Task Minder: An Intelligent Task Suggestion Agent

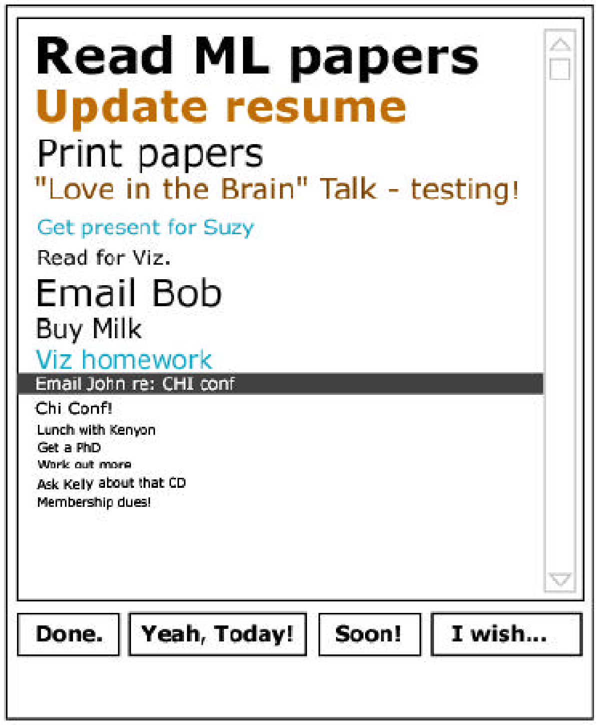
Zach Pousman, Brian Landry, Rahul Nair, Manas Tungare
CS 8802B
Georgia Institute of Technology
{zpousman,blandry,rnair,manas}@cc.gatech.edu

Introduction

Tasks are a fact of life. Nearly, everyone keeps a task list of some sort or other and these are some of the most personal information structures -- you could not make sense of someone else's list. An ML-powered agent could be implemented to display your task list and make smart suggestions.

The Task Minder attempts to create an order-of-magnitude leap forward in task management tools that exist today. The tools that exist today are suboptimal for a variety of reasons (including task entry and categorization). But fundamentally, they fail because they do not know very much at all about the user.

We seek to design an adaptive, personal "Task Minder" that will apply machine-learning algorithms to the process of task suggestion and prioritization.



Read ML papers
Update resume
Print papers
"Love in the Brain" Talk - testing!
Get present for Suzy
Read for Viz.
Email Bob
Buy Milk
Viz homework
Email John re: CHI conf
Chi Conf!
Lunch with Kenyon
Get a PhD
Work out more
Ask Kelly about that CD
Membership dues!

Figure 1: Screen prototype of Task Minder interface

We plan to look at the following features of tasks to help create this priority-list of tasks and make smarter suggestions about what a user could do next:

Time and Space Data

We suggest that certain tasks can only be done specific locations or at specific times, say, on Tuesday or only at the CoC. Our system will model user movements through space to learn where certain tasks are performed. We plan to track users on Georgia Tech's Campus via software that locates users based on their use of the LAWN network.

Start-time and End-time

We seek to leverage statistics about when tasks were entered into the system and when they were marked by a user as "completed." This, we hope, can be used to tell how long tasks take (on average) and possibly group or rank the tasks by this criterion for suggestions based on time-available. This is hard because we don't want to constantly bother users. How and even if this will be accomplished is still to be worked out.

Statistical properties of task entries

The system will do some statistical analysis of the text of task entries to find patterns and similarities. We would use these, if they prove to be useful, in categorizing tasks into dynamically generated task domains (i.e., categories)

Conversational Model of Interaction

Though it is not entirely worked out how we'll do it, we feel that it would be best to provide feedback to the tool that is as rich as possible without making users crazy. We are thinking of a way to infer concepts like "priority" / "urgency" through a software interface that encourages the user to elegantly and easily tell the system what matters and what doesn't. As Figure 1 shows, users can quickly see what tasks are on their list, select an element and click on the buttons below. In the figure above, "Email John re: CHI" is highlighted. The tasks are ordered by their current weighting in the system, from high to low. So the system thinks that the next thing for you to do is "Read ML papers". Since you're in the CoC and that's where you do muc h of your computing, the system suggests updating your resume and printing papers, two activities that are often done there. The colors and sizes represent some of the features of the system – color can be used to show you what tasks you have interacted with that day, as well as the tasks to which you've already said "Yeah, Today!"

Data Features

Time & Location

TaskMIND will determine the time and current location of the user to suggest a task that can be performed at that time and location. This context information is especially important since it can be gleaned without explicit entry by the user.

Time

Not all tasks take the same time to complete, and coupled with the fact that users have different amounts of free time during the day, this makes the time feature one of the most important that we can rely on. For example, when a user asks TaskMIND to suggest a task, the agent can run through its list of tasks to find one that can be reasonably expected to take roughly the amount of time that is available.

Location tracking will not help identify current activity, since the user can be doing one of several things while seated at his desk (i.e., in the same location).

Keyword Parsing

The tasks entered by a single user over a period of time will definitely have repeated keywords, and we can associate certain properties with each such keyword (for most keywords.)

For example, a task that contains the keyword "Email" can be reasonably expected to be completed in about 5 to 10 minutes. A task involving "Meet" takes around 20 - 30 minutes and cannot usually be performed at any location. "Call" tasks can be performed anywhere within about 10 minutes.

Although this idea seems to be strong enough to work, there are certain aspects that will need to be handled correctly: The tasks "Email John", "Meet John" & "Call John" all take different times in spite of the common keyword John. Initially, we expect the system to make incorrect suggestions, but as it receives more feedback, it will disassociate the keyword "John" from any of these expected features.

Previous interaction with the system

When the agent makes a suggestion, it receives feedback from the user on whether the task suggested was worth executing at that time or not. Such feedback lets the agent learn the user's preferences and encourages reinforcement learning on its part. Whether a user says "Wow, thanks" or "Not now" to a task will let the system know the degree to which the priorities assigned to tasks by the user and itself match (or not!).

Also, if the user says "Later" when the agent suggests a task, and this behavior is seen to be repeatedly true, the agent will learn not to suggest this task in the given set of circumstances. Positive reinforcement will occur when a user performs a suggested task immediately.

Due Date (entry optional)

Tasks can be assigned a due date. As the due date approaches, the task's priority weights can be adjusted so it pops up more often as a suggested task. Since we will also track when each task was entered and completed, it will also give a rough idea about when the user tends to complete certain tasks: too close to the deadline, or much before it. That can also lead to better task suggestions in the future.

Location

Similarly, in case of location, the machine will learn in due course of time, the regular schedule of the user. At 8:00p on a Monday evening, if the user asks for a list of tasks to do, the machine can reasonably infer that the user is or will soon be going to the grocery store and the tasks suggested will include "Buy milk", "Buy bread" and the like.

Both these features can be determined with the help of existing technology. Tracking time is a trivial task: every mobile device has a clock built-in -- what needs to be developed is keeping track of the time when tasks were added to the list, when they were suggested by the agent, when they were executed, and the frequency with which similar tasks appear.

There is no universal technology for tracking location that works in all environments. Thus, we need to use multiple technologies and limit the regions within which location tracking can be enabled. We are investigating the approaches outlined below.

Wireless LAN -based location tracking

A research project by another group suggests how to use the signal-to-noise ratio of wireless LAN signals to pinpoint a user's location to a resolution of about 10 feet. [Typically, an indoor wireless LAN uses multiple access points; the s/n ratios obtained from a mobile device to each access point is a function of the relative distance between them. An analysis of such ratios recorded over a period of time can allow pinpointing a user's location.]

GPS-based systems for outdoor tracking

Currently, GPS-enabled mobile devices are increasingly being used to perform reliable location-tracking. This can track large-scale changes in location, though it is known to be ineffective within buildings. To track locations such as grocery stores and car wash garages, there is no alternative to GPS.

RF-ID device on the agent

We plan to investigate the use of RF-ID tags in environments where such location-sensing is supported.

Desktop activity tracking

A user's activity on his desktop computer can be tracked by bugging his/her software! In technical terms, we plan to use a proxy server. Whether the user is engaging in writing/replying to email can be found by a proxy installed on the user's machine. Although this can track each and every email sent out, [which means we can track whether the task "Email John" was completed or not], we do not need information at such low granularity. All we need to know is that the user is (or is not) accessing his or her email, so a possible task suggestion would be to write another email to someone. Similarly, a user's web activity can be used to infer that he currently has Internet access, hence tasks such as "Read CHI paper" can be performed then. Of course, this data collection by the proxy occurs without intervention by the user.

HCI/UI Design Issues

The presentation system from the Task Minder is crucial for its functionality. Our main goal is to create an optimal list of tasks and then present it in a way that users can easily and quickly utilize it to "**get work done**". A secondary motivation is to create a user interface that lets users provide a host of feedback to the system (feeding our reinforcement-learning model). We will also attempt to support a few explicit actions – helping users to understand their agent and modify its parameters accordingly.

We propose building two UI modes for the system. First, **a complete list of incomplete tasks**. This view assumes that users need to see their task list at an overview level. Second, we also propose a **suggestion view**, where a single task (or top few tasks) is presented to the user on a small device. These suggestions, we hope, would be polled by a user in the 10 minutes before a class to see if a short-duration item could be accomplished in that time, or just to remind the user of a task to be accomplished.

Our vision of the interface is one that has some **"conversational" elements**. We don't intend to use a natural language interface, where users interact by typing commands and asking questions. However, we hope to include conversational interface elements to get at the fundamental modes of dealing with tasks. Instead of "done" and "not done," we plan to use a small number (possibly 4) of buttons to get at context-sensitive task management. Some of our ideas after a bit of brainstorming include (from certain to uncertain and from urgent to optional):

Done
Yeah, today!
Not today!
This week (or perhaps "Soon")
I wish

The interface will be fast and elegant in its operation. It will support browsing and visual filtering and should have only a few interface elements. The interface should prominently show the tasks themselves, and should not be cluttered with UI control widgets. This is even more important on small form-factor devices. The interface will be consistent and, though the system will learn about the user and will reorganize, re-order, and refresh the task list, the interface itself will remain constant.

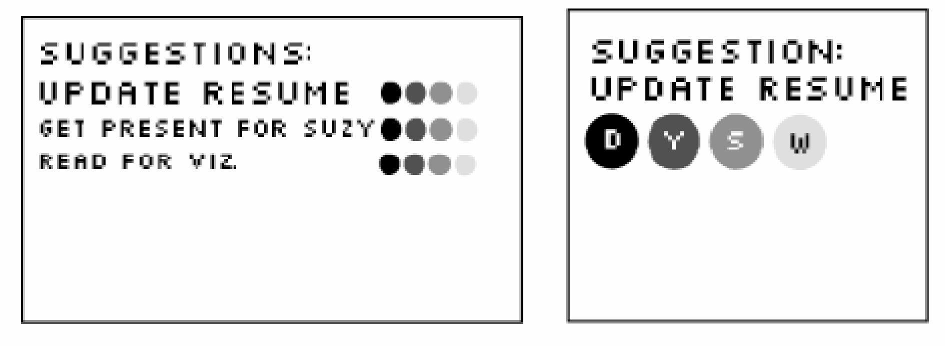


Figure 2: Possible lightweight interface for Cellphone or PDA. One or a small number of suggestions are presented. Users have intuitive and small interface choices, supporting on-the-fly use.

Machine Learning Issues

There are a number of machine learning techniques that will play a part in the success of the Task Minder. Location is one of the main features we intend to collect data about. A user's location gives us a lot of information about the context in which certain tasks are done. For example, knowing that the user is on an airplane lets us make inferences about the availability of computing resources. It would not make sense to suggest browsing the web when the user does not have an Internet connection. In addition, location shows us the location preferences users have for accomplishing tasks (e.g. email in class). We plan to track the location of our users by logging their connection to wireless access points. While this gives us an ample amount of data, from the data alone we cannot discern what sets of data points represent a specific place. To figure out the places the user does work, we will use a machine learning technique called clustering. This technique will allow us to look at the places the user has been over time and classify groups of points into places and paths. Although a user may connect to a number of different access points in the CoC Commons, all of those different data points still represent one place.

Places again are important to us because we can monitor the types of activities the user performs (e.g. writing, web surfing, etc.) in these locations. Along with the location we can look at time spent doing the various tasks to get an idea of how long the user takes to accomplish certain tasks. From this we can begin assigning probability weights to tasks currently in the system based on the user's location and time of the day. The use of probability weights prioritizes the task list based on the current location and the time available. This is important because even though reading a paper may be a good option for the allotted time period, the user may have the preference of reading in the morning rather than in the evening. For example, if a user is fifteen minutes early for class and wants a suggestion on what to do, the priority of the task will be different from when earlier the user had two hours and was looking for a suggestion.

So that the system will adapt over time, we will use a conversational interface to give feedback on the quality of the suggestions it makes. The interface as described above will allow the user to give feedback about the value of a suggestion. We are also considering letting the user give additional feedback by changing the weights of a tasks. Task completion gives positive reinforcement to the system while postponement gives negative reinforcement.

One problem that will take some time to work out is how new tasks entered into the list will be classified. Users tend to describe their tasks using systems that may seem unclear to others but makes since to the user. Many times task lists are used as reminders of what we need to accomplish. They say very little about the specifics of each task. In many cases the task list entry may only have meaning to the creator (e.g. MiniDV in Figure 1). This ambiguity makes it hard to decide what type of task has been entered complicating its association with the needed amount of time and the appropriate weight. Under the assumption that users are fairly consistent with the language they use to describe a task, our current plan is to parse the text of each entry and create dynamic categories (e.g. time required) for classifying tasks. These categories would be based on the time and resources needed for the task.

Phases and Deadlines

Phase 1: [From now till Feb 17]

• ***Gather data***

Location
Time
Proxies (collect email events and web access)
Paper task tracking

• ***Data Clustering***

Play with ML algorithms

Phase 1.5

• Develop the architecture for the suggestion agent

Phase 2 [Feb 17 - Feb 28]

• Build the suggestion agent
• Run the suggestion agent with test data

Phase 3

• Testing
• Live with the prototype