# Task Completion Detection

## A Study in the Context of Intelligent Systems

Ryen W. White
Microsoft Research
Redmond, WA 98052
ryenw@microsoft.com

Ahmed Hassan Awadallah
Microsoft Research
Redmond, WA 98052
hassanam@microsoft.com

Robert Sim
Microsoft Research
Redmond, WA 98052
rsim@microsoft.com

## ABSTRACT

People can record their pending tasks using to-do lists, digital assistants, and other task management software. In doing so, users of these systems face at least two challenges: (1) they must manually mark their tasks as complete, and (2) when systems proactively remind them about their pending tasks, say, via interruptive notifications, they lack information on task completion status. As a result, people may not realize the full benefits of to-do lists (since these lists can contain both completed and pending tasks) and they may be reminded about tasks they have already done (wasting time and causing frustration). In this paper, we present methods to automatically detect task completion. These inferences can be used to deprecate completed tasks and/or suppress notifications for these tasks (or for other purposes, e.g., task prioritization). Using log data from a popular digital assistant, we analyze temporal dynamics in the completion of tasks and train machine-learned models to detect completion with accuracy exceeding 80% using a variety of features (time elapsed since task creation, task content, email, notifications, user history). The findings have implications for the design of intelligent systems to help people manage their tasks.

## CCS CONCEPTS

• **Computing methodologies** → **Intelligent agents**; • **Information systems** → *Data mining*;

**ACM Reference Format:**
Ryen W. White, Ahmed Hassan Awadallah, and Robert Sim. 2019. Task Completion Detection: A Study in the Context of Intelligent Systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3331184.3331187

## 1 INTRODUCTION

Task management applications, productivity applications, and digital assistants store and remind people about their pending personal and work tasks [8]. Tasks can be explicitly specified by users (e.g., added to to-do lists) [11] or inferred from user data (e.g., commitments and requests extracted from email communications [16] or search tasks mined from logs [60]). Supporting task retention addresses well-known limitations in human memory [41] and can be combined with context-sensitive reminders [34] and attention-sensitive alerts [6] for prospective remembering.

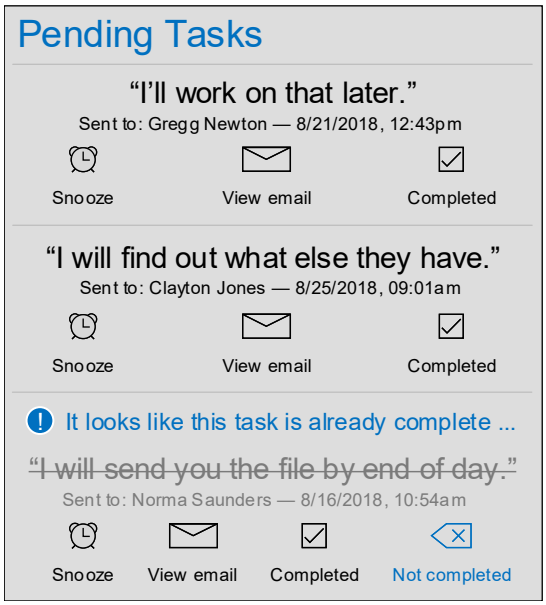



**Figure 1: Example of task completion detection in an intelligent system. Here, the system displays a user's pending tasks: commitment sentences (promises) mined from sent email. The inferred completed task (third in the list) is in gray with strikethrough. Selecting "Not completed" undoes the effect of completion detection. If no such feedback is given, after a short delay, the task could be deprecated.**

Despite the advantages, users of systems that store and surface pending tasks face at least two challenges: (1) task lists can grow over time if new tasks get added and old tasks are not retired; making task list management a burden and meaning that those lists lose their utility as a way to highlight tasks in need of attention, and (2) by ignoring task status, intelligent systems can notify users about tasks that they have already completed, leading to unnecessary interruption, frustration, and impeding user performance on their current task (e.g., studies have shown that email notifications can be highly disprutive [31]). Although tasks can be set to expire after a fixed amount of time (say, a week), this is not ideal because it ignores task state, meaning that tasks may still be active when they are marked by the system as complete. Methods to more intelligently flag and/or deprecate completed tasks are therefore desirable.


*SIGIR '19, July 21–25, 2019, Paris, France*

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331187

In this paper, we tackle the challenge of automatic task completion detection. We show that the completion of pending tasks by users can be accurately detected via machine-learned models trained on user data. In web search, log data is mined and used to estimate search success [27] and predict the likelihood of task continuation (the opposite of completion) [2, 39]. Applying task completion detection models in intelligent systems such as digital assistants has the potential to, e.g., reduce the number of non-actionable notifications that users receive. Figure 1 shows an example of how task completion detection could be applied to a set of pending commitments extracted from email (the dataset used in this study, described in detail later). In this example, the system displays tasks with a completion probability above a predefined threshold in a gray font with strikethrough, offering users the option to undo. These tasks can be quickly archived and removed from the list if no recourse is requested. Even in a notification suppression scenario (our focus later), the assistant may choose to not alert users about tasks that are likely complete but may still show them in a task list, to provide visibility on which tasks are inferred complete and offer recourse. Applying the estimated likelihood of task completion for task ranking and task prioritization represents other ways to help users focus their attention on what tasks remain to be done.

The main contributions of this paper are as follows:

- Introduce *task completion detection* as an important new machine-learning challenge, with numerous applications in intelligent systems, including digital assistants, task management systems, and productivity applications.
- Analyze task completion data from a popular digital assistant, revealing important trends in the temporal dynamics of task completion per task priority, intent, and timing (see Section 3.2).
- Train machine-learned classifiers to automatically detect the completion of tasks using a range of signals, including time elapsed since task initiation, contextual signals, and aspects of the task itself (e.g., time until due date).
- Present implications for the design of intelligent systems from being able to accurately detect task completion.

The remainder of the paper is structured as follows. We first present related work, in areas including task management and task completion. We then describe the dataset and associated analysis. The paper then covers the detection of task completion, including experiments to evaluate the effectiveness of our models. Next, we discuss our findings and their implications for the design of intelligent systems. We then conclude with a summary of our contributions and pointers to avenues of future work.

## 2 RELATED WORK

Research in three areas is particularly relevant: (a) task management, (b) task completion in general, and (c) task completion in search.

### 2.1 Task Management

A range of best practices have been proposed to help people manage and prioritize their tasks [3, 17]. To assist in this process, computer systems have been developed to help people capture tasks in a lightweight manner and support their completion via higher-order activities such as planning, question answering, and problem solving [8]. Tasks can be defined manually by users or systems can detect current tasks and forecast future tasks based on computer activities [54]. Work on issue tracking and forecasting in software development and project management is also relevant [37].

The process of manually creating task lists can be beneficial (e.g., to reduce intrusive thoughts connected to unfinished goals [45]), but it is also burdensome and may even be ineffective if disconnected from how the task will be performed [25]. Actionable statements mined from user data such as electronic mail and instant messages [9, 16, 50] can be added to task lists. However, a failure to retire tasks as they are completed reduces the value of task enumeration since it could soon be unclear which tasks require attention. Population-scale data has been used to estimate the amount of time required to complete a task [61] but not whether pending tasks are completed.

### 2.2 Task Completion in General

Prior work on task completion prediction is somewhat related to task completion detection. Prediction research has targeted time estimation [52], especially the effects of overconfidence and optimism biases on how much time people expect tasks to take [12, 33]. Time estimation failures have implications for task management: underestimating duration may make users more willing to add items to their task lists even if they are incapable of completing them in the desired timeframe. Support for remembering future tasks, so-called prospective memory [22], is a core capability of tools to assist with task completion [41]. Beyond simply remembering, successful prospective memory requires recall at the appropriate time. Systems can employ context-sensitive reminding to jog memories about the information people are likely to forget [34], while also considering factors such as current attentional demand [6].

Digital assistants such as Amazon Alexa, Google Assistant, and Microsoft Cortana provide timers to help people track short time durations and enable users to create reminders to remember to perform future tasks [26], even if the specific task timeframe is imprecise [53]. These reminders can be triggered by contextual cues such as time or location (or combinations thereof [19, 44]). Research on characteristics of interruptive notifications has explored how to design notifications to help people better prioritize and manage their tasks [51]. Work on notification strategies has focused on factors such as user busyness [30] and task urgency [58], but, importantly, not on whether the task is still valid (e.g., is incomplete) at notification time.

Research on activity recognition is also relevant (e.g., [7, 55]), where the goal is to detect the current task that a person is performing and hence also recognize the boundaries (start and end) of that task or sub-task. Although similar, completion detection focuses on methods to infer the *status* of a task, whereas activity detection would focus on recognizing the task itself. Research on task recognition and planning [59] focuses on understanding task objectives and the next steps/tasks required to complete the task, versus completion detection as we describe in this paper.

### 2.3 Task Completion in Search

Tasks play a central role in information seeking and retrieval. By considering aspects of the task, we can better predict information behavior [42] and build more effective information systems [57]. Researchers have studied different characteristics of search tasks,

including complexity [13], difficulty [4], and other facets such as type [43] and time sensitivity [48]. Research on task stage has examined its impact on relevance criteria [56] and search behavior [40]. Task progress and completion is important in the design of systems to support user studies in information retrieval [10].

Search tasks can be short-term (single search sessions) or long-term (spanning multiple sessions). In analysis of short-term search tasks, clicks on search results are often used as a signal of task completion that is used in ranking algorithms [1, 32]. Research on search satisfaction enhances click signals with additional information about landing page dwell times [23, 38] and post-click query reformulation [28] to better estimate whether users are satisfied. Focusing on search success, Hassan et al. [27] modeled session-level search goals and showed that models using search behavior could accurately predict session success. Beyond engagement with the search engine, terminal URLs (last clicks in search trails or search sessions) have also been used to identify completion events [21, 62]. Research on abandonment has attempted to estimate whether users are satisfied with search results without needing to click [20]. Task completion time has been used to evaluate search algorithms [63].

Long-term tasks can be explicitly labeled by searchers [36] or extracted automatically from search logs [60]. Liu and Belkin [43] examined the structure (parallel or dependent) of tasks that extend across different sessions, for search personalization. Research on automatically predicting the continuation of long-running tasks [2, 39] can assist with task completion detection and be used in tools such as *SearchBar* [49] to support task continuation across sessions. While the focus on task completion in search is on learning from search activity (e.g., using long dwells or last clicks [38]), tasks manifest in many ways outside of search engine interactions and evidence of task completion takes many forms.

### 2.4 Contributions over Previous Work

We make several contributions over prior work. First, we focus on task completion detection, an important new machine-learning challenge in the task management domain. This can be used in applications such as digital assistants and productivity software. Second, although there has been work on lightweight tools to capture tasks (or even extract them from communications), there have been no attempts to ensure that intelligent systems have accurate knowledge of pending tasks, to help focus people's attention during visual inspection of task lists or to decide which tasks are worth notifying people about. Third, although there has been research on search success and search task continuation, detecting task completion is an unexplored area. Fourth, through mining large-scale tasks data from a popular digital assistant, we show noteworthy trends in when people complete tasks and how that varies with task properties, including task intent and task priority. Finally, we demonstrate the feasibility of building machine-learned models to accurately detect task completion. This has implications for designing systems to help people better manage their tasks.

## 3 DATA

We use a random sample of task data from around 1.2 million users of the Microsoft Cortana digital assistant deployed to consumers in the United States (en-us locale). The assistant tracks commitments (first-person statements of actionable intent) made in email between users and other individuals, e.g., "I'll send the report by end of day." Additional examples of commitment sentences are shown in Figure 1. The assistant identifies commitments from emails using methods similar to previous work [16]. In total, a sample of 3,006,218 commitments collected in 2017 and 2018 are used in this study (average per user=2.34 tasks, standard deviation=4.89 tasks, median=1 task). To preserve privacy, text was anonymized by replacing each token in each commitment sentence with a one-way hash early in the data processing pipeline. To ensure that the list of commitments does not grow indefinitely, commitments persist in the assistant interface canvas for up to 14 days following the original commitment being made in email. As a result, we focus on temporal dynamics and completion detection during a 14-day timeframe.

### 3.1 Task Completion Labels

After the commitment task has been identified by the digital assistant, metadata is extracted by proprietary algorithms, including any due dates that are mentioned. In total, 24.1% of commitments in the dataset contain due dates and on average, that date falls within 1.78 days of the commitment being made in email (standard deviation=2.62 days, median=0.71 days). 86.3% of commitments are made on weekdays. Commitments made on days immediately preceding at least one weekend day (i.e., made on Friday or Saturday) have a significantly longer time until due than those made on other days of the week (Friday/Saturday: average=1.95 days, median=1.00 days; Other: average=1.74 days, median=0.68 days, unpaired t-test: $t(723835)=-46.81$, $p < 0.001$). In Cortana, the due date determines when the assistant should trigger notifications to remind users to complete the pending task. Periodic notifications are also shown to users if no due date appears in the commitment text. As mentioned earlier, we target notification triggering as the primary application scenario for task completion detection (one of many) and we focus on that in the remainder of the paper.

In intelligent systems such as digital assistants, tasks can be shown to users in lists or pushed to them via notifications. Users can interact with tasks in several ways (e.g., clicking "Completed" as in Figure 1). Cortana provides an affordance similar to Figure 1, which allows users to explicitly indicate task completion. These "Completed" clicks help form the ground truth for our study. However, they only tell us the task was completed *by* some time, not *when* completion occurred (i.e., they are an upper bound of task completion time). We employ the log data from Cortana by focusing on a notifications scenario, where the goal is to only notify users for tasks that are not yet completed.

Given a task initiated at datetime $t_i$ and a datetime representing the system decision point ($t_n$) (when the system must decide whether to notify), determine whether the task is already complete *by* $t_n$. Since our analysis is log-based/retrospective, we cannot control when notifications are shown to users. Instead, we simulate that system decision point by picking a random notification delay ($d$) after $t_i$ and using whether the task is complete by $t_n$ as the completion label. Specifically, we apply the following steps for each commitment task initiated at $t_i$:

(1) Generate $d$, a random integer between 1 and 14 days (inclusive).

(2) Add $d$ days to $t_i$ to generate the point in time when the system must decide whether to show a notification to the user ($t_n$). For simplicity, we assume that the only factor in this decision is completion status, but as discussed later, there are many factors in deciding whether to present interruptive notifications.
(3) Check whether the task was marked by the user as completed (we have observed a "Completed" click) at some time before $t_n$:
  (a) If *yes*, assign the task a positive label (complete by $t_n$).
  (b) If *no*, assign the task a negative label (incomplete by $t_n$).

Randomized selection of $t_n$ was our chosen simulation strategy, primarily for simplicity. It is also desirable because it allows us to frame the problem as binary classification and is similar to how such a classifier could be used in practice, e.g., to gate notifications. Alternatives (e.g., fixed offset(s) from $t_i$) are also feasible, but this involves making arbitrary decisions about $d$ when the notification strategy would likely vary per application.

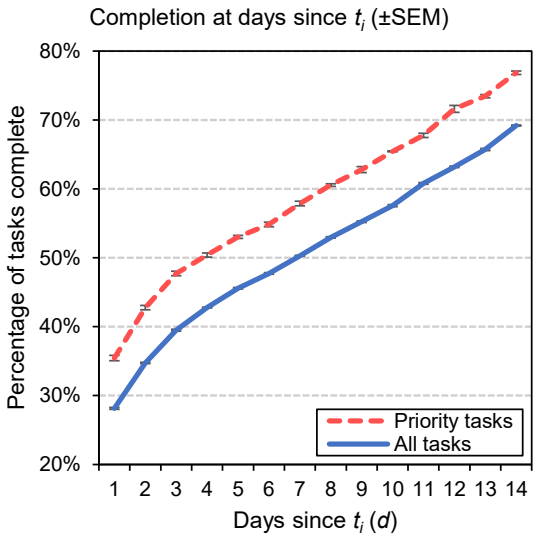
Note that we could not use the timing of the "Completed" clicks as positives because they could fall well after the task is actually completed, and it is not realistic to assume that the system would need to make decisions exactly at the time that users are clicking "Completed." Instead, we select random, simulated notification times $\{t_n\}$ and use the occurrence of "Completed" clicks by those times to generate positive and negative labels.

Labels are distributed as 1,531,747 (51.0%) positive and 1,474,471 (49.0%) negative. Task completion is time dependent: the fraction of tasks completed depends on $d$. We now explore those temporal dynamics in more detail.

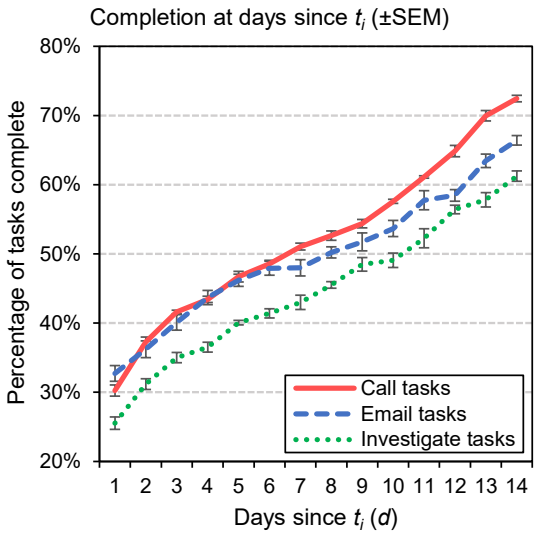
## 3.2 Temporal Dynamics

*3.2.1 Task Completion Over Time.* Using the labels generated from the process described above we can also analyze some of the temporal dynamics of task completion. Specifically, we can study how quickly tasks are completed in general and how this temporal task completion distribution changes with different tasks. On each day from 1-14 days after $t_i$, we compute the fraction of tasks that are already completed. Since $d$ is generated randomly, there is some variance in the completion percentages at each $d$, depending on how $d$ is distributed. To help address this, we randomly split the data for each $d$ into five folds and report the mean average and standard error completion percentage at each $d$.

In addition to the distribution across all tasks, we consider two aspects of the commitments: (1) intent and (2) activity. Since we lack labels from users, we look for terminology in the commitment sentence that reflects these aspects. For priority, we look for words or phrases reflective of a high priority task (e.g., "asap," "is urgent," "is important") (building on the priority vocabulary used in [29]). In total, 8.2% of the commitment sentences in the data set contain priority language. We ran proprietary intent classifiers (which use constituency trees and word embeddings to extract action-object pairs from text) on the commitments to identify the intent to perform one or more the following actions: call, email, investigate, payment, meeting, and information. Some action intent was detected in 7.8% of commitments. We inspect the temporal completion distributions for tasks containing the three most common intents appearing in commitments in our dataset (call [in 3.0% of all commitments], email [1.3%], and investigate [1.4%]). Action




**(a) Average task completion by $t_n$ for all tasks and for tasks with priority language**




**(b) Average task completion by $t_n$ for tasks broken out by intent class**

**Figure 2: Percentage of tasks completed by $t_n$ (i.e., $t_i + d$). Distributions shown (a) all tasks (solid line) and tasks with priority language (dashed line), and (b) three most common task intents: call, email, and investigate (top to bottom). Error bars denote standard error of the mean (±SEM).**

intent classification and priority language flagging is performed early in the data processing pipeline, before token hashing.

The results in Figure 2a show the task completion percentage over time, ranging from 28.1% at the end of the first day to 69.2% at the end of the fourteenth day.[1] The upward trend is expected as people make progress over time, but the specific percentages are

[1]The remaining 30.8% of commitments were not marked complete in 14 days of $t_i$.

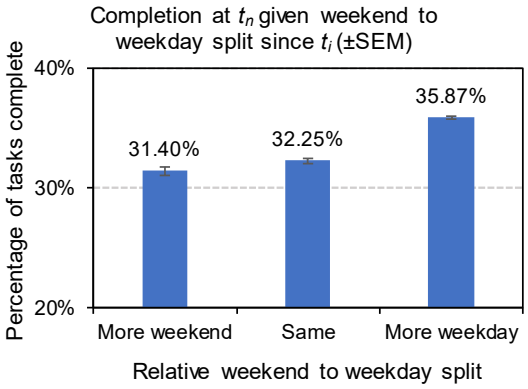



**Figure 3: Percentage of tasks already completed by $t_n$ (i.e., $t_i + d$), broken out by the relative split of weekend days and weekdays between $t_i$ and $t_n$ (limited to cases where $d$=2 days). Error bars denote standard error of the mean (±SEM).**

valuable new knowledge, including for the learning task described later. Focusing on tasks containing priority language, we see that they are more likely to be completed more quickly on the first day (35.4%) and at the end of the fourteenth day (76.8%). The presence of such clear differences in completion rate per priority language bodes well for task completion detection. However, we were also interested in whether more subtle effects, specifically, those related to task intent, impacted task completion distributions. Figure 2b shows the task completion distributions for *call*, *email*, and *investigate* tasks. Interestingly, there are differences in the distributions across these task types. Call tasks are completed more quickly, followed by email, and then by investigate. This ordering may reflect the relative complexity (on average) of these different task types. Call tasks may also tend more towards faster completion because they represent a more direct interpersonal promise versus sending an email. Call and email tasks appear similar for the first few days, perhaps because there are some email tasks that can be handled as quickly as a short telephone call.

*3.2.2 Task Completion Per Weekends and Weekdays.* In addition to analyzing the task completion rates per $d$, we were also interested in whether there were differences associated with the number of weekend days and weekdays in the time from $t_i$ to $t_n$. Weekdays (Monday to Friday) are workdays in the United States, so it is reasonable to assume that more tasks might get done on weekdays. We computed the completion rate at $t_n$ as a function of the relative number of weekend days and weekdays in the time since $t_i$. There can be many confounds in such analysis that make a fair comparison difficult, e.g., to observe four weekend days, the time period needs to include at least five weekdays. To address this, we focus on $d$=2 days and compute three groups: (1) *More weekend* (two weekend days and no weekdays), (2) *Same* (one weekend day and one weekday), and (3) *More weekday* (no weekend days and two weekdays). We computed the average completion percentage at $t_n$ for each of the three groups. The values are shown in Figure 3. The task completion percentage is higher when there are more weekdays. The differences between completion percentages are significant using a one-way analysis of variance (ANOVA) ($F(2,214923)=148.64$, $p < 0.0001$).

*3.2.3 Summary.* The presence of the temporal trends noted in this section is promising for task completion detection. Even simple features such as the amount of time elapsed since $t_i$ and number of weekdays appear correlated with completion likelihood. We can include features related to time, intent, and priority, and many other signals, in learned models to detect task completion. We describe the models and their evaluation in the next section.

## 4 DETECTING TASK COMPLETION

We now describe the methods used to detect task completion and the evaluation of those methods to determine their effectiveness. We use the dataset described in the previous section and focus on the specific challenge of detecting completion of a pending task by time $t_n$ based on signals from a variety of sources. We ground this in a notification scenario, where a system must decide at $t_n$ whether to notify a user about a pending task. We assume that if the task is truly complete by $t_n$, the user will not want to be notified.

### 4.1 Methods

We train binary classifiers to detect task completion. We use the completion labels described in the previous section as the ground truth and the commitments dataset as a source of training and test data, from which we extract many different features. The features are divided into the following five classes:

*4.1.1 Time:* The amount of time that has elapsed between $t_i$ and the time of the anticipated notification $t_n$ (i.e., $d$). Empirical analysis in the previous section already showed the strong relationship between $d$ and task completion percentages. Given the differences in task completion percentages noted in Figure 3, we also include a feature for the number of weekdays between $t_i$ and $t_n$.

*4.1.2 Commitment:* Text content of the commitment, including $n$-grams (up to 3-grams) extracted from the anonymized (token hashed) text. Some features were computed pre-anonymization. These include frequency counts of action verbs [14] and priority language, as well as features about the due date (e.g., whether there is a due date and the time between $t_n$ and that datetime). This also includes features about who has the responsibility to complete the commitment: the promisee or a delegate, or whether it is shared between multiple individuals. There are also additional features on the strength of the commitment (e.g., whether it is conditional). Finally, the output of the proprietary binary classifiers that identify the intent of the commitment (e.g., email, call, etc.) (described in detail in Section 3.2.1) are included as additional binary features.

*4.1.3 Email:* The email from which the commitment is drawn may provide additional insight on the task. The full text of the source email or email thread is unavailable in this study. We do have access to the email subject (anonymized as with the commitment text) from which we extract features including: $n$-grams, whether email is a reply, subject token lengths, whether the subject is a question, etc. Additional features about number of recipients on To:, Cc:, and Bcc:, as well the number of unique email domains are also included, since they may reveal details about the breadth of the commitment and whether or not it is made across organizational boundaries. Future studies could use such data to generate more sophisticated

email features, such as those based on follow-up emails sent by $t_n$ with replies indicating completion, or any links or attachments.

*4.1.4 Notifications:* After the commitment is made, Cortana will send the user occasional notifications to remind them about their pending tasks. The presence of such notifications between $t_i$ and $t_n$ may be a good indication of task completion since the user is prompted to both (a) recall the commitment made and (b) provide feedback about the completion status. Overall, notifications were generated for 15.6% of commitments in our dataset. The average time $t_i$ until first notification was 0.64 days (standard deviation=1.27 days, median=0.30 days). There are two types of notification: (1) *toast notifications*, which generate popup reminders, and (2) *ghost notifications*, which are placed directly into the notification center in the operating system. Both types can be viewed in the notification center for around two days post triggering. We featurize various aspects of the notifications using the information available between $t_i$ and $t_n$ for task completion detection (see Table 1). In Cortana, users need not interact with notifications to indicate completion. They can access their pending tasks on the assistant canvas or in the notification center at any time and mark tasks as complete.

*4.1.5 User:* Since there are multiple commitments for 37.7% of the users in our dataset, we could also use commitment task completion history features from the time before $t_n$. We count the number of tasks and the average and standard deviation time until task completion for that user. For time until task completion, the task needs to be explicitly marked as complete by the user before $t_n$.

The complete list of features used for completion detection is shown in Table 1, along with a description of each feature.

## 4.2 Learning Algorithms

We experimented with several common learning algorithms for task completion detection: logistic regression (LR), gradient boosting decision trees (GBDT, [24]) and neural networks. Since this is the first study of task completion detection, we wanted to understand the effectiveness of these well-known methods before developing more sophisticated approaches. We now describe each algorithm.

**Logistic Regression**: Logistic regression yields compact, interpretable models that have been previously used for task modeling on email (e.g., [16]). We train an LR model using the attributes described in Table 1 as features. Note that all features in Table 1 are numerical except for the text description of the task and the subject of the email. Numerical features can be passed directly to the model post-normalization. For the text attributes, we generate $n$-grams (up to 3-grams) and use $n$-gram TF-IDF values as features.

**Gradient Boosted Trees**: Gradient boosting techniques produce a prediction model in the form of an ensemble of weak prediction models (e.g., decision trees). GBDT is a widely-used machine learning algorithm that has been shown to achieve state-of-the-art performance in many machine learning tasks. Its advantages include efficiency, accuracy, robustness to missing or noisy data, and interpretability. We use LightGBM [35] for our experiments. LightGBM is a gradient boosted decision tree implementation optimized for high computational speed and low memory consumption.

**Neural Networks**: Motivated by the recent advances in applying neural network methods to natural language understanding tasks, we use a neural network approach to better represent the commitment text and combine it with other numerical features. Given a task description (i.e., commitment text) $T$ with a list of words $w_i, i \in 1..n$, we aim to embed it into a fixed size representation vector. For each word $w_i$ in the commitment sentence, we first transform them into dense vectors through a word embedding matrix $W \in R^{dim \times |V|}$. Here, $|V|$ is the vocabulary size and $dim$ is the dimensionality of the word embedding. We cannot use pre-trained word embeddings since we use the anonymized (token hashed) version of the text, hence we train our own word embedding vectors using a skip-gram hierarchical SoftMax model [47].

After placing words with their embedding vectors, we apply a bi-directional RNN with GRU cells to the text, scanning it forward and backward. We obtain the hidden state $h_i$ for each word $w_i$ in the appointment subject $T$ by concatenating the forward hidden state and the backward hidden state. To generate a single vector representing the whole task description $T$, we leverage the attention mechanism [5] and use the weighted average of the hidden states to represent $T$. Finally, we concatenate the text representation with other numeric features and pass them to the final classifier.

## 4.3 Evaluation

To evaluate the model, we split the full dataset of 3,006,218 commitment tasks to training, validation, and test datasets. We split the dataset such that we have a validation set of 50K instances and a test set of 50K instances. This leaves us with 2,906,218 instances for training. Data was split based on user identifier such that data from any given user would belong to either training, validation, or test data. We used the validation set to tune all hyperparameters (L1 and L2 weights for LR, learning rate, boosting iterations and number of leaves for LightGBM and batch size, learning rate, dropout rate, and GRU hidden unit size for deep learning). We use the test data for evaluation and report metrics including accuracy, F1, and area under the precision-recall curve (AUC). To test statistical significance, we use two-tailed t-tests with bootstrap sampling ($n = 10$).

*4.3.1 Overall Results.* The overall performance results are presented in Table 2 for all three models. The results show that the LR model performs worst; in terms of F1-measure. The LightGBM and the Neural Network models perform better. Their performance is comparable, with a slight improvement in F1-measure over the Neural Network model. Note that the main advantage of the Neural Network model is that it can better encode the commitment text. We will show in the ablation study below that removing the commitment text results in a small loss of performance as other commitment features tend to carry similar information. This may explain why the Neural Network model has less than the expected gain over the other models. Investigating different model architectures might be necessary to realize additional performance gains. In summary, the results show that we can predict task completion with both F1 and accuracy exceeding 80% and that the Neural Network model yields the best performance. The LightGBM model yields slightly lower performance but has the advantages of being simpler, more interpretable, and is faster to train.

*4.3.2 Effect of Data Volume.* We vary the training set size from 25K instances to 3M instances and observe the performance of each

**Table 1: Features used for the detection of task completion, broken out by feature class.**

| Feature | Description |
|---|---|
| *Time class* | |
| TimeElapsed | Amount of time elapsed from commitment being made ($t_i$) until the prospective notification time ($t_n$) |
| NumWeekdays | Number of workdays (Mon-Fri) from commitment being made ($t_i$) until the prospective notification time ($t_n$) |
| *Commitment class* | |
| CommitmentText | $n$-grams in anonymized commitment text |
| [DayOfWeek\|Hour\|Minute]CommitmentMade | When the commitment was made |
| CommitmentMadeOnWeekend | Whether commitment is made on a weekend |
| ConfidenceScore | Confidence score (in range [0,1]) from the commitment extraction model |
| CommitmentTokenLength | Token count of commitment text |
| CommitmentCharLength | Character count of commitment text |
| CommitmentAvgTokenLength | Average length of tokens in the commitment text |
| CommitmentExclamationCount | Number of exclamation marks in the commitment |
| CommitmentQuestionMarkCount | Number of question marks in the commitment |
| CommitmentCommaCount | Number of commas in the commitment |
| CommitmentPeriodCount | Number of periods in the commitment |
| IsSharedResponsibility | Whether commitment language suggests that multiple people could handle the task (e.g., "we will", "we'll") |
| IsSoleResponsibility | Whether commitment language suggests that a single person will handle the task (e.g., "I will", "I'll") |
| IsDelegatedResponsibility | Whether commitment language suggests that someone else will handle the task (e.g., "they will", "⟨firstname⟩ will") |
| UndefinedResponsibility | Whether commitment has no language on single, shared, or delegated responsibility |
| HasObligation | Whether commitment language mentions an obligation to others (e.g., "for you", "to them") |
| IsConditional | Whether commitment is conditioned on another event or action (if. . . then) |
| IsHedge | Whether commitment language suggests that the promisee is hedging (e.g., "I can", "I may") |
| Has$X$Intent | Commitment intent as detected by proprietary classifier ($X \in$ {Payment,Email,Investigate,Meeting,Information,Call,None}) |
| NumActionWordsPhrasesCommitment | Number of action verbs in the commitment text |
| NumPriorityWordsPhrasesCommitment | Number of words or phrases suggesting priority in the commitment text |
| HasDueDate | Whether commitment has a due date ($t_d$) |
| [Day\|Hour\|Minute]Due | When the commitment is due |
| IsWeekendDue | Whether commitment is due on a weekend |
| MadeDueDateDiff | Time between $t_i$ and $t_d$ |
| NotificationDueDateDiff | Time between $t_n$ and $t_d$, including negative values (i.e., task is overdue) |
| *Email class* | |
| SubjectText | $n$-grams in anonymized email subject |
| SubjectTokenLength | Token count of email subject |
| SubjectCharLength | Character count of email subject |
| SubjectAvgTokenLength | Average length of words in the email subject |
| SubjectExclamationCount | Number of exclamation marks in the email subject |
| SubjectQuestionMarkCount | Number of question marks in the email subject |
| SubjectIsEmpty | Whether the subject is empty |
| NumActionWordsPhrasesSubject | Number of action verbs in the email subject |
| NumPriorityWordsPhrasesSubject | Number of priority words or phrases in the email subject |
| SubjectIsQuestion | Whether email subject ends with question mark |
| IsReply | Whether email is a reply (subject starts with "re:" or similar) |
| IsForward | Whether email is a forward (subject starts with "fwd:" or similar) |
| ToCount | Number of recipients on the To: line |
| CcCount | Number of recipients on the Cc: line |
| BccCount | Number of recipients on the Bcc: line |
| NumUniqueDomains | Number of unique email domains across the sender and all email recipients |
| *Notifications class (based on the timespan between $t_i$ and $t_n$)* | |
| TotalNotificationCount | Total number of notifications shown |
| TotalToastNotificationCount | Total number of toast notifications (popups) |
| TotalGhostNotificationCount | Total number of ghost notifications (direct to "notification center" in operating system) |
| TotalNotificationTime | Total time (in days) notifications are active |
| TotalToastNotificationTime | Total time (in days) toast notifications are active |
| TotalGhostNotificationTime | Total time (in days) ghost notifications are active |
| TimeSinceLastNotification | Time (in days) between $t_n$ and last notification |
| TimeSinceLastToastNotification | Time (in days) between $t_n$ and last toast notification |
| TimeSinceLastGhostNotification | Time (in days) between $t_n$ and last ghost notification |
| IsNotificationActive | Whether any notification is still active at $t_n$ |
| IsToastNotificationActive | Whether a toast notification is still active at $t_n$ |
| IsGhostNotificationActive | Whether a ghost notification is still active at $t_n$ |
| TimeToFirstNotification | Time (in days) from $t_i$ to the first notification |
| TimeToFirstToastNotification | Time (in days) from $t_i$ to the first toast notification |
| TimeToFirstGhostNotification | Time (in days) from $t_i$ to the first ghost notification |
| *User class (based on time before $t_n$)* | |
| UserHistoricTaskCount | Total number of historic tasks from that user |
| UserAvgTaskCompletionTime | Average of time to complete tasks from that user (tasks must be marked as complete by $t_n$) |
| UserStDevTaskCompletionTime | Standard deviation in time to complete tasks from that user (tasks must be marked as complete by $t_n$) |

**Table 2: Performance of completion detection models on full dataset (same features, different learning algorithms). Significance testing performed using two-tailed t-tests, differences between all pairs (in F1) are significant with $p < 0.01$.**

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| Logistic Regression | 87.17 | 73.87 | 79.97 | 81.11 |
| LightGBM | 78.92 | 83.90 | 81.37 | 80.48 |
| Neural Network | 87.67 | 77.40 | 82.21 | 83.00 |

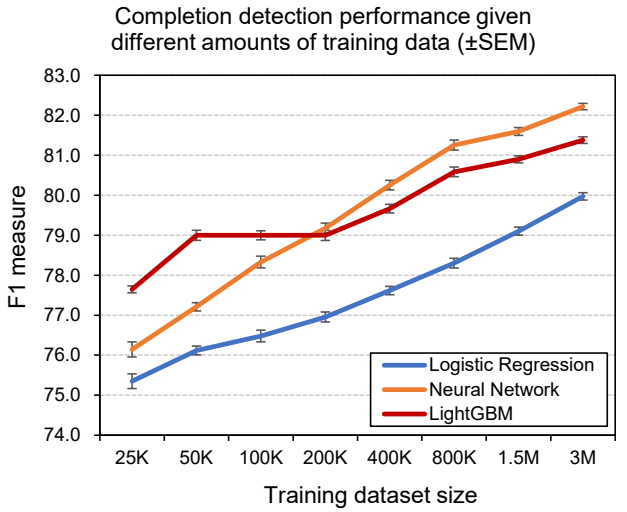



**Figure 4: Performance (as measured using F1-measure) of all models with various training set sizes ranging from 25K to 3M. Error bars denote standard error of the mean (±SEM).**

model. Figure 4 reports the performance at different dataset sizes per model. The figure shows that the LR model performs the worst at all data points. Interestingly, it performs reasonably well even when very little training data is used (75.34 and 76.11 F1-measure at 25K and 50K training data size). The performance improves as more training data is added reaching an F1-measure of 79.97 when the full dataset is used. Both the LightGBM and the Neural Network model outperform the LR model regardless of the training dataset size. The LightGBM model tends to perform the best when little training data is available (100K or less), while the Neural Network model performs better when more training data is added. Both models seem to benefit from adding more data, and the gain from adding more data is slightly larger for the Neural Network model.

*4.3.3 Effect of Features Used.* As shown in Table 1, we used a variety of features for the detection task. To understand the impact of the different feature classes, we experimented with dropping every feature class and comparing the performance of the model to that of the model using all features. We also tried training the model on only one feature class at a time and compared this to the performance of the full model. We use the LightGBM model for this experiment since it yields similar performance to the Neural Network model and is much faster to train. The results are shown in Table 3 and Table 4. In addition to the absolute performance, the table also shows the relative change compared to the model that uses all features. Significance testing is performed comparing the models trained on feature subsets with the model using all features.

Table 3 shows that as features are removed, the detection performance of the model decreases for most feature classes. The *Time*, *Email*, and *Notification* feature classes tend to have little to no impact on the performance when each is removed. If we examine the performance of the model when trained on only time, notification, or email features, we see that all of them have useful information for predicting task completion (Table 4). This suggests that the model could rely on other features to get the same information when any of these feature classes is missing. Although time is a strong signal of task completion likelihood (shown in earlier analysis, Figure 2), the performance of the model drops only slightly when the time features are excluded. In this case, other features, such as the notification features may provide enough signal about time elapsed since $t_i$ to counteract the loss of the time elapsed feature.

Interestingly, removing the commitment text does not seem to have significant impact on the performance either. This suggests that the text adds little value compared to other commitment features (many of them describe different characteristics of the commitment text such as the existence of action words, priority words, etc.). This also explains the earlier results showing the impact of a more complex model for encoding the text (the Neural Network model) is limited compared to other much simpler models. Conversely, removing all commitments features seems to have the most significant impact (with respect to dropping other feature classes) on the model performance (−14.75%). In addition, the *Commitment* feature class also shows the best results when it is the only feature class used for prediction. Finally, we notice that the user features have the second largest impact on the model performance when dropped (−7.80%). This suggests that the model performance could be further improved by tailoring it to individuals via personalization or by customizing the model to specific user segments based on engagement and other behavioral signals.

Overall, our findings suggest that we can accurately detect task completion in this setting and that all features contribute. However, the model does not heavily rely on a single feature group and can still perform well in the absence of any feature class. We now discuss our findings in more detail and present their implications.

## 5 DISCUSSION AND IMPLICATIONS

We have shown that automatic detection of task completion may be feasible at reasonable accuracies (~83%). This has implications for the design of assistive technologies to, for example, help people focus attention on unfinished tasks. Model accuracy is a reasonable starting point given that this is the first study specifically targeting automatic task completion detection. While careful user experience design and user studies are needed in fielding a system using task completion detection, it is a new area that will likely inspire further work by others. There is an important next step in understanding how users receive this technology and in determining how best to communicate the imperfect completion inferences in ways that help not hinder users. There are many options to do this and Figure 1 offered just one example of a potential user experience. We expect that false positive rates will decline over time as more sophisticated methods are developed and datasets with richer contextual signals

**Table 3: Performance of model variants after dropping one feature class versus all features. Variants are ranked (asc) by F1. Significance testing uses two-tailed t-tests, differences vs. All Features (in F1) as: * p < 0.05, ** p < 0.01.**

| Model | F1 | % Δ | Acc | % Δ |
|---|---|---|---|---|
| All feature classes | 81.37 | — | 80.48 | — |
| — Commitment** | 69.38 | −14.75% | 69.60 | −13.52% |
| — User** | 75.03 | −7.80% | 74.96 | −6.86% |
| — Time** | 80.66 | −0.86% | 79.62 | −1.07% |
| — Email* | 80.95 | −0.52% | 80.02 | −0.58% |
| — Commitment Text | 81.13 | −0.29% | 80.19 | −0.37% |
| — Notifications | 81.44 | +0.08% | 80.53 | +0.06% |

**Table 4: Performance of model variants trained on one feature class versus utilizing all features. Variants are ranked (desc) by F1. Significance testing uses two-tailed t-tests, differences vs. All Features (in F1) as: * p < 0.05, ** p < 0.01.**

| Model | F1 | % Δ | Acc | % Δ |
|---|---|---|---|---|
| All feature classes | 81.37 | — | 80.48 | — |
| Commitment Only** | 71.01 | −12.72% | 71.06 | −11.70% |
| User Only** | 66.74 | −17.98% | 71.61 | −11.02% |
| Email Only** | 64.37 | −20.89% | 62.67 | −22.13% |
| Commitment Text Only** | 60.20 | −26.02% | 61.18 | −23.98% |
| Time Only** | 59.26 | −27.17% | 59.84 | −25.64% |
| Notifications Only** | 28.45 | −65.04% | 54.55 | −32.22% |

(e.g., calendar appointments, search queries, visits to businesses) become available, with user consent. When algorithms are deployed, we would need to collect data on their efficacy (e.g., fraction of clicks on "Not completed" in Figure 1) to both improve model performance and tailor them to individual users or user cohorts.

Task completion detection is a new challenge in machine learning and data mining, and as such there are no established baselines. That being said, there is strong signal in the features used, with accuracy values well above the marginal, which is just over 50%. We experimented with several learning algorithms with varying complexities and highlighted that more complex models (Neural Networks) could yield better performance but also that simpler models (Gradient Boosted Trees) could yield very good performance. These simpler models are inexpensive to implement and run in milliseconds [35] and hence can be run at scale over billions of emails and other task sources.

We focused on a notification scenario as a way of motivating the label generation methodology and grounding the evaluation in a meaningful application of task completion detection (i.e., suppressing notifications for completed tasks). Notifications is one of many possible applications of task completion detection (others include task prioritization and task deprecation [as in Figure 1]). We need to work with system designers and users to understand the scenarios where this technology can be most useful, e.g., for workflow/protocol management in socio-technical systems [15], where people are often required to report task completion manually.

For notifications, we did not consider the many factors that impact a decision to notify a user, including the current task context [18] and the notification priority [46]. In practice, completion detection could be a *component* in an interruptive notifications pipeline, perhaps serving to gate notifications from reaching ranking, rendering, and alerting components further down the pipeline. We also focused on a specific type of task: inferred tasks (commitments) mined from email. We need to experiment with user-defined tasks and different task types, including incoming requests (e.g., "Can you share the report?"), to-do list items, and document comments, to determine the generalizability of our methods. Each domain has its own signals and definitions of completion and those need to be considered when developing completion detection models.

The user features had an outsized impact given the small number of user features used in the study. Future work could explore segmenting users based on engagement levels, as there may be behavioral traits (e.g., diligently marking tasks as complete) which lead to differences in detection performance for different subclasses of users. The effectiveness of the user features also suggests that personalized task completion detection models would be worth exploring. Beyond current features in the *User* class (e.g., average completion time), there are likely other signals in personal activities, as well as general character traits, that could enable completion detection to be tailored to specific individuals. For example, users who typically respect deadlines may be more likely to complete new tasks before a deadline. This trait could be inferred from historic task completion activity with respect to extracted due dates.

Our focus in this paper has been on detecting task completion. A more general problem is task *progression* (measuring progress toward completion, of which task completion is a special case). One way to do this could be based on the amount of time that the task is estimated to take to complete (from distributional duration data [61]) or from historic data for a user, and the time that user has spent on the task so far [54]. Further work is needed to understand the feasibility of progression detection, including how intelligent systems should adapt to individual differences in task behavior.

## 6 CONCLUSIONS AND FUTURE WORK

Detecting task completion is an important challenge with several potential applications to help people focus on what needs their attention (versus what has already been accomplished). The success of the methods presented in this paper is promising for the enhancement of digital assistants and other intelligent systems with task completion detection capabilities. This can be employed for many applications, ranging from notification suppression to task prioritization. As the first study to introduce the task completion detection challenge, there are numerous opportunities for next steps. Future work will explore enhancements to the machine-learned models created for this task (more sophisticated learning algorithms and richer features), new data collection and integration efforts to obtain more signals related to task completion (e.g., visits to businesses, search queries) with user consent, and an expansion to other applications and task types. User studies are needed to understand how users react to the receipt of this type of task support in practice in different domains (including the impact on time spent and effort expended), especially those such as task auto-deprecation (e.g., as illustrated in Figure 1), where the effect of the task completion inference visibly alters the user experience.

## REFERENCES


[1] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*. 19–26.

[2] Eugene Agichtein, Ryen W White, Susan T Dumais, and Paul N Bennett. 2012. Search, interrupted: understanding and predicting search task continuation. In *SIGIR*. 315–324.

[3] David Allen. 2015. *Getting things done: The art of stress-free productivity*. New York, NY: Viking.

[4] Anne Aula, Rehan M Khan, and Zhiwei Guan. 2010. How does search behavior change as search becomes more difficult?. In *SIGCHI*. 35–44.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473* (2014).

[6] Brian P Bailey and Joseph A Konstan. 2006. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior* 22, 4 (2006), 685–708.

[7] Ling Bao and Stephen S Intille. 2004. Activity recognition from user-annotated acceleration data. In *PerCom*. 1–17.

[8] 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 *SIGCHI*. 735–742.

[9] Paul N Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *SIGIR*. 585–586.

[10] Ralf Bierig, Jacek Gwizdka, and Michael J Cole. 2009. A user-centered experiment and logging framework for interactive information retrieval. In *SIGIR Wksh on Understanding the User*. 8–11.

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

[12] Roger Buehler, Dale Griffin, and Michael Ross. 1994. Exploring the" planning fallacy": Why people underestimate their task completion times. *J. of Personality and Social Psychology* 67, 3 (1994), 366.

[13] Katriina Byström and Kalervo Järvelin. 1995. Task complexity affects information seeking and use. *IP&M* 31, 2 (1995), 191–213.

[14] Walter A Cook. 1979. *Case grammar: development of the matrix model (1970-1978)*. Georgetown University Press.

[15] Robert Cooper and Michael Foster. 1971. Sociotechnical systems. *American Psychologist* 26, 5 (1971), 467.

[16] Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. 2004. Task-focused summarization of email. *Text Summarization Branches Out* (2004).

[17] Stephen R Covey. 1989. *The 7 habits of highly effective people*. Simon & Schuster New York, NY.

[18] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. 2004. A diary study of task switching and interruptions. In *SIGCHI*. 175–182.

[19] Richard W DeVaul, Brian Clarkson, et al. 2000. The memory glasses: towards a wearable, context aware, situation-appropriate reminder system. In *SIGCHI Wksh on Situated Interaction in Ubiquitous Computing*.

[20] Abdigani Diriye, Ryen White, Georg Buscher, and Susan Dumais. 2012. Leaving so soon?: understanding and predicting web search abandonment rationales. In *CIKM*. 1025–1034.

[21] Doug Downey, Susan Dumais, Dan Liebling, and Eric Horvitz. 2008. Understanding the relationship between searchers' queries and information goals. In *CIKM*. 449–458.

[22] Gilles O Einstein and Mark A McDaniel. 1990. Normal aging and prospective memory. *J. Experimental Psychology* 16, 4 (1990), 717.

[23] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *ACM TOIS* 23, 2 (2005), 147–168.

[24] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[25] Peter M Gollwitzer. 1993. Goal achievement: The role of intentions. *European Review of Social Psychology* 4, 1 (1993), 141–185.

[26] David Graus, Paul N Bennett, Ryen W White, and Eric Horvitz. 2016. Analyzing and predicting task reminders. In *UMAP*. 7–15.

[27] Ahmed Hassan, Rosie Jones, and Kristina Lisa Klinkner. 2010. Beyond DCG: user behavior as a predictor of a successful search. In *WSDM*. 221–230.

[28] Ahmed Hassan, Xiaolin Shi, Nick Craswell, and Bill Ramsey. 2013. Beyond clicks: query reformulation as a predictor of search satisfaction. In *CIKM*. 2019–2028.

[29] Eric Horvitz, Andy Jacobs, and David Hovel. 1999. Attention-sensitive alerting. In *UAI*. 305–313.

[30] Eric Horvitz, Carl Kadie, Tim Paek, and David Hovel. 2003. Models of attention in computing and communication: from principles to applications. *CACM* 46, 3 (2003), 52–59.

[31] Shamsi T Iqbal and Eric Horvitz. 2010. Notifications and awareness: a field study of alert usage and preferences. In *CSCW*. 27–30.

[32] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*. 133–142.

[33] Daniel Kahneman and Amos Tversky. 1977. *Intuitive prediction: Biases and corrective procedures*. Technical Report. Decisions and Designs, McLean, VA.

[34] Ece Kamar and Eric Horvitz. 2011. Jogger: models for context-sensitive reminding. In *Conference on Autonomous Agents and Multiagent Systems*. 1089–1090.

[35] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3146–3154.

[36] Diane Kelly and Nicholas J Belkin. 2004. Display time as implicit feedback: understanding task effects. In *SIGIR*. 377–384.

[37] Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. 2016. Using dynamic and contextual features to predict issue lifetime in GitHub projects. In *Int. Conf. on Mining Software Repositories*. 291–302.

[38] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *WSDM*. 193–202.

[39] Alexander Kotov, Paul N Bennett, Ryen W White, Susan T Dumais, and Jaime Teevan. 2011. Modeling and analysis of cross-session search tasks. In *SIGIR*. 5–14.

[40] Carol C Kuhlthau. 1991. Inside the search process: Information seeking from the user's perspective. *JASIST* 42, 5 (1991), 361–371.

[41] Mik Lamming and Mike Flynn. 1994. Forget-me-not: Intimate computing in support of human memory. In *Int. Symp. on Next Gen Human Interface*. 4.

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

[43] Jingjing Liu and Nicholas J Belkin. 2010. Personalizing information retrieval for multi-session tasks: The roles of task stage and task type. In *SIGIR*. 26–33.

[44] Natalia Marmasse and Chris Schmandt. 2000. Location-aware information delivery withcommotion. In *Int. Symp. on Handheld and Ubiq Computing*. 157–171.

[45] EJ Masicampo and Roy F Baumeister. 2011. Consider it done! Plan making can eliminate the cognitive effects of unfulfilled goals. *J. Personality and Social Psychology* 101, 4 (2011), 667.

[46] Tara Matthews, Anind K Dey, Jennifer Mankoff, Scott Carter, and Tye Rattenbury. 2004. A toolkit for managing user attention in peripheral displays. In *UIST*. 247–256.

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

[48] Nina Mishra, Ryen W White, Samuel Ieong, and Eric Horvitz. 2014. Time-critical search. In *SIGIR*. 747–756.

[49] Dan Morris, Meredith Ringel Morris, and Gina Venolia. 2008. SearchBar: a search-centric web history for task resumption and information re-finding. In *SIGCHI*. 1207–1216.

[50] Hamid R Motahari Nezhad, Kalpa Gunaratna, and Juan Cappi. 2017. eassistant: Cognitive assistance for identification and auto-triage of actionable conversations. In *WWW*. 89–98.

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

[52] Johanna Peetz, Roger Buehler, and Anne Wilson. 2010. Planning for the near and distant future: How does temporal distance affect task completion predictions? *J. Experimental Social Psychology* 46, 5 (2010), 709–720.

[53] Xin Rong, Adam Fourney, Robin N Brewer, Meredith Ringel Morris, and Paul N Bennett. 2017. Managing uncertainty in time expressions for virtual assistants. In *SIGCHI*. 568–579.

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

[55] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. 2014. *Plan, activity, and intent recognition: Theory and practice*. Newnes.

[56] Arthur R Taylor, Colleen Cool, Nicholas J Belkin, and William J Amadio. 2007. Relationships between categories of relevance criteria and stage in task completion. *IP&M* 43, 4 (2007), 1071–1084.

[57] Pertti Vakkari. 2003. Task-based information searching. *Annual review of information science and technology* 37, 1 (2003), 413–464.

[58] Martijn H Vastenburg, David V Keyson, and Huib Ridder. 2008. Considerate home notification systems: a field study of acceptability of notifications in the home. *Personal and Ubiquitous Computing* 12, 8 (2008), 555–566.

[59] Dan Wang and Jesse Hoey. 2017. Hierarchical Task Recognition and Planning in Smart Homes with Partial Observability. In *Int. Conf. on Ubiquitous Computing and Ambient Intelligence*. 439–452.

[60] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *WWW*. 1353–1364.

[61] Ryen W. White and Ahmed Hassan Awadallah. 2019. Task duration estimation. In *WSDM*. 636–644.

[62] Ryen W White, Mikhail Bilenko, and Silviu Cucerzan. 2007. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*. 159–166.

[63] Ya Xu and David Mease. 2009. Evaluating web search using task completion time. In *SIGIR*. 676–677.