Digital assistants and task management applications can help their users manage their tasks and their time [62], including helping them to organize, filter, prioritize, and execute tasks. Digital assistants allow users to set reminders for pending tasks, and will trigger them based on time or other contextual cues. The TaskGenies system can automatically generate action plans comprising sets of short tasks to assist with task completion [48]. The Lumière system [37] provides intelligent

assistance to software users. Lumière focuses on understanding user goals and needs in order to provide proactive assistance with activities such as document authoring.[1] ProactiveTasks [3] is an interaction lock screen prototype that proactively suggests short tasks to users that extend beyond simple application notifications and into "review" interactions (with overlap with microproductivity support [see next section]). Such tasks can be completed while waiting for another task to complete [11] or during a continuous attention task such as driving [39]. Providing this type of support still requires that systems have knowledge of which tasks are microtasks, which can be user provided (potentially burdensome), rule based (often relying on the presence of structured content [40]), or machine learned (as we tackle in this study).

2.3 Productivity

Research on microproductivity has studied how people can decompose larger macrotasks, e.g., creative composition, into smaller, context-free microtasks (e.g., using templates corresponding to idea generation, idea organization, and writing) [75, 76]. Sophisticated rules can be used to fragment macrotasks (e.g., document editing) into microtasks (e.g., fix spelling errors, accept/reject changes, reply to comments, etc.) that can be completed between interruptions [13, 76]. Related research on self-sourcing [77] applies task decomposition and context maintenance techniques drawn from crowdsourcing to help people complete personal tasks. These methods help people make progress in short periods of downtime and can even improve work quality [13]. Researchers have also examined how people spend time during multitasking [59] and have developed theoretical models of time allocation in this context [19].

Dedicated tools have been developed to help people apply microproductivity and self-sourcing methods to make progress with writing tasks [40, 45] or software development tasks [86] while mobile. Support for reminding people about their microtasks has also been weaved into social media [34]. Beyond authoring tasks, similar concepts of utilizing downtime can also be used to assist with learning activities such as language acquisition [11].

Our definition of microtask is different from microproductivity research. We are focused on user-generated tasks that are short and self-contained, but not necessarily part of a larger task. We focus on the temporal aspect (detecting short tasks), even though research on task feasibility given constraints (e.g., dependencies, other commitments) is relevant [63]. Also relevant is work on so-called "channel factors," the small but critical barriers to action that have a disproportionate effect on whether people complete a goal [70].

2.4 Crowdsourcing

Moving beyond completion of personal tasks, one domain where microtasks have been discussed extensively is in the context of crowdsourcing [71]. Crowdworkers volunteer to complete assigned tasks and short, context-free tasks are appealing to these crowdworkers for several reasons, including scheduling flexibility [60]. Providing crowdworkers with short tasks helps accommodate their time constraints [78], which have been shown to be a barrier to task completion [7]. TA$Ker is a mobile crowdsourcing (or "crowdtasking") [44] application that can suggest short tasks, involving users moving to a specified physical location, based on their predicted movement trajectories. In this research, we focus on microtasks in the context of people's personal to-do tasks rather than tasks assigned to crowdworkers in a crowdsourcing setting.

[1] https://en.wikipedia.org/wiki/Office_Assistant

2.5 Tasks in Search and Retrieval

Beyond personal task management, tasks play a central role in information seeking and retrieval. People turn to search systems to complete a range of tasks [69] and task completion time has also been used to evaluate search algorithms [89]. Search tasks can be short-term (single search sessions) [35] or long-term (spanning multiple sessions) [81]. By considering aspects of the task, we can better predict information behavior [55] and build better information systems [79]. Researchers have studied different characteristics of search tasks that can affect their duration, including complexity [10], difficulty [2], and other facets such as type [56] and time sensitivity [61]. Research on task stage has examined its impact on relevance criteria [73] and on search behavior [52]. Evidence of task completion can be readily observed in search log data [23, 82]. Although search engines can support the completion of some tasks, we have a broad definition of microtasks, which extends beyond search systems and post-hoc analyses of search time. Improving search engines is not the focus of this article, but there are clear applications of related ideas in microtask detection within search and retrieval, e.g., being able to resurface search tasks that searchers repeat periodically (known to be a common activity [74]) and can be completed quickly (e.g., those served with instant answers such as checking weather or checking news headlines [14]). Search may also play an integral role in the completion of microtasks once they have been identified, e.g., to help find required physical or digital resources.

2.6 Contributions Over Prior Work

Our research makes several contributions over prior work. The main contributions are as follows. First, we focus on automatically *detecting* microtasks, an important new challenge in task management (versus having users self-identify short tasks), and information access and use more broadly. We show that we can train machine-learned models to do this accurately, with evidence that performance improves with personalization. Second, we focus on user-generated task lists, comprising "to-do style" tasks ("take out the trash," "water the plants," etc.), rather than the assigned/auto-generated tasks that are common in crowdwork/microproductivity settings. Third, we include all tasks, not just those that are associated with an overarching macrotask and not just those that are actionable (by developing a separate actionable task preclassifier that can be combined with the microtask detector). Fourth, we target short tasks specifically, something that was not possible in previous work on task duration estimation given the calendar data used for that study [83]. Finally, although there are applications of being able to detect and surface short tasks to searchers, our focus on detecting microtasks extends beyond search tasks. We also forecast duration rather than applying it retrospectively (e.g., as a search evaluation metric [89]).

3 MICROTASK DETECTION

We now describe our approach to microtask detection, including the training data, the model settings, and experiments on model performance.

3.1 Data

Data from crowdworkers and usage logs of a popular task management application was used to obtain ground truth labels and generate features for the microtask detection model.

3.1.1 Labeled Data. To get started, we needed training data about to-do tasks and which of them were microtasks. This is challenging because data sources such as calendar (used in [83]) only contain estimates of task duration and lack short and/or mundane tasks. Task creation and completion times from logs can be unreliable (e.g., analysis of the log data outlined in Section 3.1.4 shows that users often mark tasks complete in batches, regardless of completion time). Since microtasks are

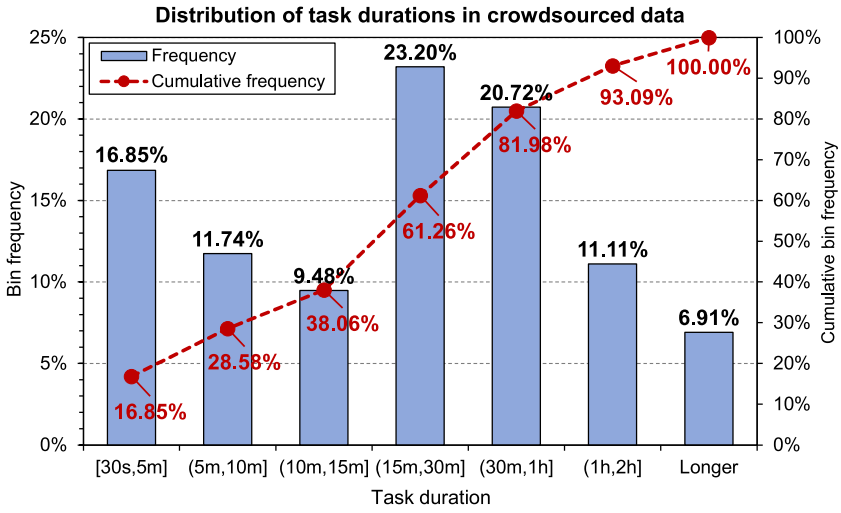


Fig. 2. Distribution of task duration frequencies in the labeled data (n=5,699), per bin and cumulative (s=seconds, m=minutes, h=hours). Each bin corresponds to a duration range that is marked with inclusive and exclusive duration bounds.

fairly infrequent, pulling a random sample of tasks from the logs of a task management application and having crowdworkers label the microtasks or having workers enter their own tasks (including non-actionable tasks), are both highly inefficient ways to obtain microtask samples.

To obtain the ground truth data we needed (but still allow some flexibility to experiment with how we define microtasks), we created a crowdsourcing task. Crowdworkers were sourced from an external vendor (clickworker.com) in the en-US market. Workers were asked to generate a task list comprising tasks they might store in a task management application and had recently completed. A screenshot of the interface for the human intelligence task (HIT) is provided in the Appendix. Workers were free to enter tasks of any kind. Anecdotally, we observe that home and leisure-related tasks were much more common in the data. Workers provided the following information about each task: (1) Task title (a textual description of the task); (2) Best estimate of the total time to complete the task (task duration); (3) A score depicting their certainty in the duration estimate (on a scale from 1-10, 10=highest), and; (4) Whether they attempted the task regularly (yes/no). Asking about regularity was important to understand the nature of the tasks. Intuitively, we might expect to-do tasks (and microtasks in particular) to be repeated often, so if this percentage was low it would signal some potential bias in the tasks collected. Workers were asked to enter different tasks, with no duplicates. The interface for the HIT had space for 10 tasks, with a minimum of five tasks required. To obtain sufficient data and to avoid skewing the data to any one worker, we required at least five tasks from each worker and at most 20 tasks (two HITs) per worker. Crowdworkers were paid 50 cents for each HIT.

Asking workers to report on the durations of recently completed tasks was preferred to asking them to speculate on future tasks, primarily to obtain higher accuracy time estimates and improve the realism of the task set that was generated. As noted earlier, forecasting task durations can lead to well-known biases in time estimation [41, 42, 49]. For example, when tasks are easy (as might be expected for short, simple tasks, such as microtasks), people have been shown to overestimate future task durations [9, 27]. That being said, collecting task titles from workers retrospectively

created an additional challenge for learning and analysis. Many of the tasks were written in past tense (e.g., "cooked dinner" rather than "cook dinner") rather than the future tense that would typically be observed in task management applications (denoting forthcoming to-dos). To address this, before performing any additional analysis we converted all task titles to future tense using the lemmatization functionality from the Stanford CoreNLP toolkit [58].

In total, we obtained 8,003 tasks from 753 workers. We reviewed the data manually and removed 367 tasks that were noise (e.g., garbled titles, single character titles, copy-paste text such as code snippets). This left 7,636 tasks from 729 workers for use in training and testing the microtask detection models. There were an average of 10.47 tasks per worker (median=10, standard deviation (*SD*)=5.62). The mean average confidence score in the estimates was 7.83 (median=8, *SD*=2.26) with 90.7% of tasks with the worker-provided confidence score (signifying their confidence in the duration estimate) of five or above. To ensure a high-quality dataset for training and testing our machine-learned models, we further filtered the data to the 5,699 tasks (74.6% of the dataset) that: (1) appeared at least once in the anonymized log data from the Wunderlist task management application (http://www.wunderlist.com, acquired by Microsoft in 2015), as described in the next subsection (to help ensure they were not bogus and were representative of typical tasks in a task management application); (2) had a confidence score of ≥ 5 (so we could trust the estimate), and; (3) had a duration (d) of ≥ 30 seconds (to reduce erroneous durations). 82.7% of tasks in the crowdsourced set were matched against the log data. The correlation between the crowdsourced tasks and the logs was moderate (Spearman's ρ (ties-corrected) = 0.639, $p < 0.001$). The mean average task duration was 17,942s (median=1,800s, *SD*=549,443s), grossly skewed by a few longer-term (year long) tasks such as "tour the world," "write a book," and "quit smoking." The definition of microtask (e.g., tasks with $d \leq 5$ minutes [300s], ≤ 10 minutes [600s], or ≤ 15 minutes [900s]), clearly affects the volume of these tasks in the data: $d \leq 300$s (n=960, 16.8% of all valid tasks), $d \leq 600$s (n=1,629, 28.6%), and $d \leq 900$s (n=2,169, 38.1%). Figure 2 shows the distribution of binned task durations (per bin and cumulative) over all 5,699 tasks in the dataset. We focus on d_t=300s, as is the case for much of our analysis.[2]

3.1.2 Popular Microtasks. We focus on a duration threshold (d_t) of 300s (five minutes or less) for microtasks for this study. Given that d_t=300s tasks were the shortest tasks in the dataset, we believed that they were likely to be the most clearly "micro" in nature. Figure 3 shows a list of popular tasks (with 20 or more instances in the labeled dataset), ranked by the percentage of task instances labeled by crowdworkers as microtasks per our definition. We applied the lemmatization functionality from the Stanford CoreNLP toolkit and removed three common stopwords ("a," "the," and "do") to further collapse synonyms.

The tasks on the far left of the figure (high percentage of microtasks) are generally short, mundane to-do tasks (e.g., "feed dogs," "sweep kitchen"). Some of the tasks with a high percentage of microtasks may appear surprising, but occur frequently in task management applications. For example, although the presence of "brush tooth" (lemmatized version of "brush teeth") as a common microtask may appear unusual for those who have developed a regular tooth brushing habit, one recent study found that over 20% of adults brush their teeth less than twice per day [29]; digital reminders can play an important role in improving oral hygiene compliance [25].

3.1.3 Historic Attributes. We also analyzed the relationship between whether the task was a microtask and crowdworker responses regarding (a) their certainty in the time estimate provided, and (b) whether the task had been done before and whether it was was something that workers did

[2]For completeness, later in the article, we report microtask detection performance for different values of d_t, ranging from 60s to 900s, inclusive.

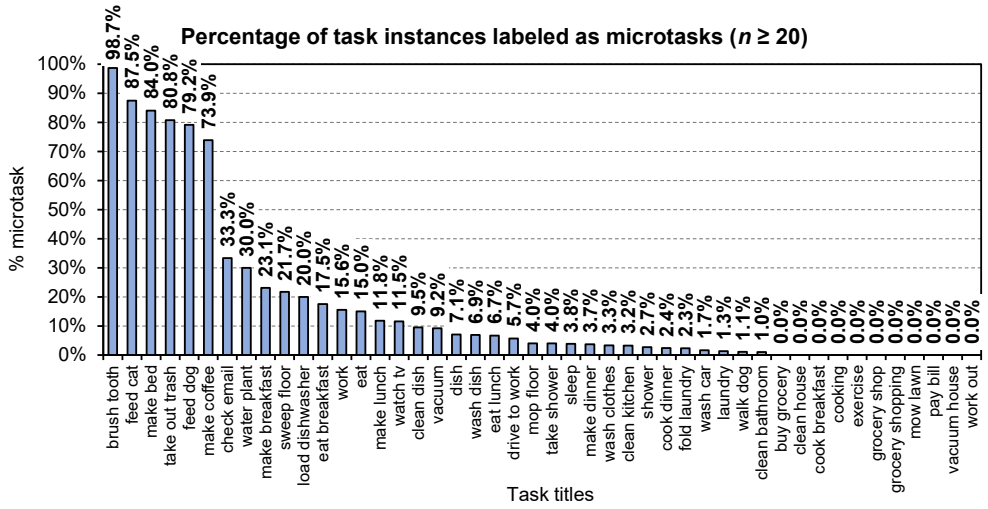
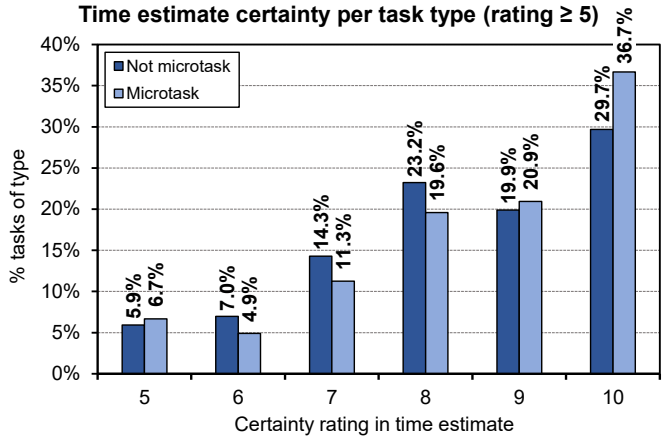


Fig. 3. Titles of the tasks in the labeled dataset which occur 20 or more times, with associated percentage of task instances that are labeled as microtasks (5 minutes or less, $d_t \leq 300$s). Note that (a) the task titles have been lemmatized, (b) these are not the full set of tasks in our crowdsourced dataset, and (c) these tasks were extracted from the crowdsourced data *after* joining with the anonymized Wunderlist dataset, so, although the high percentage of microtask labels for some tasks are surprising (e.g., "brush tooth," the lemmatized version of "brush teeth") they are representative of some of the types of tasks that people add to task management applications.

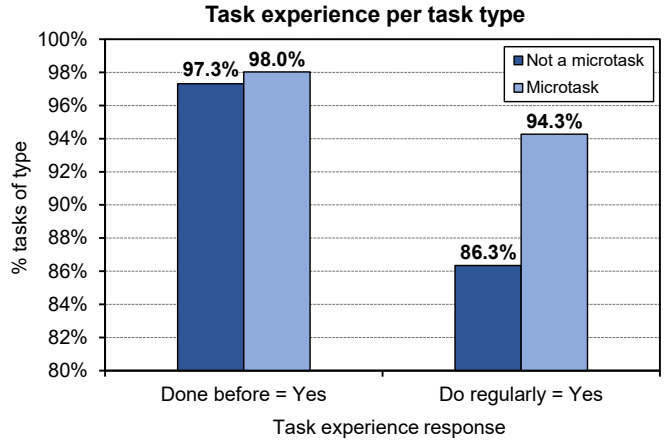
regularly. Figure 4 summarizes the findings. In particular, workers were more certain about their time estimates when they provided microtasks (e.g., 36.7% of microtasks had a rating of 10 (most certain) versus 29.7% of non-microtasks: Z=4.27, $p < 0.001$), microtasks and non-microtasks are equally likely to have been done before (Z=1.25, p=0.21), and microtasks are much more likely to be done regularly (Z=6.81, $p < 0.001$).

The changes in the responses depending on whether a task was a microtask suggest that task experiences may be valuable classification features for microtask detection. These are signals that a system could conceivably solicit from users or infer from their task creation and completion activities. We experiment with these features in the learning experiments later in the article.

3.1.4 Log Data. In addition to the crowdsourced data that was used to provide ground truth for training the models, we also used a sample of anonymized data from Wunderlist to generate additional aggregate features for each of the tasks in the labeled dataset (grouped based on task title). The Wunderlist task management application supported the creation of to-do lists and tasks, marking tasks as complete, delegating the tasks to others, and assigning additional metadata such as due dates, to the tasks. Event data, e.g., creation and completion times, were also recorded. The anonymized dataset comprised tasks, lists, and metadata from a large random sample of users in the en-US geographic locale. The mean average task title length was 2.42 tokens (median=2, SD=1.78) and 14.53 characters (median=12, SD=10.08). In total, 88.0% of tasks were marked by users as complete and 7.9% had a user-generated due date by when the task was required to be completed. Prior to analysis, the data was analyzed for personally identifiable information (PII), including person name and digits (e.g., in phone numbers), which was replaced by randomly-chosen alternatives. PII scrubbing was done using proprietary algorithms prior to analysis or modeling.



(a) Self-reported certainty in time estimates for microtasks and non-microtasks.



(b) Self-reported task experience for microtasks and non-microtasks.

Fig. 4. Distributions of microtasks and non-microtasks for crowdworker responses to questions about their certainty in their time estimations and their prior experience with the tasks they provided.

3.2 Model

We trained a machine-learned classifier to estimate whether a given task was a microtask using the text of the task and additional features such as aggregated features from user log data. We lacked sufficient labeled data to apply deep learning. Instead, since this is, to our knowledge, the first foray into the area of microtask detection, we favored a simple approach as a first step. To this end, we used logistic regression (LR) as our learning algorithm. LR is simple, interpretable, and performant; and has been used successfully for a variety of similar research studies including inferring: task intent [17], task attributes [83], and task state [84]. We compare the performance of the LR model against other machine-learned models, including a BERT baseline [22]. All features were normalized to the range [0,1]. We experimented with L1 and L2 regularization, both separately and in combination (using elastic net [92]), to help prevent model overfitting. L1-only performed best and is used for the remaining microtask detection experiments.

Table 1. Features used in the microtask detection model, grouped by feature class. Log data features are aggregated based on the task title (lowercased and whitespace pruned). Log data features marked with * are computed for all tasks. Other log data features are only computed for tasks with log frequency ≥ 10.

Feature class	Feature name	Description
Task text	WordNGrams	Word *n*-grams (up to three tokens in length) and associated counts
	CharNGrams	Character *n*-grams (up to three characters in length) and associated counts
	TaskLengthTokens	Length of the task in tokens
	TaskLengthChars	Length of the task in characters
	NumGenericLocations	Number of generic locations ("home," "work," "gym," etc.)
	NumActionVerbs	Number of action verbs
Log data (aggregated by task title)	DayOfWeekCreated	Mean average day of week when task created {Sun=0, Mon=1, …, Sat=6}
	DayOfWeekCompleted	Mean average day of week when task completed
	AvgCreationToCompletion	Mean average time difference (days) between created and completed
	StDevCreationToCompletion	Standard deviation time difference (days) between created and completed
	MedCreationToCompletion	Median time difference (days) between created and completed
	AvgCreationToDueDate	Mean average time difference (days) between created and due date
	StDevCreationToDueDate	Standard deviation time difference (days) between created and due date
	MedCreationToDueDate	Median time difference (days) between created and due date
	AvgCompletionToDueDate	Mean average time difference (days) between completed and due date
	StDevCompletionToDueDate	Standard deviation time difference (days) between completed and due date
	MedCompletionToDueDate	Median time difference (days) between completed and due date
	%Complete	Fraction of task instances that are marked complete
	%DueDate	Fraction of task instances that have a due date
	%Assigned	Fraction of task instances that are assigned to another individual
	%CompletedCreator	Fraction of task instances completed by creator
	%DeletedCreator	Fraction of task instances deleted by creator
	NumTasks*	Number of instances of the task
	NumUsers*	Number of users who created the task
	NumLists*	Number of lists containing the task
	%UniqueListNames	Fraction of list names that are unique
	AvgTaskRevision	Mean average task revision (first version revision number = 1)
	StDevTaskRevision	Standard deviation task revision
	MedTaskRevision	Median task revision
	AvgListLenTokens	Mean average list title length in tokens
	StDevListLenTokens	Standard deviation list title length in tokens
	AvgListLenChars	Mean average list title length in characters
	StDevListLenChars	Standard deviation list title length in characters
	CreationDayEntropy	Information entropy [72] of the day of week *created* distribution
	CreationWeekEntropy	Information entropy of the week of year *created* distribution
	CreationMonthEntropy	Information entropy of the month of year *created* distribution
	CompletionDayEntropy	Information entropy of the day of week marked *completed* distribution
	CompletionWeekEntropy	Information entropy of the week of year marked *completed* distribution
	CompletionMonthEntropy	Information entropy of the month of year marked *completed* distribution
Semantic embeddings	Sent2Vec	DSSM [38] embeddings (see Section 3.3.3)
	Task2Vec	Task embeddings (see Section 3.3.3)

Using the labeled data from Section 3.1.1 assumes that the model is run on actionable tasks. As we show later, actionable tasks are only about 30% of to-do tasks in the dataset used for this study. Most to-do tasks are in fact non-actionable (e.g., collections of miscellaneous items such as books to read or movies to watch, notes, grocery lists, etc.). To apply the microtask detector in practice, we may need to first identify actionable tasks. We explore this in Section 4.

3.3 Features

Three classes of features were generated for use in the learned model: (1) text of the task title (used primarily as a source of *n*-gram features), (2) historic log data, meant to capture aspects of how similar tasks were completed by other users (e.g., average timespan from creation to completion for tasks with same title (a duration proxy), percentage of those tasks with a due date, etc.), and (3) semantic embeddings meant to represent aspects of task meaning.

3.3.1 Task Text. We extract several features from the text of the task titles, including word and character *n*-grams up to three tokens and characters in length and the length of the title (in tokens and characters). We also count the number of action verbs (drawn from an online dictionary[3]) and the number of generic locations (home, work, office, etc.) appearing in the task titles. These counts reveal some additional characteristics of the task, specifically how actionable and unconstrained it is (i.e., whether it involves some direct action and whether it must happen at a known location).

3.3.2 Log Data. We also computed a variety of features from the aggregate log data. We compute frequency features (marked with * in Table 1) for all tasks. To help ensure the aggregates were reliable, we compute other aggregate statistics for tasks appearing at least 10 times in the log data. We joined the logs with the labeled dataset based on task title, after lowercasing and pruning surplus whitespace. Given the large number of users involved in the calculations of temporal features such as *AvgCreationToCompletion*, the log-based features may be less subject to the individual biases in time estimation noted earlier. For the tasks occurring fewer than 10 times, the aggregate features are left blank. Log data features included the day of the week that these tasks were created and completed, timespans between creation and completion, the fraction of tasks that were completed, had a due date, etc., as well as other characteristics such as the entropy of the creation time and completion time distributions (meant to quantify temporal variance – tasks with higher entropy might be more sporadic and hence less likely to be microtasks) and features of the lists containing those tasks.

3.3.3 Semantic Embeddings. We compute semantic embeddings for the text of the task title. We use two approaches: (1) Sent2Vec: a deep structured semantic model (DSSM) [38], for representing text strings in a continuous semantic space (with publicly available embeddings)[4], and (2) Task2Vec: a featurizer spliced from a larger neural network model trained to predict task-list membership (e.g., is "buy milk" more likely to belong to "grocery" than "work"?) from the log data. The featurizer builds on FastText embeddings [8] to encode salient words in the textual representation of tasks. In particular, embeddings are assigned to each subword character trigram; a word's representation is then obtained by a mean pooling operation of the trigram embeddings. Once each word's representation is obtained, the encoding for the entire task is generated by running a convolutional neural network (CNN) over the word sequence, followed by a max pooling operation. The Task2Vec featurizer has the following hyperparameter settings. Task titles are padded or truncated as appropriate to 12 tokens per task, and 20 characters per token. The trigram embedding length is set to 80, the CNN's kernel width to 3 and its encoding size to 320. Each of the embeddings becomes a set of features in the model.

Table 1 lists all features in the LR model, by class. There were a few thousand features in total for each task, including semantic embeddings. All features are concatenated together for each task.

3.4 Experimental Setup

We ran the LR model on the labeled dataset. We applied five-fold cross validation over 10 runs. The data was stratified so that a crowdworker and their tasks only ever appeared in the training folds or the test fold, but not in both. This is a stricter test of model generalizability than ignoring the user. We select area under the precision-recall curve (AUPRC) as the main performance metric. AUPRC works well with imbalanced data (as in our dataset, which is 16.8% positive) and focuses more on positives [21].[5] For completeness, we also report area under the receiver operator characteristic

[3]https://www.wordexample.com/list/most-common-verbs-english/
[4]https://www.microsoft.com/en-us/research/project/dssm/
[5]Positives are of more interest in this study given application scenarios such as Figure 1, which suggests positives in a highly-visible way and therefore needs to be accurate.

Table 2. Mean average microtask detection results (n=5,699) for the *All* feature set (logistic regression model, LR). Also included for reference are the results of other models: support vector machine (SVM), multiple additive regression trees (MART), and averaged perceptron (AP).

Model	AUPRC	AUROC	Accuracy	PosPrec	PosRec
LR (*All*)	0.645	0.849	0.880	0.745	0.467
SVM	0.614	0.832	0.871	0.770	0.381
MART	0.604	0.844	0.876	0.696	0.473
AP	0.614	0.835	0.847	0.577	0.570

curve (AUROC) (which is insensitive to the marginal), as well as accuracy, positive precision, and positive recall (the last three metrics use a standard classification threshold, p=0.5).

3.5 Results

We now present the results from our experiments on the microtask detection models, starting with the model trained on all features listed in Table 1.

3.5.1 All Features. The first experiment involves the full LR model with all of the features (referred to hereafter as *All*). The results are shown in Table 2, alongside the results of other models for reference (support vector machines (SVM) [18], multiple additive regression trees (MART) [28], averaged perceptron (AP) [15]). The results show that the LR model significantly outperforms the other methods on the same set of features (*All*) (e.g., for AUPRC: paired t-tests: $t(49)$=4.93, $p < 0.001$). The performance of this model is also depicted visually in the uppermost (blue) precision-recall (PR) curve in Figure 6 (with precision at 0.8 and recall at 0.4). A marginal classifier that always predicts "is a microtask" has an AUPRC of 0.168. The mean average model AUPRC (0.645), in the top row of Table 2, well exceeds that value.[6] Given our focus on positives, it is also promising that positive precision is fairly high: when the LR model predicts positive, it is correct 75% of the time. We also experimented with feature selection but did not see an improvement in performance as a result, perhaps given the L1 regularization, which has been shown to be an effective substitute [64].

3.5.2 Varying Duration Threshold. Looking beyond d_t=300s, we also examine the effect on classifier performance of different duration thresholds. We swept d_t from 60s (1 minute) to 900s (15 minutes) at 60-second increments and computed the classification performance of the LR model at each value of d_t. Distributional differences in positives between the different sets (e.g., 2.5% positive for d_t=60s versus 38.1% positive for d_t=900s) make it challenging to compare the AUPRCs directly. AUROC is unaffected by such distributional differences, so a fairer comparison is possible. Figure 5 shows the AUROC values across the range of d_t values that were tested. The results show that AUROC for d_t=300s is higher than the other thresholds. We focus on d_t=300s for the remainder of this article, given this better performance, and that it better matches some of our envisaged application scenarios (e.g., Figure 1) and prior work [40].

3.5.3 Feature Class Ablations. Beyond the overall performance of the model, we were also interested in the contributions of the various feature classes. To this end, we experimented with ablating different feature classes and compute the performance of the LR model thereafter. We report the performance metrics in Table 3 and illustrate the impact of these changes in Figure 6.

We compared the different ablations to the full LR model (*All*) based on AUPRC (our primary performance metric). The differences with *All* were significantly lower (paired t-tests: all $t(49)$

[6]Note that all LR models in the article significantly outperform the always-positive marginal baselines at $p < 0.001$.

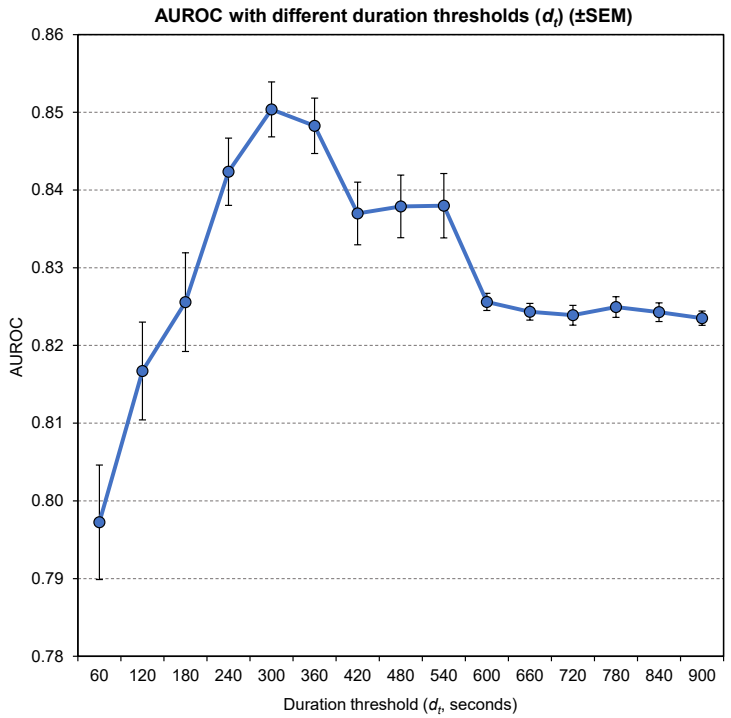


Fig. 5. AUROC of the *All* model for different d_t ranging from 60s to 900s inclusive, at 60s increments. Error bars denote standard error of the mean (SEM) across all folds and runs. The chart shows that d_t=300s has the highest AUROC of the d_t values tested.

Table 3. Mean average feature class ablation results (E=Embeddings, L=Log data) for microtask detection (n=5,699). The full LR model (*All*) is in the first row as starting point for the feature class ablations. The percentage difference from *All* is shown for each of the metrics.

Model	AUPRC	AUROC	Accuracy	PosPrec	PosRec
All	0.645	0.849	0.880	0.745	0.467
−E	0.625	0.840	0.877	0.684	0.507
	−3.07%	−1.06%	−0.34%	−8.30%	+8.51%
−L	0.609	0.835	0.875	0.670	0.513
	−5.55%	−1.69%	−0.59%	−10.09%	+9.85%
−(E,L)	0.582	0.827	0.874	0.660	0.530
	−9.72%	−2.65%	−0.62%	−11.45%	+13.37%

≤ -2.73, all $p < 0.01$). Removing the embeddings does not have as much effect as removing the log data. One explanation is that the aggregate user activity more directly captures typical task durations (e.g., the average timespan from creation to completion, which is available to the model as a feature) than task title semantics, which may have a more indirect relationship with task duration. Turning attention to the PR curves in Figure 6, we see that the differences between model variants mainly lie in the upper left of the curves (at precision ≥ 0.7 and recall ≤ 0.5, where the

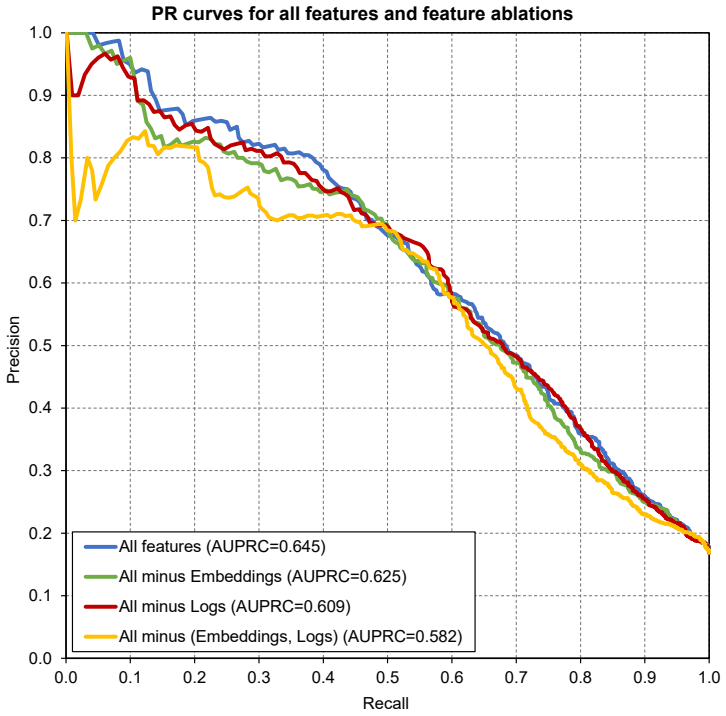


Fig. 6. Precision-recall curves from cross-validation for microtask detection using the full feature set (*All*) and select feature ablations, dropping classes of features and re-computing the performance of the classifier.

model probabilities are around p=0.5 or above). The model still performs well even with just basic text features (primarily word and character n-grams), which do not require additional data sources such as the semantic embeddings, are inexpensive to implement, and run efficiently at scale.

3.5.4 Individual Feature Contributions. Focusing on the feature classes masks the relative importance of some of the individual features in the model, which can offer additional insight into key associations between specific features and microtasks in the data. Since the model is dominated by the n-gram features, we split this analysis into two parts: (1) n-gram features and (2) log-based features. Table 4 shows the top-ranked features for each of the two feature groups, ranked by absolute feature weight. Note that we do not include the embeddings features in this analysis since it is challenging (if not impossible) to ascribe meaningful explanations to specific elements in the semantic embedding vectors.

The findings of this analysis in Table 4 illustrate the importance of the n-gram features, which carry much more weight in general than the log-based features. Most of the top n-grams have a positive association with microtasks and most are self-explanatory.[7] Turning our attention to the log-based features, there was a negative association between the timespan between task creation and task completion (i.e., *AvgCreationToCompletion*, *MedCreationToCompletion*) and microtasks. It seems reasonable to assume that tasks that take longer to be marked complete are less likely to be microtasks. The specification of locations (home, gym, etc.) (i.e., *NumGenericLocations*) is also negatively associated with microtasks, perhaps because of the additional time involved

[7]The unigram "check" is usually found in short tasks such as "check mail" (likely postal mail not electronic mail).

Table 4. Most important features in the LR model ranked by absolute evidential weight, grouped by (1) (lemmatized) *n*-grams and (2) log-based features. Feature weights are normalized w.r.t. the top-ranked feature in the model (the unigram "garbage"). The table also includes the sign for each feature to indicate the directionality of the relationship between the feature value and the microtask label.

(a) Top 10 *n*-gram features

Feature	Weight	Sign
"garbage"	1.000	+
"tooth"	0.754	+
"recycle"	0.678	+
"make bed"	0.643	+
"trash"	0.568	+
"mail"	0.536	+
"clean bathroom"	0.529	−
"feed cat"	0.515	+
"check"	0.511	+
"shopping"	0.492	−

(b) Top 10 log-based features

Feature	Weight	Sign
AvgCreationToCompletion	0.326	−
NumGenericLocations	0.278	−
MedCreationToCompletion	0.186	−
NumActionVerbs	0.168	+
%DueDate	0.154	+
%CompletedCreator	0.112	+
CompletionDayEntropy	0.098	−
%UniqueListNames	0.081	+
AvgListLenTokens	0.078	−
NumUsers	0.029	−

or perhaps because there are less likely to be microtasks in certain locations (e.g., as Figure 3 shows, exercising is never labeled as a microtask in our data). Next is a collection of features, all focused on action/completion that have a positive association with microtasks (i.e., *NumActionVerbs*, *%DueDate*, *%CompletedCreator*). Microtasks typically involve clear actions and are easily doable. Interestingly, the fraction of the tasks that are completed in general (*%Complete*) is less associated with microtasks than completed by creator (*%CompletedCreator*). Given their simplicity, microtasks may be more likely to be completed by the person who created the tasks rather than being delegated to others, which would be included in the *%Complete* feature. Other salient features, albeit with lower normalized weights (< 0.1), include the entropy of the distribution of days on which the task was marked as complete (which captures aspects of task irregularity and is negatively associated with microtasks, as hypothesized in Section 3.3.2) and properties of the lists to which the tasks belong: *%UniqueListNames* (higher uniqueness means more consistency in the list names associated with a task) which is positively associated with positive labels, and *AvgListLenTokens* (the average number of tokens in the titles of the lists containing the task), which is negatively associated with positive labels. Task popularity (*NumUsers*) is negatively associated with positive labels, perhaps because most tasks are not microtasks, including the most common tasks in the labeled dataset (the top three tasks by frequency are "do laundry," "wash dishes," and "grocery shopping").

3.5.5 BERT Baseline. We also compared our approach to a BERT model [22] for microtask detection. Specifically, we use pre-trained BERT [88] – which consists of 12 transformer layers, 12 attention heads, and a hidden output dimension size of 768 – and add a final linear layer for classification. This model is then fine-tuned on our dataset of microtasks. The input to the BERT-based model is the text snippet of the task, and the output is the probability under the model that this is a microtask. We use the same five-fold cross-validation over 10 runs to train and validate the model. Training is conducted for each experiment over 10 epochs using Adam [47] with an initial learning rate of 5e-5; the loss function is cross-entropy and the batch size for training is set to 48.

The AUPRC of this BERT-only model is 0.552, significantly lower than the 0.645 reported for the LR model in Table 2 (paired t-tests: $t(49)=9.54$, $p < 0.001$) and similar to the performance of the LR model with task text features only (AUPRC=0.582, $t(49)=1.73$, $p=0.06$). We also ran experiments that combined a BERT-based transformer for the textual input of a task, with a fully-connected layer for

encoding its other numeric attributes (i.e., Log + Semantic Embedding features) (hereafter called "BERT-plus"). However, BERT-plus performed even worse than the BERT-only model (AUPRC of full model: 0.488 vs. AUPRC of BERT baseline: 0.552, paired t-tests: $t(49)=-8.49$, $p < 0.001$).

We hypothesize that there are a few reasons that these neural approaches perform worse than a simple linear model. First, due to the limited size of our dataset – further exacerbated by performing experiments across five folds – we do not have enough training data to tune the large parameter space of the transformer based approaches. This is also likely why the neural model that additionally considers the numeric features (BERT-plus) performs worse than the BERT-only model. While the latter is pre-trained, the former adds over 100,000 additional randomly initialized parameters, with only about 4,000 training instances on which to set them appropriately. It is thus unsurprising that a simple linear model with few parameters is able to outperform more complex approaches on predictive generalization. Additionally, BERT – with its masked language and next-sentence objectives – is designed and trained to act as a language model on sentence or paragraph lengths of text. It is unclear that such a model would provide a clear representational advantage for short snippets of text, such as to-do tasks, which are often just 1 or 2 words.

3.5.6 Applying Historic Features. As noted earlier, there were some differences in crowdworker responses to questions about whether a task had been done before and whether it was done regularly, depending on whether a task was a microtask. While these features may be difficult to obtain in practice (a system would need to have access to a user's task history and timing information about completion events), we still wanted to understand whether including these features in the LR model had any affect on microtask detection performance. We added the two binary features and re-ran the experimental pipeline (five-fold cross validation over 10 runs). The results show a significant improvement in the performance of the LR model when the two historic features are added: AUPRC=0.673 with historic features included versus AUPRC=0.645 without those features (paired t-tests: $t(49)=5.64$, $p < 0.001$). This is certainly promising and suggests that personalization may be beneficial. Additional related features (e.g., considering the frequency with which the task is performed by the user, its periodicity, etc.) could further enhance performance. More experiments are needed on the value of personal and contextual features, such as where the task typically gets completed, the time of day and day of week when it gets done, and the relative sequencing with respect to other tasks.

3.5.7 Summary. The results show the strong performance of our machine-learned model for automatically detecting microtasks. Feature analysis shows the importance of the behavioral features and the strength of the model based on textual features only. We also show that our model outperforms a BERT baseline and that there is some promise in integrating features related to the user's task history. We now turn our attention to another issue of scaling up microtask detection. Although our findings have demonstrated good performance, it is only for actionable tasks. Since not all tasks meet this requirement in practice, we may need an additional classifier to determine when to apply microtask detection.

4 ACTIONABLE TASK PRECLASSIFICATION

All of the tasks used to train and test the microtask detection had an associated duration provided by crowdworkers (Section 3.1.1). In practice, not all tasks in task management applications, digital assistants, and so on will be conducive to duration estimation. For example, tasks could be items in a collection of movies to watch or books to read, notes (e.g., names or addresses), or constituents in a list that itself comprises a task (e.g., food items in a grocery list). Running microtask detection on these tasks could lead to erroneous microtasks being detected and flagged to users. To help address this potential issue, we developed a task preclassifier to determine which tasks were *actionable*

(i.e., tasks that need to be acted on and are therefore conducive to duration estimation). This classifier could be run before the microtask detector as a prefilter. It could also be integrated into the microtask detection model as an additional feature. We experiment with both options in Section 5, but first, we describe the preclassifier and present an evaluation of its performance.

4.1 Labeled Dataset

To build a labeled dataset for training and testing, we created a human judgment task. Crowdworkers in the en-US market annotated tasks with whether they were actionable (suitable for duration estimation). Task lists were randomly selected from an anonymized and aggregated version of the logs from the Wunderlist task management application described earlier.

Each worker was presented with 100 lists and their associated task items. Workers were asked to decide whether the list itself or any of its items required duration estimation. "Yes"/"No"/"Don't know" labels were collected on each list and on each item. Workers were asked to mark "Don't know" if they did not understand the task (e.g., if it was non-English, garbled text, single character, etc.) or if they could not determine whether or not duration estimation was appropriate for the task. Each instance (list title and list items) was labeled by three independent workers. The "Yes," "No," and "Don't know" labels were distributed as 37.5%, 49.8%, and 12.7% respectively. The final label for each item was decided based on the majority vote. The "Don't know" labels were counted as "No" labels since the intended application was to detect actionable tasks vs. everything else. The Cohen's Kappa score (κ) was 0.59 (signifying "moderate" agreement [53] between workers) and the inter-annotator agreement (all three workers agree) was 85.0% over the labeled instances. The process resulted in a set of 549 task instances that was used to train and test the preclassification model. In this dataset, 28.6% of the tasks were labeled as actionable and 71.4% of tasks were non-actionable. Since the microtask detector was trained only on actionable tasks, it is likely to underperform when encountering non-actionable tasks. The frequency of these non-actionable tasks illustrates the important role that an actionable task preclassifier could play in microtask detection.

4.2 Preclassification

4.2.1 Model. We experimented with several learning algorithms for the actionable task preclassifier: namely, LR, SVM, MART, and AP (the same four algorithms as used earlier). As shown in Table 5, the results on the different algorithms were similar. Differences between the algorithms were not significant (e.g., for AUPRC, all $p \geq 0.429$). We settled on using LR because of its strong performance, simplicity, and interpretability.

4.2.2 Features. The preclassification model used character and word n-grams from task/list titles (up to trigrams) and associated counts. We normalized features to the range [0,1]. We used grid search to set hyperparameters (e.g., L1 and L2 regularization). We experimented with additional features using Sent2Vec [38] and GloVe [67], fine tuned on the vocabulary of our dataset, but did not observe any significant improvement in preclassification performance.

4.2.3 Results. We ran the model over the labeled dataset (in Section 4.1), applying five-fold cross validation. The performance metrics, averaged across all five folds, are provided in Table 5. The results are strong and clearly well above an always-positive marginal model that always predicts "actionable," which has an AUPRC of 0.286. The top weighted n-gram features in the LR model are verbs such as "clean," "go," "buy," "get," and "plan" – all of which are clearly connected to action. Given that the model has strong classification performance, the next step was to use the preclassifier alongside the microtask detector to assess the impact on microtask detection performance.

Table 5. Mean average actionable task preclassifier metric results, based on five-fold cross-validation on the labeled dataset (from Section 4.1) (n=529). Included are the results for logistic regression (LR), support vector machine (SVM), multiple additive regression trees (MART), and averaged perceptron (AP).

Model	AUPRC	AUROC	Accuracy	PosPrec	PosRec
LR	0.729	0.868	0.835	0.738	0.660
SVM	0.732	0.869	0.834	0.718	0.694
MART	0.728	0.869	0.837	0.738	0.674
AP	0.729	0.871	0.840	0.756	0.669

5 COMBINING CLASSIFIERS

We wanted to understand how microtask detection would perform on an unfiltered set of tasks; a similar scenario to what it would encounter in production. We used a similar approach to Section 4.1 to create a separate test set of tasks using the anonymized and aggregated log data. Unlike the dataset in Section 3, we selected all tasks, meaning that most will likely be non-actionable.

5.1 Labeled Dataset

We constructed a labeled dataset focused on microtasks. We refer to this as the "held out set" and use it for testing. Access to the sample of common to-do tasks in the anonymized and aggregated log data was strictly limited to the project team. We could not share the data with external judges for labeling purposes. Two of the authors labeled a set of tasks pulled from 100 randomly-selected lists from the aggregated log data. The tasks were labeled with a "Yes" or "No" depending on whether the judge believed that they were microtasks (defined as whether the task could be completed in five minutes or less, ignoring contextual constraints). In these lists, there were 325 tasks in total. The initial Cohen's Kappa value was strong (κ=0.85), signifying "almost perfect" agreement between the judges [53] and the inter-annotator agreement was 97.2% over the labeled instances. Disagreements were resolved based on discussion between the two judges to settle on a final rating for each task. In the end, 11.4% of tasks (n=37) in the held out set were labeled as microtasks. Examples of microtask titles include "call dentist," "pay phone bill," and "schedule budget meeting."

5.2 Model Performance

We use the data in the previous subsection to evaluate the performance of the microtask detection. Since this applies the model across a full range of to-do tasks, this is a more realistic setting than the evaluation described in Section 3. We report on the results of two experiments: (1) performing microtask detection on all 325 tasks in the held out dataset, and (2) applying preclassification, either as a prefilter or as a feature in the microtask detection model.

5.2.1 Microtask Detection Only. We train the microtask detector on the full set of data from Section 3.1. The top row in Table 6 contains the results from just applying the microtask detection on all tasks in the held out set. The red line in Figure 7 reports the performance (AUPRC=0.229). While the values exceed an always-positive marginal baseline (AUPRC=0.114), the poor performance is a barrier to deploying microtask detection in practice. In examining the classification errors, many of the false positives were non-actionable tasks such as items to purchase (e.g., "bananas," "bandaids") (most of the errors), or had an unclear intent (e.g., "mail," "healthcare").

5.2.2 Applying Preclassification. We can use the preclassifier output to adjust the performance profile of the microtask detector. As a first experiment, we apply actionable task preclassification

Table 6. Microtask detection results on held out dataset with and without preclassification, used as either a filter (Preclassification [FILTER]) or as a feature (Preclassification [FEATURE]) (n=325).

Model (*All* with)	AUPRC	AUROC	Acc	PosPrec	PosRec
No preclassification	0.233	0.612	0.868	0.364	0.216
Preclassification [FILTER]	0.650	0.932	0.902	1.000	0.135
Preclassification [FEATURE]	0.268	0.671	0.889	0.556	0.135

as a prefilter, then apply the microtask detection on the tasks that make it through. We train the preclassifier on the full labeled dataset from Section 4.1, which is separate from the held out set, and pick a model probability that maximizes the F1 score on that dataset. That probability is then used as a decision boundary (prefilter) to determine whether to pass the task to the microtask detection. This enables the two models to be fully decoupled so that they can be applied in different settings (e.g., understanding whether a task is actionable has many applications beyond microtask detection, including simply being able to auto-segregate tasks into actionable/non-actionable categories) and because it offers more control over and visibility into the use of the preclassifier. This comes at the expense of the potential for compounding errors and the need to maintain the two models and monitor their performance, both separately and together. The preclassifier probability can also be integrated as a feature into the microtask detection model. We experimented with both options and report results in this section.

Filter: The results of applying the preclassifier are shown in the middle row of Table 6 and in Figure 7. The blue line in the figure has the PR curve for microtask detection with preclassification applied as a prefilter, resulting a much stronger performance profile (70% precision at 35% recall). The results are clearly better when the preclassifier is applied (a 179% increase in AUPRC [0.650 versus 0.233]). We applied bootstrap sampling [24] (10 samples, 100 cases per sample) and paired t-tests to compare the AUPRCs of the model with and without the preclassification prefilter. The differences are statistically significant ($t(9)$=11.04, $p < 0.001$). The gains from the preclassifier as a prefilter for the microtask detector are clearly apparent.

Feature: We also experimented with using preclassifier score as a feature in a combined model (i.e., apply the preclassifier to every example in the dataset from Section 3, generate a feature for that score, and then retrain the microtask detector with that additional feature included). The precision-recall curve is shown in Figure 7 (dashed gray line) and the metrics appear in the last row in Table 6. The performance gain over the model with no preclassification was minimal (AUPRC=0.268) and not statistically significant (with bootstrap sampling [10 samples, 100 cases per sample]: $t(9)$=0.88, p=0.40).[8] One explanation is that the preclassifier feature weight is being lessened by contributions from and interactions with other features. Inspecting the weights in the model offers some support for this explanation: the evidential weight for the preclassifier feature is almost zero and around the 40th percentile of all features. There may simply be too many n-gram features in the model capturing different aspects of the tasks for the preclassifier feature to have a significant impact. Applying the feature as a filter enables us to isolate the preclassifier feature and increase its impact. The effect on classification performance is immediately apparent from inspection of Figure 7.[9]

[8]This was true even when we experimented with alternative models (e.g., SVM, MART, AP), which we might expect to branch at the preclassifier feature if it was an important discriminator, as well as feature selection and re-weighting to boost the preclassifier contribution to the microtask detection model.

[9]Note that although the positive recall reported in Table 6 may appear low (0.135), as noted earlier, the accuracy, positive precision, and positive recall values are based on a standard classification threshold of p=0.5. Careful selection of this

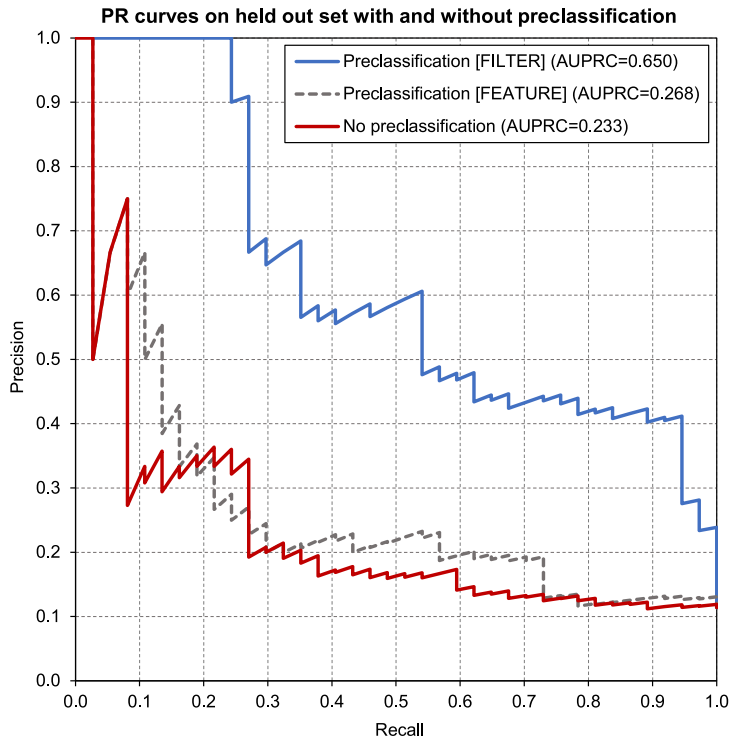


Fig. 7. Precision-recall curve for microtask detection on held out dataset, with and without preclassification. For preclassification, we also report the performance with the preclassifier applied as a feature (dashed gray line) and as a filter (solid blue line).

5.2.3 Summary. Overall, the results reported in this section are promising and demonstrate the value of the preclassifier for the microtask detection task. More research is needed on different ways to integrate the preclassifier, including even deeper explorations of feature selection and feature re-weighting methods to boost its contributions to the combined model.

6 DISCUSSION AND IMPLICATIONS

The study has shown that it is feasible to predict microtasks from task lists. Microtasks are fairly common: 11.4% of tasks in our sample drawn from a popular task management application are microtasks (with a five-minute threshold and more if we increase the threshold). The performance of the microtask detection model on the set of actionable tasks was strong. We expect further improvements in performance as we experiment with other learning algorithms and additional features. The semantic embeddings did not appear to add much predictive value; much of the task meaning may already be captured in the *n*-grams. The features mined from usage logs had more direct proxies for actionable and/or duration (e.g., task has due date, time to mark as complete), and appeared to be more useful. Focusing on text-based features (*n*-grams, etc.) only still yielded reasonable performance (> 90% of *All*). This is promising given that the semantic embeddings and log-based features may be unavailable or too (computationally) costly to obtain in production.

threshold is necessary for the application of the model in practice. For example, we can see from Figure 7 that by selecting a different classification threshold, a reasonable precision (around 0.7) can be attained at a recall of 0.35.

We also showed that we could detect actionable tasks, which are only a fraction (28.6%) of the tasks in our dataset, and that this is important for microtask detection. The combination of the preclassifier and the microtask detector yielded radically superior performance on a set of all (actionable+non-actionable) tasks versus applying the microtask detection alone. We show that we could find 35% of positives with 70% precision. These results are promising, but also signify the difficulty of this task and the opportunity to improve performance through more sophisticated modeling and orders of magnitude more training data.

We should acknowledge several limitations of this study. First, the evaluation of microtask detection performance on a held out set (Section 5) was based on third-party judgments of microtasks, when user judgments of their own tasks would be preferable. We did observe strong inter-annotator agreement (κ=0.85), which adds to our confidence in the labels generated. Second, we define microtasks for the bulk of this analysis as those that require $\leq$ 5 minutes to complete. While this may be intuitive, we need to experiment more with other definitions and engage with users on what they would deem to be a microtask. We did show that the detection performance with other definitions of microtask (i.e., d_t ranging from 60s to 900s) is still strong. Third, we used LR as the learning algorithm for several reasons, including limited data. Applying other methods such as deep neural networks requires considerably more training data, as was suggested by the poor performance of the BERT-based models. We need to explore how to expand the labeled dataset while still preserving user privacy. There may be signals in the log data that offer information that could be used as implicit labels, e.g., the fraction of times a task is assigned a due date could be used as a sign of whether that task and similar tasks are actionable, lists with names such as "short tasks," "5 min tasks," etc., could be reasonable sources of microtasks.[10] Task history for an individual could also be useful since repetition of to-do tasks is common: 87.7% of to-dos in our user-generated dataset (Section 3.1.1) are tasks people perform regularly (94.3% of [d_t=300s] microtasks are performed regularly). That data can be mined from historic logs of a task management application (e.g., which tasks are marked as microtasks, if available) and/or including explicit user feedback on whether inferred microtasks are accurate. We could also consider using features from external sources (e.g., knowledge bases such as Wikipedia) and characteristics of the task titles (e.g., abstract versus concrete, whether they contain verbs). Finally, as mentioned earlier, the tasks in the dataset were more focused towards home and leisure activities rather than work activities. While we believe this is sufficient for an initial investigation, if this classifier were to be deployed in an enterprise setting, we may need to retrain it using data labeled based on clear instructions requesting work-related tasks and perhaps a different cohort of crowdworkers (e.g., information workers, professionals, etc.).

Beyond demonstrating the feasibility of microtask detection, which is important in its own right, the study has the following additional implications:

- **Consider task feasiblity:** The study focuses on determining whether a given task can be completed in a short amount of time. This is an important first step, but since it focuses only on duration, it does not consider the *feasibility* of the task (e.g., can the task be done in the current context? are there any dependencies?). For example, while "take out the trash" and "brush teeth" are certainly microtasks, they can only be accomplished when someone is at home and suggesting them while a user is at work or on the go has limited utility beyond making them top of mind. Recent research has sought to understand where people perform certain types of tasks and activities [6]. By combining location with time information (e.g., as in context-aware task management systems [46, 68]), we may be able to assist users by suggesting the right tasks

[10] Although our initial explorations in the Wunderlist data of using list names revealed that such explicit depictions of task durations ("short tasks" etc.) in list names are quite rare, but may still be sufficient to train a machine-learned model.

at the right time [43]. Determining task feasibility given a context is an important direction in general, with many avenues for exploration.

- **Focus on actionable tasks:** The high fraction of tasks that are non-actionable (over 70% per our labeling) and the poor performance of the microtask detector on all tasks demonstrates the importance of the preclassifier for microtask detection. While the high fraction may be connected to the nature of the Wunderlist application, there is nothing unique about Wunderlist that would make it more likely that non-actionable tasks would be entered. Even if it was not as high as 70% in other applications, it is unlikely to be a small fraction. Decoupling the preclassifier and the microtask detector enables actionable task detection to be used independently for other activities, including deciding which tasks to recommend irrespective of time, or to segment/sort actionable and non-actionable tasks. The task management application Todoist already allows users to manually segment their tasks in this way; auto-segmentation could reduce user effort.
- **Leverage context:** Going forward, we need to explore different ways of using microtask detection, e.g., for suggestions and/or scheduling in task management applications, digital assistants, and calendaring systems – and perform user studies of these different implementations to understand when the ability to apply microtask detection inferences is maximally useful to users. Previous research has shown that surfacing reminders as a prospective memory aid increases the likelihood that tasks will be completed at a future time [12]. The example in Figure 1 illustrates how microtasks could be surfaced in a system such as a digital assistant or task management tool as a general reminder (independent of context or feasibility), but users might be susceptible to habituation effects (and ignore the reminder) if it is visible for the whole day or longer [80]. More sophisticated reminder and notification strategies, especially those that consider current and/or prospective contexts, would likely be required to realize the full benefit of microtask detection.
- **Realize potential for personalization:** The machine-learned models described in this article were all global models, with the same model applied to all users and all tasks. Individual and task differences appear in many settings, including in task management [33, 57] and in search [85]. Personalization is likely to be important as we move forward with microtask detection models, just as it is in productivity in general [54]. Our initial findings in Section 3.5.6 demonstrate some potential benefits from personalization. People have different task completion practices and preferences, and even different perceptions of what constitutes a microtask.[11] To be truly helpful to users, systems must automatically adapt to these differences and provide ways for users to customize system settings (e.g., notification policies, priorities) and the user experience.

Overall, the success of the methods presented in this article suggest that they could begin to be deployed in operational systems, especially if appropriate feedback mechanisms are in place and classification thresholds are selected carefully given the intended application scenario. Beyond flagging tasks that could be done quickly, microtask detection could also be used by systems to help to find time for tasks, to automatically set reminders given user schedules, and as metadata about tasks to support re-ordering or other manipulations of task lists directly by users. The user experience for microtasks in these systems also needs careful attention and further study with users, with questions such as when and how to surface reminders, how to long to persist reminders following their initial presentation, how to collect user feedback, and if and how to provide system explanations for any inferences made and suggestions offered.

7 CONCLUSIONS AND FUTURE WORK

We have presented methods for detecting microtasks from user-generated to-do tasks. Task and time management are key capabilities of information systems, such as digital assistants and task

[11] As is illustrated by the lack of any tasks with a percentage microtask of 100% in Figure 3.

management applications. Surfacing the right tasks to users at the right time is a central pillar of effective information access and use. Microtasks constitute a fairly large fraction (over 10%) of to-do tasks, and flagging them to users could bring considerable productivity and time utilization benefits. We showed that we can accurately detect microtasks from a set of actionable tasks and a set of all tasks when a preclassifier is used. We evaluated methods on data from a crowdwork task and from a task management application. This included comparing against BERT-based models, which underperformed the LR model that we developed and demonstrated some gains from limited personalization features. Future work involves collecting more data (including by leveraging proxy signals for microtasks, e.g., list names), integrating task histories, and developing more sophisticated methods that use deep learning. Once microtask detection is deployed in production, we can also capture implicit and explicit feedback about any microtask suggestions that are shown. Although task duration is important and has been a primary focus in this article, we also need to consider the contextual aspects of the task, including the user's schedule, location, and task dependencies, in making decisions about which tasks to recommend and when to suggest them. Looking beyond duration, task feasibility (given the task plus current and/or future context(s)), and personalization are important next frontiers in task intelligence; additional steps on the journey toward developing information systems (including elements of search systems, even if just focused on repeat tasks that users can do quickly) that are capable of helping people more effectively manage *and* accomplish their tasks.

REFERENCES

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

[2] Anne Aula, Rehan M Khan, and Zhiwei Guan. 2010. How does search behavior change as search becomes more difficult?. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 35–44.

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

[4] 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.

[5] 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 ACM SIGCHI Conference on Human Factors in Computing Systems*. 345–352.

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

[7] Yochai Benkler. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* Yale University Press.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *ACM Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[9] Marilyn G Boltz, Cara Kupperman, and Jessica Dunne. 1998. The role of learning in remembered duration. *Memory and Cognition* 26, 5 (1998), 903–921.

[10] Katriina Byström and Kalervo Järvelin. 1995. Task complexity affects information seeking and use. *Information Processing and Management* 31, 2 (1995), 191–213.

[11] Carrie J Cai, Philip J Guo, James R Glass, and Robert C Miller. 2015. Wait-learning: Leveraging wait time for second language education. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 3701–3710.

[12] Stephen J Ceci and Urie Bronfenbrenner. 1985. "Don't forget to take the cupcakes out of the oven": Prospective memory, strategic time-monitoring, and context. *Child Development* (1985), 152–164.

[13] Justin Cheng, Jaime Teevan, Shamsi T Iqbal, and Michael S Bernstein. 2015. Break it down: A comparison of macro-and microtasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 4061–4064.

[14] Lydia B Chilton and Jaime Teevan. 2011. Addressing people's information needs directly in a web search result page. In *Proceedings of the International Conference on the World Wide Web*. 27–36.

[15] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP Conference on Empirical Methods in Natural Language Processing*.

1–8.
[16] Kenneth Conley and James Carpenter. 2007. Towel: Towards an intelligent to-do list. In *Proceedings of the AAAI Spring Symposium*. 26–32.
[17] Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. In *Text Summarization Branches Out*. 43–50.
[18] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.
[19] Decio Coviello, Andrea Ichino, and Nicola Persico. 2014. Time allocation and task juggling. *American Economic Review* 104, 2 (2014), 609–623.
[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 Systems*. 2382–2393.
[21] Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of the International Conference on Machine Learning*. 233–240.
[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[23] Doug Downey, Susan Dumais, Dan Liebling, and Eric Horvitz. 2008. Understanding the relationship between searchers' queries and information goals. In *Proceedings of the ACM CIKM Conference on Information and Knowledge Management*. 449–458.
[24] Bradley Efron and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC Press.
[25] Matthew Eppright, Bhavna Shroff, Al M Best, Elvi Barcoma, and Steven J Lindauer. 2014. Influence of active reminders on oral hygiene compliance in orthodontic patients. *The Angle Orthodontist* 84, 2 (2014), 208–213.
[26] Darryl K Forsyth and Christopher DB Burt. 2008. Allocating time to future tasks: The effect of task segmentation on planning fallacy bias. *Memory and Cognition* 36, 4 (2008), 791–798.
[27] Jan A Francis-Smythe and Ivan T Robertson. 1999. On the relationship between time management and time estimation. *British Journal of Psychology* 90, 3 (1999), 333–347.
[28] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2000. Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28, 2 (2000), 337–407.
[29] Carolina Ganss, Nadine Schlueter, Susanne Preiss, and Joachim Klimek. 2009. Tooth brushing habits in uninstructed adults—frequency, technique, duration and force. *Clinical Oral Investigations* 13, 2 (2009), 203.
[30] Victor M González and Gloria Mark. 2004. Constant, constant, multi-tasking craziness: Managing multiple working spheres. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 113–120.
[31] Andrey Gusev, Nathanael Chambers, Pranav Khaitan, Divye Khilnani, Steven Bethard, and Dan Jurafsky. 2011. Using query patterns to learn the duration of events. In *Proceedings of the International Conference on Computational Semantics*. 145–154.
[32] Jacek Gwizdka. 2002. Reinventing the inbox: Supporting the management of pending tasks in email. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems EA*. 550–551.
[33] Jacek Gwizdka. 2004. Email task management styles: The cleaners and the keepers. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems: Extended Abstracts*. 1235–1238.
[34] Nathan Hahn, Shamsi T Iqbal, and Jaime Teevan. 2019. Casual microtasking: Embedding microtasks in Facebook. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 19.
[35] Ahmed Hassan, Rosie Jones, and Kristina Lisa Klinkner. 2010. Beyond DCG: User behavior as a predictor of a successful search. In *Proceedings of the ACM WSDM Conference on Web Search and Data Mining*. 221–230.
[36] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 159–166.
[37] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 256–265.
[38] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the ACM CIKM Conference on Information and Knowledge Management*. 2333–2338.
[39] Shamsi T Iqbal, Yun-Cheng Ju, and Eric Horvitz. 2010. Cars, calls, and cognition: Investigating driving and divided attention. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 1281–1290.
[40] Shamsi T Iqbal, Jaime Teevan, Dan Liebling, and Anne Loomis Thompson. 2018. Multitasking with play write: A mobile microproductivity writing tool. In *Proceedings of the ACM UIST Symposium on User Interface Software and Technology*. 411–422.
[41] Robert A Josephs and Eugene D Hahn. 1995. Bias and accuracy in estimates of task duration. *Organizational Behavior and Human Decision Processes* 61, 2 (1995), 202–213.

[42] Daniel Kahneman and Amos Tversky. 1977. *Intuitive Prediction: Biases and Corrective Procedures.* Technical Report. Decisions and Designs Inc Mclean Va.

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

[44] Thivya Kandappu, Archan Misra, Shih-Fen Cheng, Nikita Jaiman, Randy Tandriansyah, Cen Chen, Hoong Chuin Lau, Deepthi Chander, and Koustuv Dasgupta. 2016. Campus-scale mobile crowd-tasking: Deployment and behavioral insights. In *Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work and Social Computing*. 800–812.

[45] Bumsoo Kang, Chulhong Min, Wonjung Kim, Inseok Hwang, Chunjong Park, Seungchul Lee, Sung-Ju Lee, and Junehwa Song. 2017. Zaturi: We put together the 25th hour for you. Create a book for your baby. In *Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work and Social Computing*. 1850–1863.

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

[47] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[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), 27.

[49] Cornelius J König. 2005. Anchors distort estimates of expected duration. *Psychological Reports* 96, 2 (2005), 253–256.

[50] Zornitsa Kozareva and Eduard Hovy. 2011. Learning temporal information for states and events. In *Proceedings of Semantic Computing*. 424–429.

[51] Justin Kruger and Matt Evans. 2004. If you don't want to be late, enumerate: Unpacking reduces the planning fallacy. *Journal of Experimental Social Psychology* 40, 5 (2004), 586–598.

[52] Carol C Kuhlthau. 1991. Inside the search process: Information seeking from the user's perspective. *Journal of the Association for the Information Science and Technology* 42, 5 (1991), 361–371.

[53] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* (1977), 159–174.

[54] 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.

[55] Yuelin Li and Nicholas J Belkin. 2008. A faceted approach to conceptualizing tasks in information seeking. *Information Processing and Management* 44, 6 (2008), 1822–1837.

[56] Jingjing Liu and Nicholas J Belkin. 2010. Personalizing information retrieval for multi-session tasks: The roles of task stage and task type. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 26–33.

[57] Thomas W Malone. 1983. How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Information Systems* 1, 1 (1983), 99–112.

[58] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[59] Gloria Mark, Victor M Gonzalez, and Justin Harris. 2005. No task left behind? Examining the nature of fragmented work. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 321–330.

[60] David Martin, Benjamin V Hanrahan, Jacki O'Neill, and Neha Gupta. 2014. Being a turker. In *Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work and Social Computing*. 224–235.

[61] Nina Mishra, Ryen W White, Samuel Ieong, and Eric Horvitz. 2014. Time-critical search. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 747–756.

[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] Karen L Myers and Neil Yorke-Smith. 2005. A cognitive framework for delegation to an assistive user agent. In *Proceedings of the AAAI Fall Symposium*. 94–99.

[64] Andrew Y Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the International Conference on Machine learning*. 78.

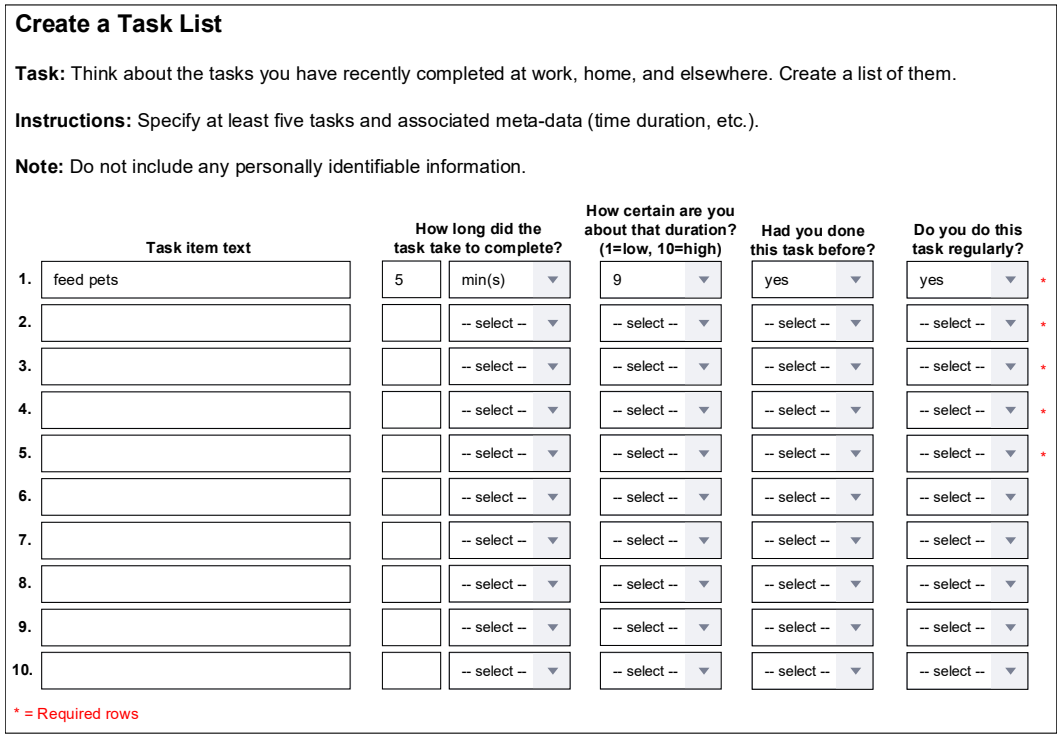
[65] Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. TORQUE: A reading comprehension dataset of temporal ordering questions. *arXiv preprint arXiv:2005.00242* (2020).

[66] Feng Pan, Rutu Mulkar-Mehta, and Jerry R Hobbs. 2011. Annotating and learning event durations in text. *Computational Linguistics* 37, 4 (2011), 727–752.

[67] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the EMNLP Conference on Empirical Methods in Natural Language Processing*. 1532–1543.
[68] Bradley J Rhodes. 1997. The wearable remembrance agent: A system for augmented memory. *Personal Technologies* 1, 4 (1997), 218–224.
[69] Daniel E Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the International Conference on the World Wide Web*. 13–19.
[70] Lee Ross and Richard E Nisbett. 2011. *The Person and the Situation: Perspectives of Social Psychology*. Pinter & Martin Publishers.
[71] Cristina Sarasua, Elena Simperl, and Natalya F Noy. 2012. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the International Semantic Web Conference*. 525–541.
[72] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell Systems Technical Journal* 27, 3 (1948), 379–423.
[73] Arthur R Taylor, Colleen Cool, Nicholas J Belkin, and William J Amadio. 2007. Relationships between categories of relevance criteria and stage in task completion. *Information Processing and Management* 43, 4 (2007), 1071–1084.
[74] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. 2007. Information re-retrieval: Repeat queries in Yahoo's logs. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 151–158.
[75] Jaime Teevan, Shamsi T Iqbal, Carrie J Cai, Jeffrey P Bigham, Michael S Bernstein, and Elizabeth M Gerber. 2016. Productivity decomposed: Getting big things done with little microtasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 3500–3507.
[76] Jaime Teevan, Shamsi T Iqbal, and Curtis Von Veh. 2016. Supporting collaborative writing with microtasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 2657–2668.
[77] Jaime Teevan, Daniel J Liebling, and Walter S Lasecki. 2014. Selfsourcing personal tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems EA*. 2527–2532.
[78] Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S Bernstein. 2014. Twitch crowdsourcing: Crowd contributions in short bursts of time. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 3645–3654.
[79] Pertti Vakkari. 2003. Task-based information searching. *Annual Review on Information Science and Technology* 37, 1 (2003), 413–464.
[80] OU Vortac, Mark B Edwards, and Carol A Manning. 1995. Functions of external cues in prospective memory. *Memory* 3, 2 (1995), 201–219.
[81] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *Proceedings of the International Conference on the World Wide Web*. 1353–1364.
[82] Ryen W White, Mikhail Bilenko, and Silviu Cucerzan. 2007. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 159–166.
[83] Ryen W White and Ahmed Hassan Awadallah. 2019. Task duration estimation. In *Proceedings of the ACM WSDM Conference on Web Search and Data Mining*. 636–644.
[84] 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 ACM SIGIR Conference on Research and Development in Information Retrieval*. 405–414.
[85] Ryen W White and Diane Kelly. 2006. A study on the effects of personalization and task information on implicit feedback performance. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 297–306.
[86] Alex C Williams, Harmanpreet Kaur, Shamsi Iqbal, Ryen W White, Jaime Teevan, and Adam Fourney. 2019. Mercury: Empowering programmers' mobile work practices with microproductivity. In *Proceedings of the ACM UIST Symposium on User Interface Software and Technology*. 81–94.
[87] Jennifer Williams. 2012. Extracting fine-grained durations for verbs from Twitter. In *Proceedings of the Association for Computational Linguistics Research Workshop*. 49–54.
[88] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *ArXiv* (2019), arXiv–1910.
[89] Ya Xu and David Mease. 2009. Evaluating web search using task completion time. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. 676–677.
[90] Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "Going on a vacation" takes longer than "Going for a walk": A study of temporal commonsense understanding. *arXiv preprint arXiv:1909.03065* (2019).
[91] Ben Zhou, Qiang Ning, Daniel Khashabi, and Dan Roth. 2020. Temporal common sense acquisition with minimal supervision. *arXiv preprint arXiv:2005.04304* (2020).

[92] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67, 2 (2005), 301–320.

A APPENDIX



Create a Task List

Task: Think about the tasks you have recently completed at work, home, and elsewhere. Create a list of them.

Instructions: Specify at least five tasks and associated meta-data (time duration, etc.).

Note: Do not include any personally identifiable information.

	Task item text	How long did the task take to complete?		How certain are you about that duration? (1=low, 10=high)	Had you done this task before?	Do you do this task regularly?	
1.	feed pets	5	min(s)	9	yes	yes	*
2.			-- select --	-- select --	-- select --	-- select --	*
3.			-- select --	-- select --	-- select --	-- select --	*
4.			-- select --	-- select --	-- select --	-- select --	*
5.			-- select --	-- select --	-- select --	-- select --	*
6.			-- select --	-- select --	-- select --	-- select --	
7.			-- select --	-- select --	-- select --	-- select --	
8.			-- select --	-- select --	-- select --	-- select --	
9.			-- select --	-- select --	-- select --	-- select --	
10.			-- select --	-- select --	-- select --	-- select --	

* = Required rows

Fig. 8. Screenshot of the interface for the human intelligence task (HIT) presented to crowdworkers to elicit to-do tasks. Workers were asked to enter the titles of at least five tasks they had recently completed plus corresponding answers related to time on task, certainty, and task history. The task time is captured via a text box, with units specified in terms of seconds, minutes, hours, or days in the adjacent drop down. For each task, the interface captured information on crowdworker certainty in their reported duration (on a scale from 1-10, 10 is highest), whether they had done this task before (yes/no), and whether they do this task regularly (yes/no). A random example task from the dataset ("feed pets") and the associated additional responses is included for reference.